Data-guided Estimation and Tracking Methods for Unmanned Aerial Vehicles

Soumya Vasisht

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

2019

Reading Committee:

Mehran Mesbahi, Chair

Kristi Morgansen

Behçet Açikmeşe

Santosh Devasia

Program Authorized to Offer Degree:
Aeronautics & Astronautics

University of Washington

Abstract

Data-guided Estimation and Tracking Methods for Unmanned Aerial Vehicles

Soumya Vasisht

Chair of the Supervisory Committee:
Professor Mehran Mesbahi
William E. Boeing Department of Aeronautics & Astronautics

Autonomous aerial robots provide new possibilities to study interesting phenomena and offer a unique vantage point for many surveillance and tracking tasks. Tracking a rogue or an unknown target is an important task in which an agent typically adopts a reactive strategy to the changes reflected in the target observations. As these aerial vehicles increasingly share airspace with fixed wing commercial airplanes, it has become critical to establish reliable, high quality tracking strategies. This work seeks to leverage the concepts of modern control theory, statistics and reinforcement learning to enhance traditional tracking control design strategies to achieve improved tracking performance. A data-guided approach is proposed which shows that embedding observation data in to the control loop improves tracking performance for certain classes of target systems. A comparative study of model-based and model-free approaches for tracking is presented in which an agent, guided by vision-based sensors, directly learns an optimal policy to track the unknown reference trajectory. In addition, a distributed framework is developed in which multiple agents perform consensus on the learned parameters to improve tracking accuracy. Numerical simulations are presented to validate this data-guided tracking scheme for a single agent and a network of agents.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

The author wishes to express her sincere gratitude and appreciation to her advisor Prof. Mehran Mesbahi for his tireless encouragement, immense patience and for all the interesting discussions and stimulating ideas. His guidance, optimism and constant support renewed her confidence during moments of self-doubt and actuated her drive to bring her PhD to fruition. His calm demeanor, his sense of humor and the genuine delight he takes in his work are qualities she aspires to retain.

She extends her deepest thanks to the rest of her doctoral committee Profs. Kristi Morgansen, Behçet Açikmeşe, Santosh Devasia and Uri Shumlack for their guidance and invaluable comments. All her colleagues at the RAIN lab starting with alumni Drs. Marzieh Nabi, Airlie Chapman, Eric Schoof, Unsik Lee, Prachya Panyakeow, Ran Dai, Saghar Hosseini, Mathias Hudoba de Badyn to her current lab-mates Siavash Alemzadeh, Mengyuan Wang, Dillon Foight, Niyousha Rahimi, Thomas Miesen, Aditya Deole, Jingjing Bu, Taylor Reynolds, Shahriar Talebi and Bijan Barzgaran have been a constant source of inspiration to her through the years. She is grateful for the conversations and discussions, research related or otherwise. She is privileged to have spent her PhD years with such pioneers in their respective fields.

She expresses her deepest gratitude to Wanda Frederick, Ed Connery and Danyel Hacker for their help with all the logistical details and for always having her back. She conveys special thanks to Hal Perkins for trusting her time and again to be on his teaching staff for more quarters than she cares to remember. Her time as his teaching assistant taught her many bankable skills and she is very grateful for every

opportunity.

She wishes to thank all her friends in Seattle, especially Aditya Vashishta, Saumya Sinha and Vikash Kumar, for sharing their own journey with her, for enriching her with their company, for always willing to lend a sympathetic ear and offer what advice they could.

Above all, she is deeply indebted to her family, particularly her husband, Vivek, her companion and cheerleader, for his unrelenting support, patience and understanding, for guiding her with a perfect balance of affection and austerity. She is aquiver with anticipation of things in store for them. She expresses heartfelt thanks to her grandmother (Ajji) for her prayers and blessings, her parents (Amma and Anna) for their ceaseless faith and pride in her and for celebrating even her smallest achievements, her parents-in-law (Ma and Bada) for their unconditional love and trust, her little sister Sony for being her parent when the need arose and her brother-in-law Soma for being a great reliable friend. She is humbled by everyone's generosity and feels blessed to have them in her life.

DEDICATION

For my family and my dearest Vivek.

You give me wings and so I promise to fly.

Chapter 1

INTRODUCTION

Research on Unmanned Aerial Vehicles (UAVs) has increased many folds over the past decade. With the introduction of compact and more sophisticated hardware and software, many tasks deemed too ambitious or challenging for these vehicles just a few years ago, are within reach and are achieved rather impressively. Dangerous and mundane tasks can be autonomously performed with little to no human intervention. Beyond passive aerial surveillance, UAV applications have expanded to include real-time autonomous tracking, monitoring and responding to unknown or familiar targets and events [3, 4]. A well-known application of autonomous flight is surveillance and tracking of mobile targets guided by visual sensors [5–8]. The targets may be mobile ground or aerial robots with unknown dynamics or some time-evolving physical features in the target search area like traffic management, crowd behavior and movement, disease monitoring in plants, etc. One of the prime examples of such applications include intruder tracking, where an unauthorized agent enters a restricted area [9–11]. The primary task in this tracking problem involves maintaining continuous knowledge of the target's partial or full-state through measurement and prediction. This has inspired a rich field with numerous different techniques, spanning multiple science and engineering disciplines, including biological and chemical processes, underwater, ground and aerospace robotics, among many others.

Tracking capabilities of an aerial vehicle depend on a number of factors such as the environment in which the vehicle operates, availability of measurements, quality of the computer and the instruments on board, the dynamics of the target, the amount of sensor noise, etc. Classical control techniques grounded in strong mathe-

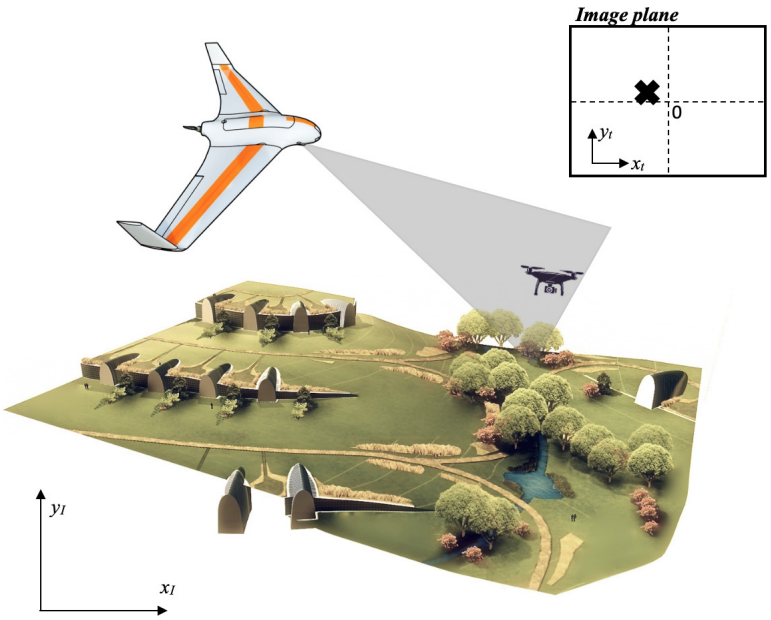


Figure 1.1: Sketch of the target tracking problem

matical foundations have paved the way for more heuristic approaches with certain relaxations, where system or control parameters and characteristics can be learned through interactions with the environment. Machine learning-based techniques with some performance guarantees are increasingly being used to control autonomous systems like aerial vehicles when the underlying equations of motion are too complex or very difficult to model from first principles or when limited information is available about the system or the environment. Typically, large amounts of input-output data is collected to fit a system model or to learn policies for a particular task. However, input data to a remote target is almost always unavailable. In applications such as intruder tracking, the state of the target in question is unknown and has to be discovered through measurements. We may also, however, have access only to partial state information through observations.

Ideally, in tracking, an agent's output is expected to follow a signal from a certain class of trajectories. The target motion, however, may be a result of its linear or

nonlinear dynamics. Nonetheless, short intervals of the observed trajectory may be closely modeled by an approximate function, such that the output of the tracking agent follows the response of this artificial target system. Access to such information is unlikely when we encounter a target that is previously unknown. We can, however, devise a way to compute a function that captures the target's behavior in the form of a reference model.

The linear quadratic (LQ) tracking problem is an extension to the standard linear quadratic regulator (LQR) problem obtained by penalizing the state deviations from some desired reference trajectory. Consider the environment in which a tracking agent, in our case, the unmanned aerial vehicle (UAV), adopts a reference tracking policy and is given access to the target's position at the current time step. This measurement, however, is available to the UAV only when the target is close enough to the UAV, such that the on board camera is able to capture the target in its field of view or vision. A potential issue emerges when the target has faster dynamics than the UAV and performs aggressive maneuvers or is being evasive. Another common problem is the presence of measurement errors, usually induced due to erratic sensors. Such errors are most commonly modeled as gaussian noise and are a factor of the sensor characteristics with zero mean and some known variance.

Regardless of the target's dynamics and noise-ridden measurements, the target motion may be observed to derive its inherent behavior to learn a better strategy to improve tracking. In this dissertation, we examine the efficacy of using past observation data for tracking unknown targets in the region of interest. We devise a way to capture the target's dynamic behavior from the time-series data of its position measurements for direct use in the tracking control law design. Our results show that incorporating this learned model-following approach can indeed improve tracking performance as compared with standard LQ tracking methods with position error feedback for a variety of target systems, even in the presence of noise.

Next, we observe that model-free reinforcement learning (RL) techniques can also

provide a way to learn a policy when there is uncertainty in the UAV and the target state. RL methods allow a UAV or a team of UAVs to learn and navigate through the changing environment without a model of the environment. We explore this domain to evaluate the trade-off between convergence rate and tracking accuracy. Here, the training information available to the algorithm arises as a result of following some trajectory through the state space and the functions being approximated by the algorithm are updated after every state transition.

To mitigate a single UAV's inability to maintain close proximity to the target, we consider deploying multiple UAVs to gather measurements, where each agent builds its own estimate of the target parameters and seeks to improve its estimate by exchanging information with a neighboring agent. This step of consensus may be performed with the learned target parameters or the values of the computed future target trajectory for a short time horizon. We explore these strategies for different classes of trajectories, in which the UAVs cooperatively track a moving target.

1.1 Literature Review

Target tracking is an important area of research and has been extensively studied in several disciplines. Several approaches have been explored for tracking with unmanned aerial vehicles. The Lyapunov vector field approach [12, 13] obtains stable circular loiter patterns to design various path-planning and tracking strategies. The Model Predictive Control (MPC) technique [14, 15] achieves set-point tracking with constraints and has been extensively studied. State estimation techniques involving the de facto standard, Extended Kalman filter (EKF), for nonlinear estimation provide estimates of the system states using measurements and a suitable target model while assuming a priori statistical models for the system and measurement noises. These techniques, however, suffer from problems such as data association, convergence, linearity and computational complexity [16]. These issues in turn can introduce large errors for highly nonlinear systems which may lead to sub-optimal performance and

sometimes divergence of the filter. There have been several other improvements on the EKF that also have their limitations, especially high computational cost for real time applications. The Unscented Kalman Filter (UKF) [17] overcomes some of these limitations by representing the probability distribution of the states defining the system with a set of samples, called sigma-points, instead of the covariance matrix. In this case, UKF avoids the need to compute the Jacobian leading to similar computational load as the EKF.

System identification methods, such as the Eigensystem Realization Algorithm (ERA) and the Observer Kalman Filter Identification (OKID), were developed to aid in the discovery of input-output models for systems with control [18, 19]. The dimension of the measurements, though, were assumed to be low, the system linear for a known excitation signal. Additionally, when input data to a remote target is almost always unavailable, which is also the case with our tracking problem at hand, modal analysis techniques which perform output-only system identification will be more appropriate. In applications such as intruder tracking, the state of the target in question is unknown and has to be discovered through measurements. We may, however, have access only to partial state information through observations.

There is recent interest from the machine learning community in data-driven control and non-asymptotic analysis. Putting aside the reinforcement learning literature and restricting our attention to linear state-space models, the work in this area can be divided into two categories: (i) learning the parameters of the system model from limited data [20–23], (ii) directly learning the control inputs to optimize a control objective or analyzing the predictive power of the learned representation [24, 25]. The former problem aims to learn a general purpose model that can be used in different control tasks, for instance, by combining it with robust control techniques [21, 22, 26]. The focus here has been to analyze data–accuracy trade-offs. On the other hand, for the latter problem, the focus has been on exploration/exploitation type formulations and regret analysis. Since the goal is to learn how to control the system to achieve

a specific task, the system is not necessarily fully learned. Model-free reinforcement learning (RL) methods, when used online, have been shown to converge to an optimal policy even for continuous state and action spaces [27]. A great deal of research has been done to adapt incremental dynamic programming to problems with continuous state and action spaces.

Data-driven modal decomposition methods such as the proper orthogonal decomposition (POD) and dynamic mode decomposition (DMD) are useful in extracting dynamical features of the system from both experimental and numerical data. DMD has acquired popularity as a method for systems with nonlinear dynamics, due to a strong connection between DMD and the Koopman operator theory [28, 29]. Our analysis uses a recent characterization of DMD [30] for feature extraction to compute a finite-dimensional approximation of the Koopman operator, a linear operator that captures that dominant modes of any dynamical system in such a way that this approximation has the form of a linear uncontrolled dynamical system. The operator obtained this way may not necessarily be stable for use with LQR methods. A constraint generation approach can be used to stabilize the ensuing system approximation [1]. The effectiveness of data-driven approaches depends on a clean dataset that is not corrupted by noise, which is usually not the case with experimental data. One of the major advantages of using data-driven approaches like DMD is that it can be applied directly to data, without the need for the knowledge or construction of the system matrix; moreover, the effect of noise on DMD is often deterministic [31]. This not only allows us to accurately predict the effect of noise, but also allows for a correction to be implemented to recover the noise-free dynamics. In this work, we only address sensor noise during the measurement process and assume that the process noise is negligible.

Considerable work has been done in the general area of coordinated target tracking, with coordinated standoff tracking comprising the greatest body of work in this area. Standoff tracking with perfect knowledge of the target state has been studied

in [32] and [33] where the most prevalent control strategies involve the use of vector fields and nonlinear feedback. Approaches with only partial information of the target state are presented in [34] and [35]. The authors of these works utilize observers, adaptive control, and extended Kalman filtering to estimate the full target state.

1.2 Overview of Contributions

This work demonstrates the effectiveness of utilizing past observation data to track detected targets more accurately. In lieu of training a model offline from large of amounts of data, we explore a strategy that builds the necessary machinery during flight. The main contributions of this dissertation are summarized below.

1. Development of the data-guided control strategy that embeds data in an LQ tracking setup to enhance tracking performance for certain classes of target systems.

2. Evaluation of the real-time tracking performance using the Q – learning approach for model-free tracking.

3. Application of the data-guided approach to a distributed setup that cooperatively learns the target's behavior through consensus to improve tracking accuracy.

1.3 Organization of Thesis

In this chapter, we reviewed relevant research problems and solution strategies explored in related work to highlight the gaps in literature that the present work intends to close. Chapter 2 presents the necessary background and preliminaries to make the dissertation self-contained. The data-guided approach for a single agent tracking a target is introduced in Chapter 3. A model-free approach is explored in Chapter 4 to learn a control policy that drives the tracking error to zero. A distributed framework

is developed in Chapter 5 in which the agents exchange information about the target and update that parameters that each determined about the target motion to obtain a more accurate target representation. Finally, a brief summary of the results and additional discussion is presented in Chapter 6 along with directions for potential future extensions of this work.

Chapter 2

BACKGROUND AND PRELIMINARIES

2.1 The Aerial vehicle

The agent is assumed to be a small fixed-wing aircraft equipped with a gimbaled camera for visual sensing. Consider a linear time-invariant system to describe the UAV j, flying at a fixed altitude $\boldsymbol{h}^j$. The state vector for the UAV j is given by $x^j = [\mathbf{p}_a^j,\ \mathbf{v}_a^j]^\top \in \mathbb{R}^4$, with $\mathbf{p}_a^j = \left[x_1^j,\ x_2^j\right]$ and $\mathbf{v}_a^j = \left[\dot{x}_1^j,\ \dot{x}_2^j\right]$ representing the position and velocity of UAV j in $\mathbb{R}^2$. The discrete time instances $t_k,\ k \in \mathbb{Z}_{\geq 0}$, assuming a perfect clock with sampling period $T_s > 0$, is given as $t_k = kT_s$. The state of vehicle j at time step $k+1$ is

$$\begin{aligned} x_{k+1}^j &= A^j x_k^j + B^j u_k^j \\ y_{k+1}^j &= C^j x_{k+1}^j, \qquad\qquad x_0 \text{ given,} \end{aligned} \tag{2.1}$$

where

$$A = \begin{bmatrix} I_2 & T_s I_2 \\ 0 \cdot I_2 & I_2 \end{bmatrix} \in \mathbb{R}^{4\times 4}, \qquad B = \begin{bmatrix} (T_s^2/2) I_2 \\ T_s I_2 \end{bmatrix} \in \mathbb{R}^{4\times 2};$$

in this case assuming homogenous vehicles, $A^j = A$ and $B^j = B$ for every $j > 0$, I_n is a $n \times n$ identity matrix and $u_k^j \leq \bar{u} \in \mathbb{R}^2$ is the vector of acceleration inputs for some maximum absolute acceleration $\bar{u}$.

The terms agents, aerial vehicles, tracking agents and UAVs are used interchangeably throughout this text.

2.2 LQ methods for target tracking

Most of this section is an excerpt from [36]. Linear-quadratic (LQ) control is one of the most widely studied problems in control theory. It has been applied successfully to problems in statistics, econometrics, robotics, social science and physics. In recent years, it has also received much attention from the machine learning community, as increasingly difficult control problems have led to demand for data-driven control systems.

In LQ control, both the state and action are real-valued vectors. The dynamics of the environment are linear in the state and action, and are perturbed by Gaussian noise. The cost is quadratic in the state and control (action) vectors. The optimal control policy, which minimizes the cost, selects the control vector as a linear function of the state vector, and can be derived by solving the algebraic Ricatti equations.

As described in [36], the regulator problem is a special case of a wider class of problems where it is required that the outputs of a system follow or track a desired trajectory in some optimal sense. For the regulator, the desired trajectory is simply the zero state. However, we can apply the regulator theory and extend its results to solve a control problem that involves achieving a desired trajectory. Here is a brief description of LQ-based tracking.

Given a linear system with state equations as described in (2.1), the LQ tracking problem extends LQR by penalizing state deviations from some reference trajectory $\boldsymbol{r}_k$. It is assumed that the policy has access to the values of the reference trajectory for the time interval, $k_0 \leq k \leq T$, where T is some pre-defined time horizon. The finite horizon cost function is defined as follows

$$V(T, x_0) = \sum_{k=0}^{T-1} (x_k - \tilde{x}_k)^\top Q (x_k - \tilde{x}_k) + u_k^\top R u_k + (x_T - \tilde{x}_T)^\top P_T (x_T - \tilde{x}_T). \quad (2.2)$$

Recognizing that $\boldsymbol{r}_k$ is externally prescribed, the reference trajectory may be described

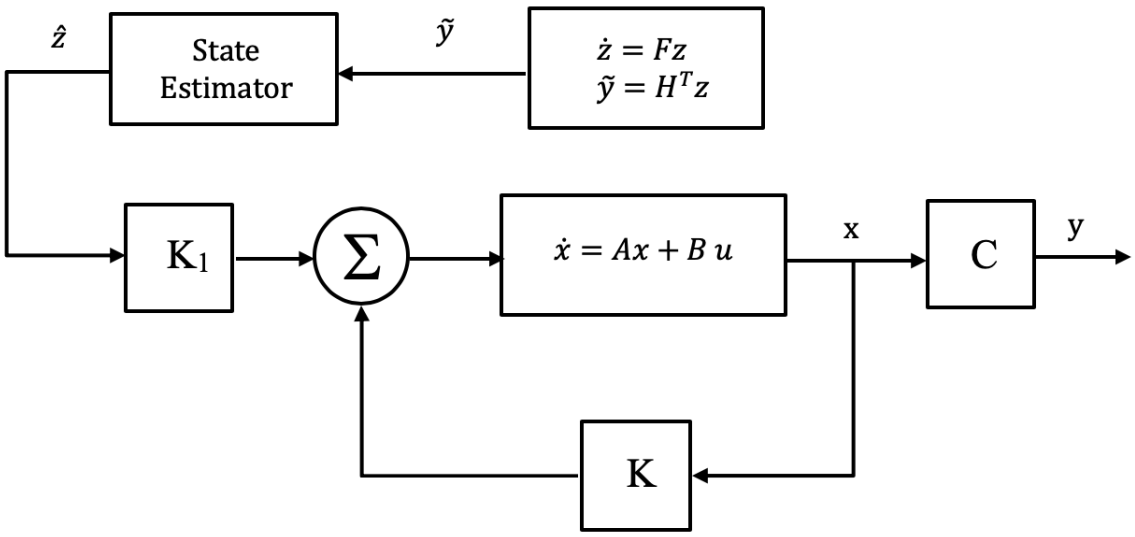


Figure 2.1: Optimal servo system

as the output of some known p-dimensional linear reference model

$$\dot{z} = Fz$$
$$\tilde{y} = H^\top z$$

for some initial state z_0. The pair $[F, H]$ is not necessarily assumed to be completely observable. The goal of this optimal tracking problem is to find the optimal control u^* for the system in (2.1), such that the output y tracks the incoming signal $\boldsymbol{r}$, minimizing the index given by (2.2), where Q is symmetric positive semi-definite and R is symmetric positive definite representing weighting matrices specifying appropriate penalties for the tracking error cost and control cost respectively.

The tracking problem may now be recast as a standard LQR problem by defining

an augmented state vector $\hat{x}_k = [x_k, z_k]^\top \in \mathbb{R}^{n+p}$ and associated state-space matrices

$$\hat{A} = \begin{bmatrix} A & 0 \\ 0 & F \end{bmatrix}, \ \hat{B} = \begin{bmatrix} B \\ 0 \end{bmatrix}$$
$$\hat{Q} = \begin{bmatrix} Q & -QLH^\top \\ -HL^\top Q & HL^\top QLH^\top \end{bmatrix} \tag{2.3}$$

Applying regulator theory to this new minimization problem, we get the optimal control u^*,

$$u^* = -R^{-1}\hat{B}^\top \hat{P}\hat{x},$$

where $\hat{P}$ is the solution of the Riccati equation

$$-\dot{\hat{P}} = \hat{P}\hat{A} + \hat{A}^\top \hat{P} - \hat{P}\hat{B}R^{-1}\hat{B}^\top \hat{P} + \hat{Q}, \quad \hat{P}(T) = 0. \tag{2.4}$$

The minimum index is

$$V^* = \left(\hat{x}_0^\top \hat{P}\hat{x}_0\right),$$

where $\hat{P}$ can be partitioned as

$$\hat{P} = \begin{bmatrix} P & P_{12} \\ P_{12}^\top & P_{22} \end{bmatrix},$$

which gives the new control law

$$u^* = Kx + K_1 z,$$

where

$$K = -R^{-1}B^\top P$$
$$K_1 = -R^{-1}B^\top P_{12}.$$

Here, P_{12} is determined by the reference model that determines the feedforward control law to minimize the index in (2.2) as a solution to the equation

$$-\dot{P}_{12} = P_{12}F + A^\top P_{12} - PBR^{-1}B^\top P_{12} - QLH^\top$$

with boundary conditions $P_{12}(T) = 0$.

2.3 Vision-based tracking

Each UAV is assumed to be equipped with a monocular camera which makes image-plane measurements of the target. The following description is applicable to every UAV, so we'll skip the superscript j in this section for better readability. In vision-based target tracking, the image processing software is responsible for determining the centroid pixel coordinates of the target moving in the image frame. Using these pixel coordinates, along with the intrinsic and extrinsic camera parameters and terrain data, one can estimate the three-dimensional location of the target in inertial coordinates and compute the associated error covariance. This vision-based measurement of the target's position is also referred to as the geolocation estimate. The error associated with the geolocation estimate is highly sensitive to the UAV's position relative to that of the target. As the UAV's planar distance from the target increases, the associated error covariance grows and becomes significantly elongated in the viewing direction. When a UAV is directly above the target, the measurement error is smallest, as the corresponding error ellipse is circular. Thus, a UAV would ideally hover directly above the target, but the relative dynamics between a fixed-wing UAV and a moving ground target typically preclude this viewing position from being maintained over a period of time. To mitigate a single UAV's inability to maintain close proximity to the target, one can employ multiple UAVs to gather measurements, which are then fused to obtain an improved geolocation estimate. This is referred to as cooperative (or coordinated) target tracking.

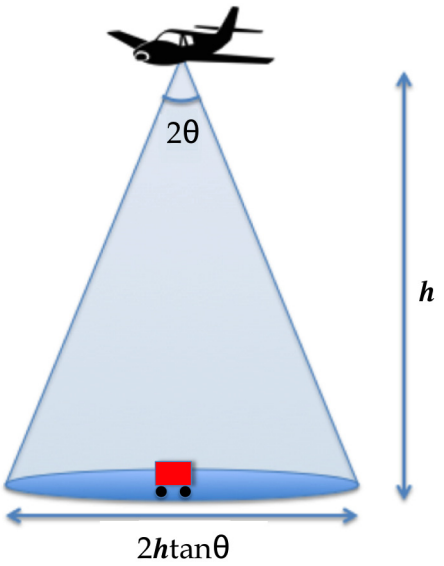


Figure 2.2: Camera field of view (FOV)

Definition 1. The sensor field of view (FOV) of the camera on board a given UAV is the half-angle θ in radians, that covers a search area whose radius is represented by the linear function $c(\boldsymbol{h})$,

$$c(\boldsymbol{h}) = \begin{cases} 0 & \text{if } \boldsymbol{h} < \boldsymbol{h}_0 \\ \boldsymbol{h}\tan\theta & \text{if } \boldsymbol{h}_0 \leq \boldsymbol{h} \leq \boldsymbol{h}_{max} \\ 0 & \text{if } \boldsymbol{h} > \boldsymbol{h}_{max} \end{cases}, \tag{2.5}$$

where $(\boldsymbol{h}_0, \boldsymbol{h}_{max})$ is the altitude range where visual based sensing is viable and yields reasonable resolution of the target for the image processing software to make target position estimates with low error covariances.

Alternatively, the projected planar position of the UAV, $\mathbf{p}_{a_k}$, is at the center of the circle with radius $c(\boldsymbol{h})$. The relative planar distance of a single UAV to the target can be given by

$$d_k = \|\mathbf{p}_{a_k} - \bar{y}_k\|_2 , \tag{2.6}$$

where $\bar{y}_k \in \mathbb{R}^2$ is the target position measurement in the inertial frame at time step k.

When $d_k = 0$, the UAV is directly above the target and the measurement error is the smallest. The target is "visible" to the UAV, if $d_k < c(\boldsymbol{h})$. As $d_k \to c(\boldsymbol{h})$, the associated error covariance grows and becomes significantly elongated in the viewing direction. The UAV attempts to minimize d_k, which is essentially the tracking error.

Our goal in this work is to transform the information in the image to an appropriate command signal for the UAV to follow in order to remedy the deviation from the target's output trajectory. We consider the the overall measurement model in vision-based target tracking. The sensor attitude matrix or a rotation matrix $T_S^T(\theta)$ relates the coordinates of the line-of-sight vector ℓ_S from the UAV to the target in the North-East-Down sensor frame (centered at UAV's position) to the coordinates of the same vector in the local East-North-Up topographic coordinate frame [14]. This transformation is a nonlinear function of the 3-2-1 Euler-angle sequence of yaw, pitch, and roll denoted by $\theta \in \mathbb{R}^3$. The image tracking software controls the camera's gimbal platform to keep the target in the center of the camera's field of view and reports the Euler angles of the camera sensor as well as the line-of-sight vector ℓ_S.

The 3-dimensional target position measured by UAV j with 3D position $\mathfrak{a}^j = \left[\mathbf{p}_a^j,\ \boldsymbol{h}^j\right]^\top$ is denoted by $\mathfrak{t}^j$. Its estimate is computed in [14] and is given by

$$\hat{\mathfrak{t}}^j = \mathfrak{a}^j + \hat{r}^j T_S^T(\theta^j)\ell_S^j = \mathfrak{a}^j + \hat{r}^j \hat{\ell}^j, \tag{2.7}$$

where $r^j = \|\mathfrak{t}^j - \mathfrak{a}^j\|_2$ and its estimate $\hat{r}^j$ is provided by the flat-Earth approximation $\hat{r}^j = (\boldsymbol{h}_0 - \boldsymbol{h}^j)/\hat{\ell}^{j,3}$, where $\boldsymbol{h}_0$ is the height of the ground plane in the topographic coordinate frame and can be taken as zero without loss of generality. We will also assume that the altitude of the target is always less than $\boldsymbol{h}^j$ for any j.

2.4 The Koopman operator and Dynamic Mode Decomposition (DMD)

A dynamical system has an associated Koopman operator, which encodes many important properties of the system. The Koopman operator $\mathcal{A}$ is defined for a dynamical system

$$\xi_{k+1} = f(\xi_k) \tag{2.8}$$

evolving on a finite-dimensional manifold M [28, 29], where $\xi_k \in M$ is the possibly unobserved state of the system. The dynamics need not be discrete in time, but we retain this form since we presume that the data to be analyzed are as such. In this direction, define an observation function

$$\psi_k = g(\xi_k), \tag{2.9}$$

where $g \in \mathbb{G} : M \to \mathbb{R}$ or $\mathbb{C}$; the Koopman operator now satisfies the relation,

$$\left[\mathcal{A}g\right](\xi) \triangleq g\left(f(\xi)\right). \tag{2.10}$$

Although the underlying dynamics may be nonlinear and finite-dimensional, the Koopman operator is a linear infinite-dimensional operator; the linearity is a result of the definition in (2.10), and not due to any linearization. Note that the Koopman operator maps elements in $\mathbb{G}$ to elements in $\mathbb{G}$, unlike f which evolves system states in M. Furthermore, note that

$$\left[\mathcal{A}g\right](\xi_k) = g\left(f(\xi_k)\right) = g\left(\xi_{k+1}\right).$$

This shows that the Koopman operator propagates the output of the system forward. It is straightforward to show that if the dynamics (2.8) are linear, with $f(\xi) = \mathcal{L}\xi$, then the eigenvalues λ_j of $\mathcal{L}$ are also eigenvalues of the Koopman operator [30].

Chapter 3

DATA-GUIDED AERIAL TRACKING

3.1 Introduction

Tracking an unknown target reference trajectory typically compels an agent to adopt a reactive strategy to the changes seen in the target observations. Embedding the time-series data obtained from target observations in to the control loop may enhance tracking performance, especially in the presence of sensor latency or occlusions due to the addition of a predictive component. In this chapter, we construct a data-guided approach that helps improve the tracking performance of an agent following an unknown mobile target, such as an intruder, for certain classes of target trajectories. The agent is assumed to be a small fixed-wing aircraft equipped with a gimbaled camera for visual sensing. We formulate a control policy by building an augmented state for a Linear Quadratic (LQ) tracking system reinforced by an approximated linear operator that indirectly captures the properties of the observed target system.

The tracking error depends on the quality of data collected during the measurement phase. Most recent works in learning-based control train their models on several trajectories over multiple iterations to achieve high-precision tracking [37]. These works either discuss methods to estimate the target's true position at each time step based on measurements from one or multiple sensors or assume a pre-defined structure for the target model and identify parameters to get the best fit. In our work, on the other hand, we build an artificial dynamical system that generates a reference signal to propagate the state of the UAV to match the target position as closely as possible.

Much of this chapter has been published in a publication by the author [38].

3.2 Problem Formulation

Consider the linear servomechanism problem in which an error-sensing feedback controller achieves asymptotic tracking or output regulation for a modeled class of reference and disturbance signals. We consider a case where the reference signal is revealed as a single way-point at each time step, is unmodeled and unknown a priori, but can be estimated in real-time. In order to accomplish this objective, the position of the moving target is observed at each time step as $\bar{y}_{k+1} \in \mathbb{R}^2$. The target may be evolving according some dynamics which may not be linear. It may also be driven by some time-varying control input. We will, however, approximate a linear operator that describes the target from partial state observations for a small time window and continue to update our estimate as more information becomes available.

In vision-based target tracking, it is generally assumed that the visual sensor specifications are known and the sensor noise is independent and normally distributed with zero mean and a given variance. A suitable transformation function is used to convert the target position coordinates from the sensor reference frame to the global inertial reference frame as described in Section 2.3. The measurement vector associated with the target after this transformation can be given as

$$\bar{y}_k = \tilde{y}_k + w_k,$$

where $\bar{y}_k \in \mathbb{R}^2$ and $w_k \in \mathbb{R}^2$ is the normally distributed, zero mean noise, that deviates from the true value, $\tilde{y}_k \in \mathbb{R}^2$ with a known variance σ_t^2. This noise represents the uncertainties arising from errors in the transformations and sensor inaccuracies.

3.2.1 Approximating the Koopman Operator

As described in [39], expressing nonlinear systems in a linear manner is a desirable property for many reasons. For example, Koopman eigenfunctions reveal state par-

titions along which the nonlinear dynamics evolve linearly. The ability to obtain geometric properties of nonlinear systems using the Koopman eigenvalues has drawn the attention of the scientific community. The global stability of a system using the eigenfunctions of the Koopman operator were investigated in [39], and the authors in [40] extend the local linearization around a stationary point to the whole basin of attraction.

Our aim is to obtain a generalizable approach to data-guided tracking that is only contingent on data and does not rely on prior knowledge of the target behavior. Barring the case where the target travels with a constant velocity, all other trajectories may be approximated by a linear time-varying operator that is routinely updated as more data is acquired. The constant velocity case simply converges to a realization of the original target system that produces the observed trajectory. The Extended Dynamic Mode Decomposition (EDMD) computes a finite-dimensional approximation of this linear infinite-dimensional Koopman operator [41] from the data. EDMD is data-driven and leads to an approximation of the leading eigenvalues, eigenfunctions, and modes of the Koopman operator defined for a dynamical system without the need for governing equations. Its predecessor, the DMD, was a method first described for analysis of fluid mechanical systems [42]. DMD operates on snapshots of the flow-field (e.g., velocity, vorticity, pressure) and their time-shifted counterparts - obtained either from experiments or numerical simulations - in order to compute the eigenvalues ("DMD eigenvalues") and eigenvectors ("DMD modes") of a linear predictor that best fits the associated dynamics in a least-squares sense.

As outlined in [38], in order to obtain sufficient data of target position observations, we incorporate a measurement phase at the beginning of the tracking process to build a set of snapshot data that records the target's evolution. The UAV spends a few, say m, time steps, where $m \geq n$ (we have $n = 4$)) in this measurement phase taking position measurements of the target, $\bar{y}_k$ with respect to the inertial frame at each time step k. These m *snapshots* of data can be grouped in to two data matrices denoted

by X and Y,

$$X = \begin{bmatrix} \bar{y}_1 & \bar{y}_2 & \cdots & \bar{y}_{m-2} & \bar{y}_{m-1} \end{bmatrix}, \tag{3.1}$$

$$Y = \begin{bmatrix} \bar{y}_2 & \bar{y}_3 & \cdots & \bar{y}_{m-1} & \bar{y}_m \end{bmatrix}, \tag{3.2}$$

where the columns of the matrix Y have a one time step shift with respect to the columns of matrix X, such that $X \cap Y = X \setminus \bar{y}_1 = Y \setminus \bar{y}_m \in \mathbb{R}^{2\times m-1}$. The time between snapshots may be fixed to be T_s or varying based on the application or the computational power on board the UAV. More importantly, the time between the snapshots or data samples may be irregular due to faulty sensors, occlusions or other malfunctions. This allows for measurements to be considered if and when they are available. This requires a careful selection of window size m, that is large enough to nullify the effects of irregular sampling.

We are not interested in the spectral properties of the dynamics, but rather in the time-domain prediction of the corresponding trajectories for the purpose of control. A decomposition algorithm like DMD will estimate a linear operator that best represents the data in X such that the columns of $X = \begin{bmatrix} \bar{y}_1 & \mathbf{A}\bar{y}_1 & \mathbf{A}^2\bar{y}_1 & \cdots & \mathbf{A}^{m-2}\bar{y}_1 \end{bmatrix}$ form the order $m-1$ Krylov subspace [43].

The linear transformation $\mathbf{A}$ maps the data at time k to the data at time $k+1$ such that $\bar{y}_{k+1} \approx \mathbf{A}\bar{y}_k$. Thus, $Y \approx \mathbf{A}X$. The first step of the DMD algorithm is essentially a least-squares step that is based on the singular value decomposition of the data matrix X, where $X = U\Sigma V^*$ with $U \in \mathbb{R}^{2\times 2}$, $\Sigma \in \mathbb{R}^{2\times m-1}$ and is diagonal and V^* is the complex conjugate transpose of $V \in \mathbb{R}^{m-1\times m-1}$, with $X, Y \in \mathbb{R}^{2\times m-1}$. We can compute the approximation $\mathbf{A} \approx YV\Sigma U^*$ and attempt to iteratively minimize the quantity,

$$\mathcal{A} = \ \text{argmin}_{\mathbf{A}} \left\| Y - \mathbf{A}X \right\|_F^2, \tag{3.3}$$

where $\|\cdot\|_F$ denotes the Frobenius norm for matrices.[1] The unique minimum-norm solution to this least-squares problem is given by

$$\mathcal{A} = YX^+, \tag{3.4}$$

where X^+ denotes the Moore-Penrose pseudo-inverse of X.

Here, we assume that X has a full row rank, in which case $XX^\top$ is invertible, and

$$X^+ = X^\top \left(XX^\top\right)^{-1}. \tag{3.5}$$

This assumption is valid in the case where the number of snapshots m is large, as compared with the state dimension n.

3.2.2 Preparing the Data: Scaling data matrices X and Y

The measurement phase of the UAV collects position measurements of the target motion with the onboard camera. The cost function (3.3) derives an operator that is piecewise linear along the observed trajectory. This time-varying parameter will be revised periodically in real time. In such a situation, we might wish to place more weight on recent snapshots, and gradually "forget" the older snapshots, by minimizing a cost function of the form

$$\tilde{J} = \min_{\mathcal{A}} \left\| \tilde{Y} - \mathbf{A}\tilde{X} \right\|_F^2, \tag{3.6}$$

where

$$\begin{aligned} \tilde{X} &= \left[\sigma^{k-1}\bar{y}_1 \sigma^{k-2}\bar{y}_2 \cdots, \bar{y}_{m-1}\right], \\ \tilde{Y} &= \left[\sigma^{k-1}\bar{y}_2 \sigma^{k-2}\bar{y}_3 \cdots, \bar{y}_m\right], \end{aligned} \tag{3.7}$$

[1] The Frobenius norm of matrix A, $\|A\|_F = \sqrt{\text{trace}\,(A^\top A)}$.

with $\sigma,\ 0 < \sigma \leq 1$, defined as a "forgetting factor" that places more weight on recent snapshots. This weighting scheme is analogous to that used in real-time least-squares approximation [44].

3.2.3 Sensor Noise

The target positions may have uncertainties arising from sensor noise and time delays with w_t representing the random noise vector whose elements are independent and normally distributed with zero mean and a variance σ_t^2. With the true measurement (ground truth) data X_g and Y_g, the data matrices as given in (3.1) and (3.2) may thus be represented as,

$$X = X_g + N_X \text{ and } Y = Y_g + N_Y,$$

with random matrices N_X and N_Y built from arranging the sensor noise vector associated with each data point. Note that some (or most) columns of random data in NX might also be in NY , but shifted to a different column. We will assume that the noise is independent of the true data, and is independent in both space and time, so that each element of a given noise matrix is a random variable taken from a fixed zero-mean normal distribution.

The effect of noisy data on $\mathcal{A}$ computed in (3.3)-(3.9) from the DMD formulation is shown to be deterministic and biased to sensor noise. The measured operator $\mathcal{A}$ can be computed as

$$\begin{aligned}\mathcal{A} = YX^{+} &= (Y_g + N_Y)\,(X_g + N_X)^{+}\,, \\ &= (Y_g + N_Y)\,(X_g + N_X)^{*}\,[(X_g + N_X)\,(X_g + N_X)^{*}]^{+}\,,\end{aligned}$$

where the identity given in (3.5) is used. For a sufficiently small noise, a correction to the bias that is present in the noise-free approximation $\hat{\mathcal{A}}$ can be computed from

noisy data (see [31] for derivation) as,

$$\hat{\mathcal{A}} \approx \mathcal{A}\left(I + m\sigma_N^2 \Sigma_m^{-2}\right), \tag{3.8}$$

where σ_N^2 is the variance of each independent component of the noise matrix, $U_m \Sigma_m V_m$ be the singular value decomposition of the noisy data X, where m is the number of snapshots. The size of the bias in DMD has been quantified and is related to the Signal-to-Noise ratio (SNR).

Lemma 1. *[31] The bias error in DMD becomes dominant when* $\dfrac{n}{m} < SNR^2$.

Intuitively, the above result implies that larger number of snapshots of very noisy data yields poor tracking performance.

3.2.4 Batch processing of snapshots versus online parameter updates

Standard parameter identification algorithms process all snapshots from a given trajectory data in one batch. The so-called "batch DMD" initializes and applies the linear regression step after m snapshots are gathered and repeats the process when a new operator needs to be computed. The Algorithm 1 below describes this in detail, but in brief, the agent is required to maintain a buffer of m snapshots at any given time k. In very high dimensional systems, $m \ll n$. In this work, we are interested in obtaining a matrix $\mathcal{A}$ that may be assumed time-varying owing to the potentially nonlinear target system, giving us a local linear model for the dynamics, but in the standard batch DMD approach, one seeks a single matrix $\mathcal{A}$ [2]. However, we can keep our focus on the *over-constrained* problem, in which $m > n$. We can realistically collect enough snapshots to achieve this. When the problem is *under-constrained*, the model will tend to overfit the data, and any noise present in the data will lead to poor performance of the model [45].

The computational cost of batch processing of snapshots where we compute the

operator $\mathcal{A}$ at the final time step when all target data is collected is

$$T_{standard} = \mathcal{O}\left(nm\min\left(m,n\right) + mn^2\right) = \mathcal{O}\left(mn^2\right).$$

This is derived from the fact that using (3.4) and (3.5), we compute $n \times m$ pseudo-inverse and an $n \times m$ and $m \times n$ matrix multiplication. This is not practical for real-time usage as we do not have prior access to the entire trajectory. On the other hand, we could initialize and apply the standard algorithm to the first few m_0 snapshots and maintain a window of m with every new snapshot and compute an update to the matrix at each time step while maintaining knowledge of all past data. The computational cost of that operation is

$$T_{batch} = \mathcal{O}\left(\sum_{k=m_0}^{m}\left(nk\min\left(k,n\right) + kn^2\right)\right) = \mathcal{O}\left(m^2n^2\right),$$

which is more realistic but can get expensive as m and n scale, for $m > n$.

There are ways to update $\mathcal{A}$ online with a new snapshot pair $\left(x_k, y_k\right)$, after an initialization step with m snapshots. This approach seems attractive to model a time-varying signal as it does not require storing the snapshots leading to a better space complexity. The initialization step is the same as assumed for computing $T_{standard}$ with additional at most $\alpha,\ 1 < \alpha < m$, multiples of $n \times n$ matrices [2]. The overall cost of this approach is

$$T_{online} = \mathcal{O}\left(mn^2 + \alpha n^2\right) = \mathcal{O}\left(mn^2\right).$$

These time complexity evaluations are explained in more detail in [2]. We will explore both methods of matrix updates for updating the linear operator in the following sections.

3.3 Frobenius Norm Minimization with Stability Constraints

In our subsequent analysis, we make the assumption that the dataset used to build matrices X and Y correspond to a stable target system. The Frobenius norm expression in (3.3) can be rewritten as

$$\begin{aligned}\mathcal{A} &= \text{argmin}_{\mathbf{A}} \left\{ \text{tr} \left[(Y - \mathbf{A}X)^\top (Y - \text{A}X) \right] \right\} \\ &= \text{argmin}_{\mathbf{A}} \left\{ \text{tr} \left(\mathbf{A}XX^\top \mathbf{A}^\top \right) - 2\text{tr} \left(XY^\top \mathbf{A} \right) + \text{tr} \left(Y^\top Y \right) \right\},\end{aligned}$$

which can be formulated as a quadratic program of the form

$$\text{argmin}_a \, a^\top Pa - 2q^\top a + r, \tag{3.9}$$

where $a \in \mathbb{R}^{4\times 1},\ q \in \mathbb{R}^{4\times 1},\ P \in \mathbb{R}^{4\times 4}$ and $r \in \mathbb{R}$ are respectively $\text{vec}(\mathbf{A})$, $\text{vec}(XY^\top)$, $I_2 \otimes (XX^\top)$ and $\text{tr}(Y^\top Y)$. I_2 is the 2×2 identity matrix and $\otimes$ denotes the Kronecker product with P being a symmetric positive-semidefinite matrix [1]. The objective function in (3.9) is a quadratic function of a.

This process results in a linear predictor of the form

$$z_{k+1} = \mathcal{A} z_k, \tag{3.10}$$

for $k > m$, where $z_k = \bar{y}_k,\ z_k \in \mathbb{R}^2$ for $k = 1, \ldots, m$.

3.3.1 Generating stability constraints

The quadratic objective function minimizes the output sequence reconstruction error but may not yield a stable operator for use with linear quadratic methods. Several techniques have been developed to enforce stability including [1,46], where the authors formulate the problem as a semidefinite program (SDP) whose objective minimizes the state sequence reconstruction error and whose constraint bounds the largest singular

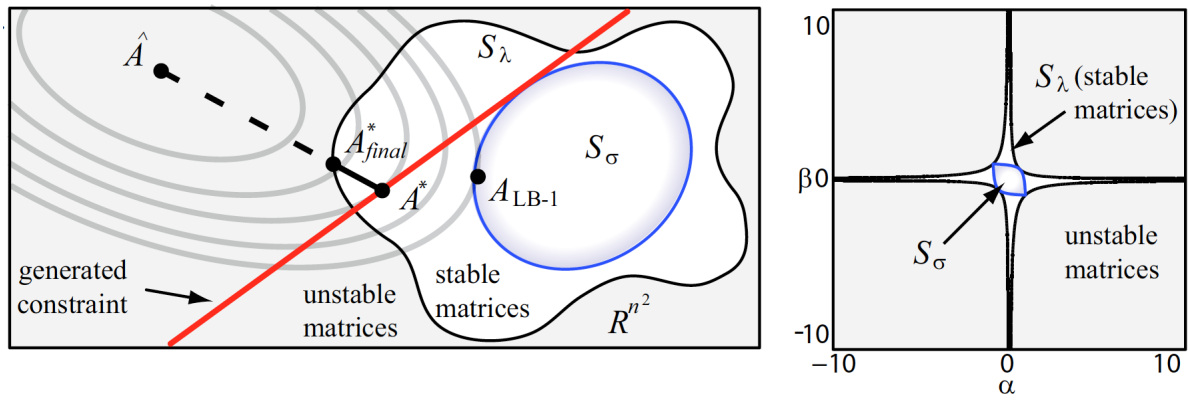


Figure 3.1: Conceptual depiction of the space of $n \times n$ matrices. The region of stability S_λ is non-convex while the smaller region of matrices with $\sigma_1 \leq 1$ S_σ is convex for $n > 1$. One iteration of constraint generation yields the constraint indicated by the line labeled 'generated constraint', and (in this case) leads to a stable solution A^* (Figure from [1]).

value by 1.

The following definitions and results have been extracted from [1] and [47] to learn a stable operator from the time-series data of the target output sequence.

Definition 2. A dynamical system with system matrix $M \in \mathbb{R}^n$ is *Schur stable* if all of M's eigenvalues λ_i, $i = 0, \ldots, n$, have magnitude of at most 1, i.e., spectral radius $\rho(M) < 1$.

Lemma 2. *[1] The set of all stable matrices S_λ is non-convex for all $n > 1$.*

Instead, we turn our focus to the singular values of the matrix M.

Definition 3. The singular values $\sigma_1, ..., \sigma_r$ of an $m \times n$ matrix M are the positive square roots, $\sigma_i = \sqrt{\lambda_i} > 0$, of the nonzero eigenvalues of the associated symmetric, positive semi-definite square Gram matrix $K = M^\top M$. The corresponding eigenvectors of K are the singular vectors of M.

Lemma 3. *[47] If $M = M^\top$ is a symmetric matrix, its singular values are the absolute*

values of its nonzero eigenvalues: $\sigma_i = |\lambda_i| > 0$; *its singular vectors coincide with the associated non-null eigenvectors.*

Proof. When M is symmetric, $K = M^\top M = M^2$. So, if $Mv = \lambda v$, then $Kv = M^2 v = \lambda^2 v$. Thus, every eigenvector v of M is also an eigenvector of K with eigenvalue λ^2 . Therefore, the eigenvector basis of M is also an eigenvector basis for K, and hence also forms a complete system of singular vectors for M. □

Lemma 4. *[48] The singular values of* M *are related to the eigenvalues of* M *as*

$$\sigma_{\min}(M) \leq \lambda_i(M) \leq \sigma_{\max}(M), \ \forall i = 1, \ldots, n \tag{3.11}$$

This is the end of the listing of results from [1]. This generates a convex constraint, $\sigma_1 = \sigma_{\max}(M) > 1$ for instability. Using singular value decomposition,

$$M = U\Sigma V^\top \Rightarrow \quad \Sigma = U^\top M V \quad \Rightarrow \quad \sigma_1(M) = u_1^\top M v_1 = \operatorname{tr}\left(u_1^\top M v_1\right), \tag{3.12}$$

the constraints of the form

$$g^\top \mu \leq 1, \tag{3.13}$$

may thus be generated [1], where $g = \operatorname{vec}(u_1 v_1^\top)$ and $\mu = \operatorname{vec}(M)$, where u_1 and v_1 are the corresponding vector forms of U and V associated with σ_1.

We can now iteratively minimize (3.9) subject to these additional new constraints in order to obtain a stable $\mathcal{A}$.

In Figure 3.2, the eigenvalues of the operator computed for a sample trajectory is presented. The unconstrained quadratic program yields unstable eigenvalues ($\triangleleft$). With each iteration of the quadratic program subject to the stability constraints, the process yields a stable operator ($\circ$) .

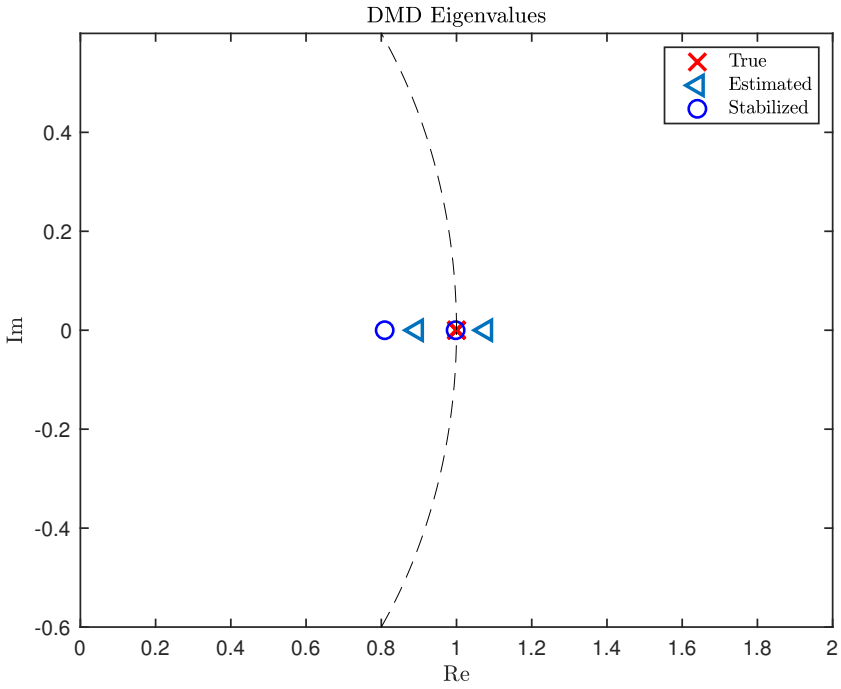


Figure 3.2: DMD Eigenvalues after applying the stability constraint.

3.4 Interpretation of the operator $\mathcal{A}$

Computing the linear operator as a solution to the linear least squares problem (3.3) produces a local linear model that our tracking agent can follow for some reasonably long time horizon, until the dynamics of the target forces an update to the linear operator. We can interpret the operator $\mathcal{A}$ as a low rank approximation of a higher dimensional system that may be slowly varying in time, so that the matrix $\mathcal{A}$, or in fact $\mathcal{A}_k$, can evolve as k increases. In this case, the entries of $\mathcal{A}_k$ may be updated as more data becomes available by processing the most recent m snapshots or in an online fashion [2]. Furthermore, if the system is assumed to be time varying, it may make sense to weight recent data snapshots more heavily as compared to the older snapshots as described in 3.2.2. We thereby consider minimizing a modified cost

function,

$$\mathcal{A}_k = \ \text{argmin}_{\mathbf{A}} \left\| \tilde{Y}_k - \mathbf{A}\tilde{X}_k \right\|_F^2,$$

with $\tilde{X}$ given as,

$$\tilde{X}_k = \begin{bmatrix} \rho^{k-1}\bar{y}_1 & \rho^{k-2}\bar{y}_2 & \cdots & \bar{y}_k \end{bmatrix},$$

and $\tilde{Y}$ is defined similarly as the time-shifted version of $\tilde{X}$, for some constant ρ, satisfying $0 < \rho \leq 1$. When $\rho = 1$, this cost function is the same as (3.3), and when $\rho < 1$, errors in past snapshots are discounted.

We assume that we can only measure position data directly from the onboard image sensors. This implies that $n_z = 2$ and $C_z = I_2$, and the above approach reduces to the setup we discussed previously in (3.10). We compute an approximate linear predictor that captures the flow of the target system given by (3.10) that behaves like a virtual leader. However, Koopman analysis assumes the knowledge of the full state which is not the case here. Since the target velocity is not measured directly, this framework will be based only on the inertial target coordinates returned by the visual sensor's image processing software. Through (3.3), we find the linear predictor that essentially enables us to minimize d_k for the current time step k further, when compared to a standard target following approach with partial state feedback. Note that $\hat{\mathcal{A}}$ behaves like a propagation matrix or a transfer matrix. If we assume that a portion of the target trajectory can be approximately modeled by (3.10), the UAV system (2.1) may be augmented directly with $\hat{\mathcal{A}}$. However, the data in (3.1) and (3.2) may be generated as the output of a higher dimensional system. The output equation may thus be rearranged in terms of $\bar{y}$ as

$$\bar{y}_{k+1} = C_z z_{k+1} = C_z A_z z_k = C_z A_z C_z^{+} \bar{y}_k. \tag{3.14}$$

Therefore, from (3.10) and (3.14), we can deduce that

$$\hat{\mathcal{A}} \approx C_z A_z C_z^{+}. \tag{3.15}$$

In order to have $y - \bar{y} \to 0$ with $u \to 0$, we can assume that the UAV and approximate target states, x and z are such that

$$A_z = \begin{bmatrix} \hat{\mathcal{A}} & 0 \\ 0 & A_{z_2} \end{bmatrix}.$$

3.5 Alternate methods for parameter identification

In order to compute the linear operator efficiently without storing m snapshots at any given time, we look at the streaming DMD setup that performs online updating of the DMD modes and eigenvalues [49]. Streaming DMD keeps track of a small number of orthogonal basis vectors and updates the DMD matrix projected onto the corresponding subspace. A similar approach [2] may be derived that updates the matrix directly from a new snapshot pair, without the need to store raw data.

The following result is obtained in [2] and is given here for completeness. Recall the the modified cost function and the scaled data matrices in (3.6) and (3.2.2). The unique least-squares solution that minimizes this cost function, given $\tilde{X}_k$ has full row rank, is given by

$$\mathcal{A}_k = \tilde{Y}_k \tilde{X}_k^{+} = \tilde{Y}_k \tilde{X}^{\top} \left(\tilde{X} \tilde{X}^{\top} \right)^{-1} = \tilde{Q}_k \tilde{P}_k,$$

where

$$\tilde{Q}_k = \tilde{Y}_k \tilde{X}_k^{\top},$$
$$\tilde{P}_k = \left(\tilde{X}_k \tilde{X}_k^{\top} \right)^{-1}.$$

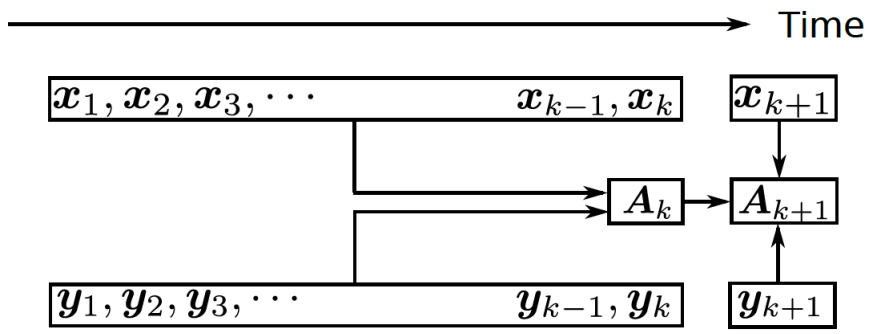


Figure 3.3: Schematic diagram of online DMD setup. Arrow indicates the information flow. (Figure from [2])

The update $\mathcal{A}_{k+1} = \tilde{Q}_{k+1}\tilde{P}_{k+1}$, can be derived as

$$\begin{aligned}\tilde{Q}_{k+1} = \tilde{Y}_{k+1}\tilde{X}_{k+1}^\top &= \begin{bmatrix} \sigma\tilde{Y}_k & y_{k+1} \end{bmatrix} \begin{bmatrix} \sigma\tilde{X}_k & x_{k+1} \end{bmatrix}^\top \\ &= \sigma^2\tilde{Y}_k\tilde{X}_k^\top + y_{k+1}x_{k+1}^\top \\ &= \rho\tilde{Q}_k + y_{k+1}x_{k+1}^\top,\end{aligned}$$

where (x_{k+1}, y_{k+1}) is the new snapshot pair of target positions at time $k+1$. Similarly,

$$\tilde{P}_{k+1} = \rho\tilde{P}_k^{-1} + x_{k+1}x_{k+1}^\top.$$

The updated DMD matrix is then given by

$$\mathcal{A}_{k+1} = \tilde{Q}_{k+1}\tilde{P}_{k+1} = \left(\rho\tilde{Q}_k + y_{k+1}x_{k+1}^\top\right)\left(\rho\tilde{P}_k^{-1} + x_{k+1}x_{k+1}^\top\right)^{-1}. \tag{3.16}$$

Lemma 5. *(Inversion lemma, Woodbury formula) Suppose M is an invertible square matrix, and u,v are column vectors. Then $M + uv^\top$ is invertible iff $1 + v^\top Mu \neq 0$,*

and the inverse is given by

$$\left(M + uv^\top\right)^{-1} = M^{-1} - \frac{M^{-1}uv^\top M^{-1}}{1 + v^\top M^{-1}u}. \tag{3.17}$$

The above inversion lemma is also known as the Sherman-Morrison formula [50,51]. Applying this to (3.16), we obtain

$$\tilde{P}_{k+1} = \frac{\tilde{P}_k}{\rho} - \gamma_{k+1}\frac{\tilde{P}_k}{\rho}x_{k+1}x_{k+1}^\top\frac{\tilde{P}_k}{\rho},$$

where

$$\gamma_{k+1} = \frac{1}{1 + x_{k+1}^\top\left(\tilde{P}_k/\rho\right)x_{k+1}}.$$

Let us define $\hat{P}_k$ as

$$\hat{P}_k = \frac{\tilde{P}_k}{\rho} = \frac{1}{\rho}\left(\tilde{X}_k\tilde{X}_k^\top\right)^{-1}.$$

The formula for $\mathcal{A}_{k+1}$ can now be given by

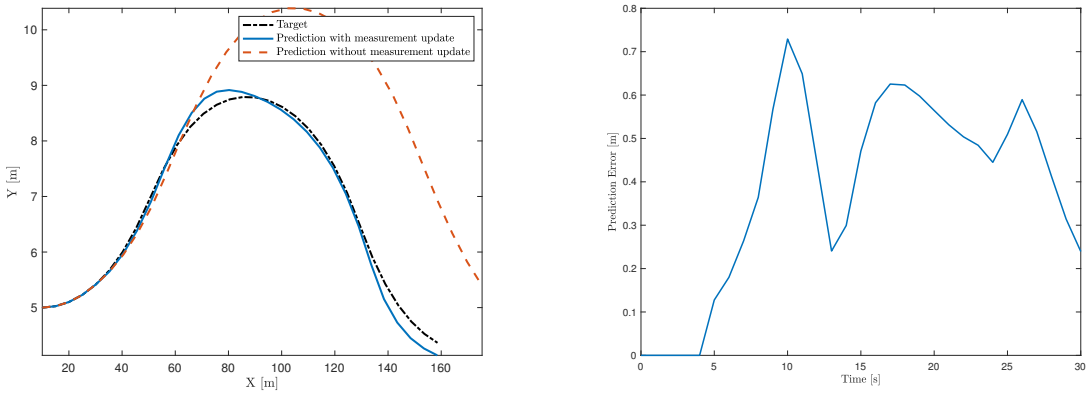
$$\mathcal{A}_{k+1} = \mathcal{A}_k + \gamma_{k+1}\left(y_{k+1} - \mathcal{A}_k x_{k+1}\right)x_{k+1}^\top\hat{P}_k, \tag{3.18}$$

with

$$\hat{P}_{k+1} = \frac{1}{\rho}\left(\hat{P}_k - \gamma_{k+1}\hat{P}_k x_{k+1}x_{k+1}^\top\hat{P}_k\right), \tag{3.19a}$$

$$\gamma_{k+1} = \frac{1}{1 + x_{k+1}^\top\hat{P}_k x_{k+1}}. \tag{3.19b}$$

When $\rho = 1$, the above formulas weight all snapshots equally. With initialization from $m = 8$ snapshots, the predicted trajectories for a sample target motion is shown in Figure 3.4(a). At every time step k, if we set the snapshot pair (x_{k+1}, y_{k+1}), as



(a) (b)

Figure 3.4: Predicted trajectories from the linear operator $\mathcal{A}$ for sample nonlinear trajectory. (a) The solid line represents the prediction performance of $\mathcal{A}_k$ applied to the measurement value $\bar{y}_k$, for $k > m$. The dashed line shows the prediction when the previous estimate is propagated through without regard for current measurement. (b) The estimation error for the former case.

$x_{k+1} = \bar{y}_{k-1}$ and $y_{k+1} = \bar{y}_k$, and obtain a new prediction for the target position as

$$\hat{y}_{k+1} = \mathcal{A}_k \bar{y}_k, \tag{3.20}$$

where the target position is updated from the latest measurement.

Define prediction residual error, $\mathcal{E}$ as

$$\mathcal{E} = \|\hat{y}_k - \bar{y}_k\|_2 ,$$

for $k > m$ which is presented in Figure 3.4(b). However, if we let the input of the next prediction step be equal to the prediction generated at the previous time step ($\hat{y}_{k+1} = \mathcal{A}_k \hat{y}_k$) without accessing the current measurement, larger errors can be seen, as expected.

3.6 Augmented LQR System

Following the derivation of an LQR system for the optimal servo problem [36], define a new variable

$$\hat{\mathbf{x}}_k = \begin{bmatrix} x_k - \tilde{x}_k \\ z_k \end{bmatrix} \tag{3.21}$$

and new system matrices

$$\hat{A} = \begin{bmatrix} A & F \\ 0 & \mathcal{A} \end{bmatrix} \quad \hat{B} = \begin{bmatrix} B \\ 0 \end{bmatrix}, \tag{3.22}$$

where $F = ALC_z + LC_z\mathcal{A} + \dfrac{d}{dt}(LC_z)$ and $L = C^\top(CC^\top)^{-1}$, with $\tilde{x_k} = L\bar{y}_k$. We know that the optimal control for this setup is given as

$$u^* = K(x - \tilde{x}) + \hat{K}z, \tag{3.23}$$

with

$$K = -R^{-1}B^\top P \text{ and } \hat{K} = -R^{-1}B^\top P_{12},$$

where P and P_{12} are solutions to the steady-state Riccati equations, where u^* minimizes the cost given by

$$V(\hat{\mathbf{x}}, u, t_0) = \int_{t_0}^{\infty} u^{*\top} R u^* + \hat{\mathbf{x}}^\top Q \hat{\mathbf{x}}\, dt,$$

with usual LQR weights Q and R, with Q positive semidefinite and R as positive definite.

The frequency of these updates depends on whether we apply batch processing or online updates to the linear operator. If we choose to process m snapshots at a time (batch), in order to keep $\mathcal{E} < \epsilon$, for some threshold ϵ, we would want to occasionally

Algorithm 1 Augmented LQR for UAV tracking control

1: Initialize discounted data matrices $\tilde{X}$ and $\tilde{Y}$ from m target position snapshots for some value of ρ.
2: Compute a stable noise-corrected linear operator $\mathcal{A}$ from the quadratic program (3.9) subject to constraints (3.13), such that $Y \approx \mathcal{A}X$.
3: Build augmented system (3.21) with $\mathcal{A}$ and compute control laws K and $\hat{K}$.
4: **for** time $k = m + 1$ **to** end
5: Compute optimal control u_k^* .
6: Obtain new UAV state x_{k+1} from u_k^* and the new target snapshot $\bar{y}_{k+1}$.
7: Update $\mathcal{A}_k$ with the new snapshot using the rank-1 update.
8: Compute the residual error $\mathcal{E}$ for the latest snapshot.
9: **if** $\mathcal{E} > \epsilon$, **then**
10: Update the feed-forward control law $\hat{K} = \hat{K}_k$ corresponding to system augmented with $\hat{\mathcal{A}}_k$.
11: **end if**
12: **end for**

update the linear operator $\mathcal{A}$ with more recent measurement data. This is outlined in Algorithm 1. Factors that affect $\mathcal{E}$ are the relative velocity of the tracking agent and the target or the nonlinearity of the target trajectory. For a target with fairly constant velocity, the quadratic program (3.9) eventually converges to a close representation of the target, making multiple updates unnecessary. However, if the target is relatively faster and more dynamic, the prediction error $\mathcal{E}$ grows large fairly quickly leading to frequent recomputations of the linear operator $\mathcal{A}$.

In order to avoid storing multiple snapshots to recompute the noise-corrected linear operator $\mathcal{A}$ when the residual $\mathcal{E}$ becomes large, we can adopt the online approach that updates $\mathcal{A}_k$ when a new snapshot becomes available. The rank-1 update strategy discussed above may be used to update the linear operator $\mathcal{A}_k$ at every time step k. This leads to a time-varying feed-forward control law, $\hat{K}_k$ that may not be ideal to implement. Instead, we will compute $\mathcal{A}_k$ at every time step, but choose to implement the new control law only when $\mathcal{E} > \epsilon \approx \beta c(\boldsymbol{h})$, for some suitable $\beta \in (0, 1)$.

Algorithm 2 Augmented LQR for UAV tracking control with online updates

1: Initialize discounted data matrices $\tilde{X}$ and $\tilde{Y}$ from m target position snapshots for some value of ρ.
2: Compute a stable noise-corrected linear operator $\mathcal{A}$ from the quadratic program (3.9) subject to constraints (3.13), such that $Y \approx \mathcal{A}X$.
3: Build augmented system (3.21) with $\mathcal{A}$ and compute control laws K and $\hat{K}$.
4: **for** time $k = m + 1$ **to** end
5: Compute optimal control u_k^* .
6: Obtain new UAV state x_{k+1} from u_k^* and the new target snapshot $\bar{y}_{k+1}$.
7: Update $\mathcal{A}_k$ with the new snapshot using the rank-1 update .
8: Update the feed-forward control law $\hat{K} = \hat{K}_k = -R^{-1}B^\top(A + BK)^{-1}Q\tilde{x}_{k+1}$, $\tilde{x}_{k+1} = L\mathcal{A}_k\bar{y}_k$.
9: **end for**

3.7 Numerical Simulation Results

We now demonstrate how building the augmented system from recorded data leads to improved tracking performance; this is done by demonstrating the performance of the algorithm on a few classes of target trajectories. First, we consider a target moving with a constant velocity and zero sensor noise and later consider trajectories generated by linear and nonlinear systems with varying levels of sensor noise. The parameters used in our simulation are listed in Table 3.1. The number of snapshots is typically varied based on the trajectory but defaults to the value given in Table 3.1. The typical execution time for solving the steps involved in the algorithm at each time step using MATLAB is between 10-80 milliseconds on a MacBook Pro running a 2.4 GHz Intel® Core™ i5 processor. We note that with T_s at $1s$, such computation is feasible for real time implementation.

3.7.1 Constant velocity target with and without noise

For a target trajectory with constant velocity $\mathbf{v}_t < \mathbf{v}_a$, the proposed approach either drives the tracking error d and the residual error $\mathcal{E}$ to zero or reduces the bias in the error significantly subject to initial conditions $\mathbf{x}_0$. This is because after enough infor-

Parameter	Description	Value	Units
T_s	Sampling Period	2	s
$\boldsymbol{h}$	UAV altitude	50	m
T	Total Simulation time	60	s
m	Number of snapshots	(10,15)	-
σ_t^2	Sensor Noise Variance	(1,4)	m^2
$\bar{\mathbf{v}}_a$	Max UAV speed	10	m/s
$\bar{\mathbf{v}}_t$	Max target speed	8	m/s

Table 3.1: Simulation Parameters

mation has become available, the estimated linear operator closely models the true target system properties resulting in better prediction, thus reducing or completely eliminating the bias present in the UAV output. As described in Section 3.2.3, the effect of sensor noise on DMD is often deterministic. Assuming zero mean normally distributed sensor noise with variance σ_t^2, we can in fact correct for this noise as described in (3.8). The simulation results for this case and the results for tracking noisy measurements is given in Figure 3.5. As these results demonstrate, the augmented LQR system tracks the true trajectory more closely than the regular LQR.

3.7.2 Target modeled by a linear system with and without noise

The quadratic program in (3.9) converges to a low rank estimate of the target system model that generates the flow recorded in the data matrices. For an uncontrolled system, with $n_z = 2$, our estimate provides the exact model that describes the target leading to $d \to 0$ and $u \to 0$. For a target system with some time-varying control input, $\hat{\mathcal{A}}$ captures the closed loop properties of the target system. This procedure is more effective on a trajectory that is smooth and is neither too slow nor fast. Faster and less smooth the target trajectories require more frequent updates for estimating the corresponding linear operator. Trajectories much slower than the UAV's minimum

velocity, that is the velocity required to stay in flight, will suffer from constant reroutes from having to loop around to keep d low. Sample trajectories and tracking errors for this case are shown in Figure 3.6.

3.7.3 Target modeled by a nonlinear system with and without noise

If the target evolution is described by a set of nonlinear equations, the linear operator computed from the proposed approach captures the dominant modes of the system that approximately describes the target behavior. When the full target state information is available, the DMD/Koopman operator approach "lift" the nonlinear dynamics in to a space where the evolution becomes linear acting on the space of observables [52]. The observable functions we consider (in our case, x_1 and x_2), however, give us access only to the partial target state. The linear operator computed from this data has missing information that causes the residual error to grow rapidly, requiring more frequent updates. Hence, this nonlinear trajectory is learned as a piece-wise linear approximation, with local learned models guiding the tracking process, which still out-performs a purely target following LQR approach. A trajectory generated by a nonlinear system with a time-varying input signal with and without sensor noise is presented in Figure 3.7.

3.7.4 Target with aggressive trajectories

The standard LQ method for tracking from only position measurements of the target performs quite satisfactorily for targets with slower dynamics. However, if the target executes more aggressive maneuvers, LQ quickly falls short in terms of its tracking performance. In this case, having a model of how the target evolves provides a way to compensate for its fast dynamics within the constraints of the following agent. Figure 3.8 demonstrates this quality, where the target accelerates quickly and slows down rapidly before returning to a more constant pace. The standard LQ falls behind as these maneuvers are made, whereas the setup where we have access to future

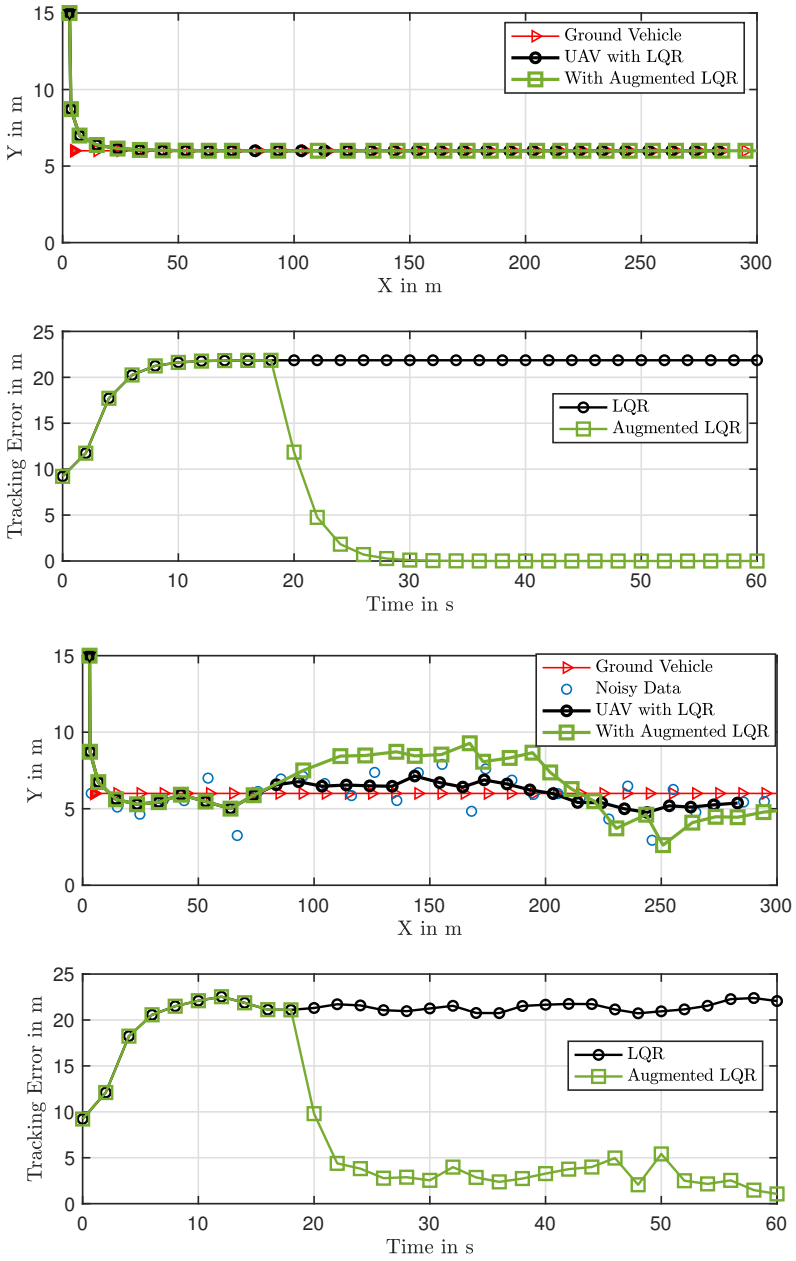


Figure 3.5: Trajectories and Tracking errors for the Constant Velocity case.

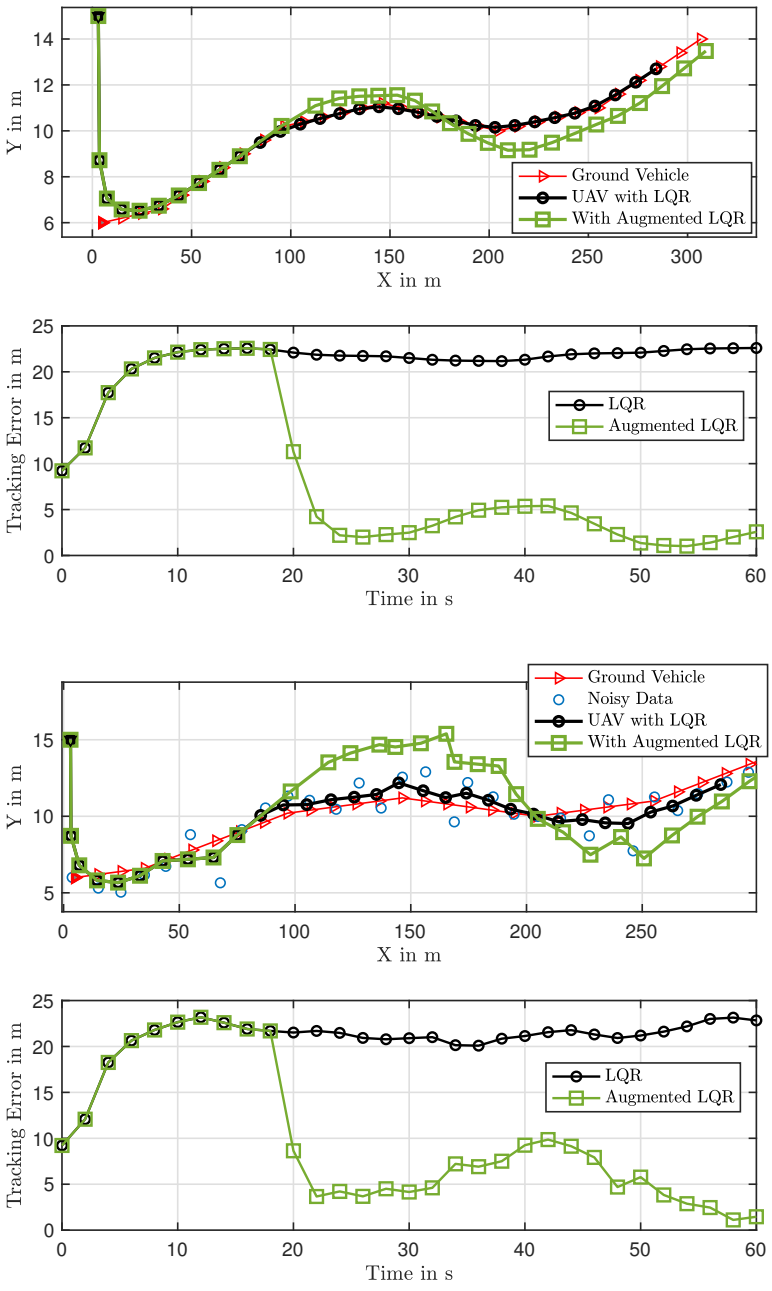


Figure 3.6: Trajectories and Tracking errors using the standard LQR and with the augmented system for a Linear Controlled case with max target acceleration of $0.5m/s$.

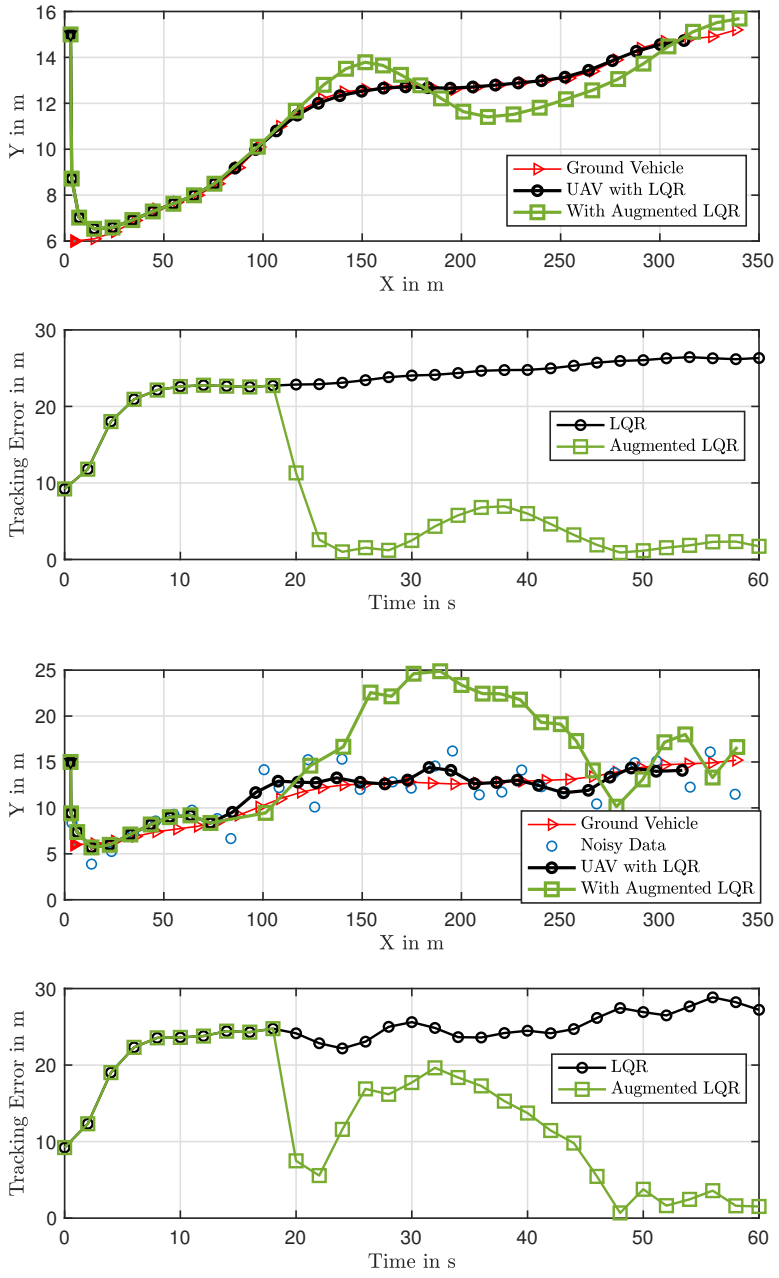


Figure 3.7: Trajectories and Tracking errors using the standard LQR and with the augmented system for a faster trajectory with maximum target acceleration of $2m/s$.

trajectory information from a known model follows the target more closely (blue line), only slightly faltering due to sudden changes in the input, before returning to a zero tracking error. Unfortunately, we usually do not have access to future trajectory information in the case of an intruder or a non-cooperating agent with unknown dynamics that we wish to track in the region of interest. In this case, building the artificial dynamical system as described in Algorithm 1 significantly improves the tracking performance and allows for a way to recover from a potentially catastrophic failure to obtain timely measurements due to sensor issues and occlusions.

3.8 Tracking Performance Analysis

In order to evaluate the relative performance improvement achieved with the data-guided augmented LQR approach, as compared with standalone LQR, let us consider the optimal cost incurred with the augmented system, given by

$$\begin{aligned} J_D^* &= \hat{\mathbf{x}}_0^\top \hat{P}(t_0)\hat{\mathbf{x}}_0 \\ &= \begin{bmatrix} \mathbf{x}_0 - \tilde{\mathbf{x}}_0 \\ z_0 \end{bmatrix}^\top \begin{bmatrix} P & P_{12} \\ P_{12}' & P_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 - \tilde{\mathbf{x}}_0 \\ z_0 \end{bmatrix}, \end{aligned} \tag{3.24}$$

where $\hat{P} \in \mathbb{R}^{6\times 6}$ is the solution to the Riccati equation for the augmented system and $P \in \mathbb{R}^4$ denotes the solution to the Riccati equation for the original system and $P_{12} \in \mathbb{R}^{4\times 2}$ as described earlier.

Define the matrix $\bar{P}$ as

$$\bar{P} = \begin{bmatrix} P & 0 \\ 0 & 0 \end{bmatrix},$$

such that $\bar{P} \in \mathbb{R}^{\dim(\hat{P})}$. Then, the minimum cost for the original system (LQR only) can be expressed as

$$J^* = \hat{\mathbf{x}}_0^\top \bar{P} \hat{\mathbf{x}}_0. \tag{3.25}$$

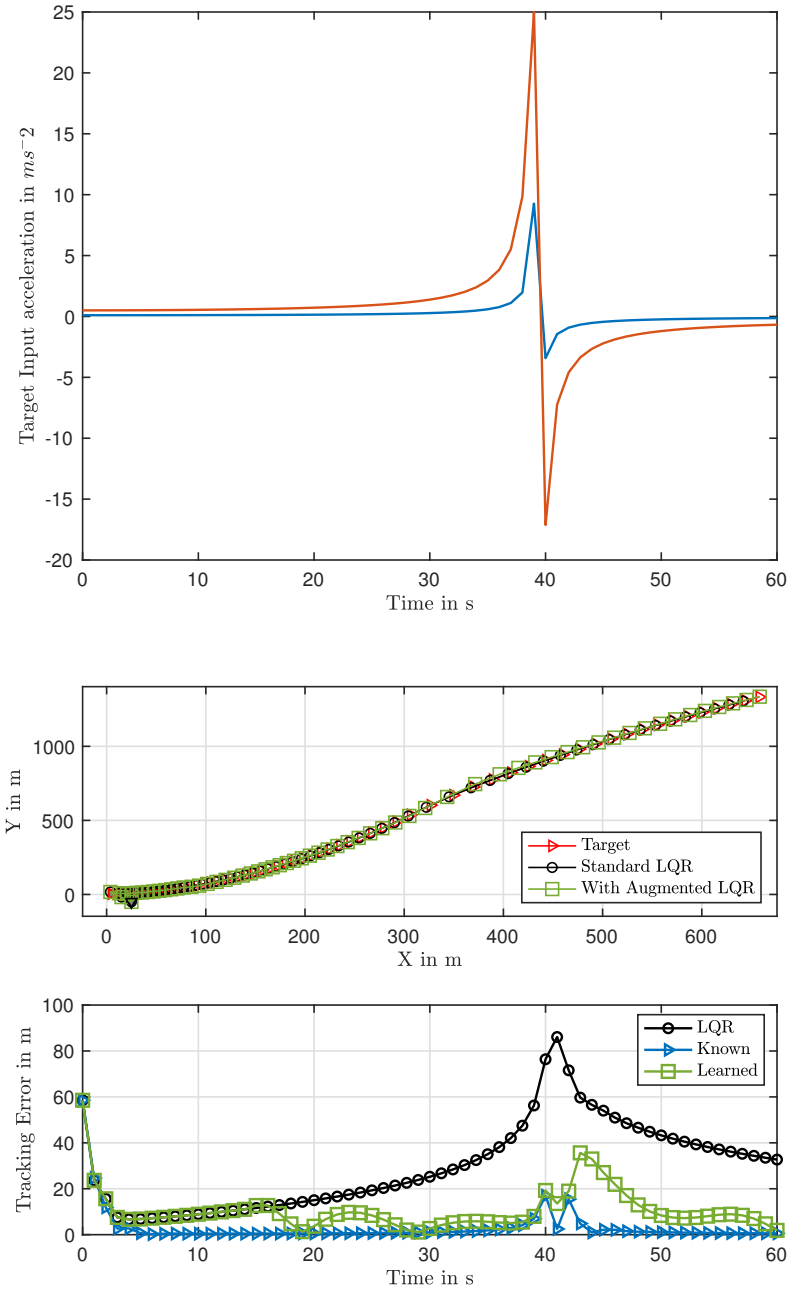


Figure 3.8: Response to fast trajectories

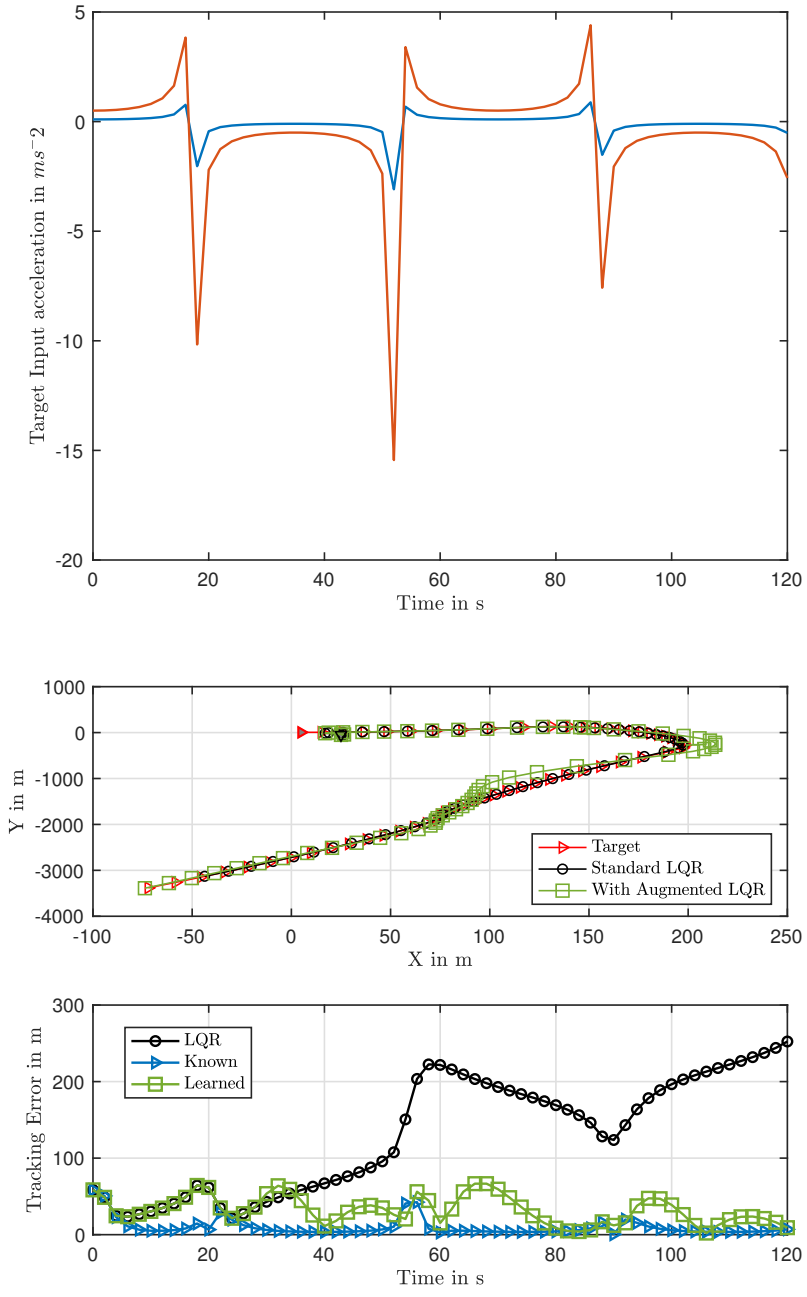


Figure 3.9: Additional sample trajectories

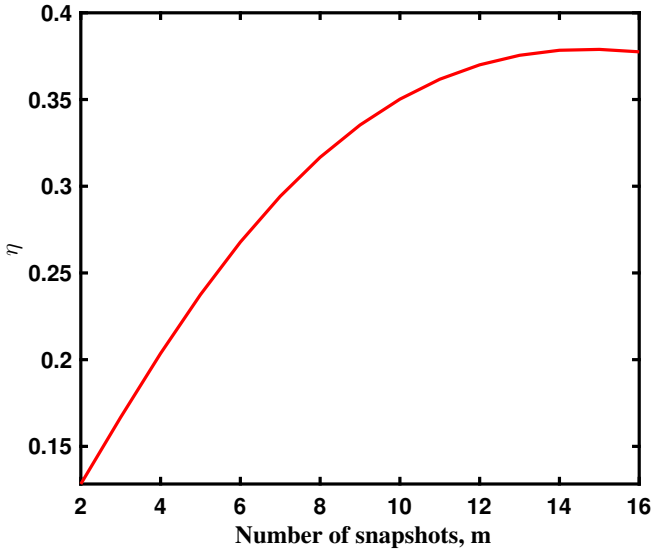


Figure 3.10: Relative Performance η for the linear target system trajectory shown in Figure 3.6, with different amounts of data considered.

Let us consider the quantity η [similar to the one described in [53]] given as

$$\eta = \frac{J^*(\mathbf{x}_0) - J_D^*(\mathbf{x}_0)}{J^*(\mathbf{x}_0)}. \tag{3.26}$$

Ideally, we want η to be non-negative for any given $\mathbf{x}_0$, i.e., in the worst case, the tracking performance with the augmented system is at least as good as with LQR alone. From (3.25) and (3.24), we see that η assumes

$$\begin{aligned} \eta &= \frac{\hat{\mathbf{x}}_0^\top \bar{P} \hat{\mathbf{x}}_0 - \hat{\mathbf{x}}_0^\top \hat{P} \hat{\mathbf{x}}_0}{\hat{\mathbf{x}}_0^\top \bar{P} \hat{\mathbf{x}}_0} \\ &= 1 - \frac{\hat{\mathbf{x}}_0^\top \hat{P} \hat{\mathbf{x}}_0}{\hat{\mathbf{x}}_0^\top \bar{P} \hat{\mathbf{x}}_0}. \end{aligned}$$

The best performance improvement that can be achieved with the model following

approach is when $\hat{P}$ is positive semidefinite. Therefore

$$\eta \geq \inf_{\mathbf{x}_0 \neq 0} 1 - \frac{\hat{\mathbf{x}}_0^\top \hat{P} \hat{\mathbf{x}}_0}{\hat{\mathbf{x}}_0^\top \bar{P} \hat{\mathbf{x}}_0}. \tag{3.27}$$

The above inequality (3.27) is in the form of a Rayleigh quotient and its infimum is given by $\lambda_{\min}(\bar{P}, \hat{P})$, where $\lambda_{\min}(M, N)$ is the smallest generalized eigenvalue of the pair (M, N), for M symmetric and N symmetric positive definite. As such, (3.27) can be rewritten as,

$$1 - \lambda_{\max}(\hat{P}, \bar{P}) \leq \eta \leq 1 - \lambda_{\min}(\hat{P}, \bar{P}). \tag{3.28}$$

We know that matrix $\hat{P}$, and in particular the sub-matrices P_{12} and P_{22}, are functions of the approximated linear operator generated from the measurement data and therefore a function of the data itself. in fact, the accuracy or correctness of the linear operator $\mathcal{A}$ depends on m, the number of snapshots used to build the data matrices X and Y. The choice of the optimal $\hat{P}$ depends on the LQR weights, the class of the trajectory (linear vs nonlinear or fast vs slow) and the number of available snapshots. We can carefully choose $\hat{P}$ based on these different factors in order to make the bound (3.28) less conservative.

Beyond a certain number of snapshots m, further improvement in the parameter η becomes insignificant. In fact, the performance improvement deteriorates if m is very large. Intuitively, larger the value of m, less time spent tracking with the augmented system; in addition, the effect of noise becomes more pronounced for large datasets. If the output data obtained from the target observation is highly nonlinear or random, the number of updates to the linear operator $\mathcal{A}$ increases and η remains low. However, η remains non-negative, which shows that the performance of the data-guided tracking is at least as good as using LQR feedback alone. A general guideline for choosing m, is to have $m > \text{rank}(X_k)$, in order to avoid dealing with the ill-conditioning issue and

under-fitting.

Furthermore, in order to obtain a lower bound that maximizes the performance improvement, we attempt to minimize the quantity,

$$\min \ \lambda_{\max}(\hat{P}, \bar{P}) - 1$$

$$\text{subject to } \hat{P} \succeq 0$$

3.9 Comparison with model-free reinforcement learning

In order to provide a constructive comparative study, we attempt to learn policy parameters of a given feedback policy structure $u = \pi(\hat{\mathbf{x}})$ that performs well over a given class of reference trajectories for a system given in (3.21) and (3.22). The total cost may be defined as earlier,

$$V(x_0) = \Sigma_{k=0}^{\infty}(x_k - \tilde{x}_k)^T Q(x_k - \tilde{x}_k) + u_k^T R u_k,$$

that penalizes the state deviations from the reference target point $\tilde{x}_k$. This value is represented by the Q-function which is a state-action combination defined for a single agent system as

$$Q_U(x_k, u_k) = R(x_k, u_k) + \gamma Q_U(x_{k+1}, u_{k+1}),$$

where $R(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k$, is the one-step cost incurred when moving from state x_k to x_{k+1}, $Q_U(x_k, u_k) = x_k^T P x_k$, and P is the cost-to-go matrix obtained as a solution to the algebraic Riccati equation. Following through the derivation of the

Q-function [27] with the augmented system described above, we obtain,

$$\begin{aligned} u_k &= -(\hat{B}^\top \hat{P} \hat{B} + R)^{-1} \hat{B} \hat{P} \hat{A} \hat{x}_k \\ &= -(B^\top P B + B^\top P_{12} B + R)^{-1} \dots \\ &\quad [\ B^\top P A \quad B^\top P F + B^\top P_{12} A_z\ ] \begin{bmatrix} x_k - \tilde{x}_k \\ z_k \end{bmatrix}. \end{aligned}$$

Q-learning also allows for a direct estimation of the Q-function without the need to identify the system matrices A, B, the cost matrices P and P_{12} or the one step cost separately. Assuming a quadratic basis from the elements of the state vector x_k, we can compute a linear relationship between the reward and the Q-function parameters. Solving this, yields a kernel matrix, $\hat{H}_k$ that corresponds to a parameter matrix,

$$\hat{H}_k = \begin{bmatrix} \hat{A}^\top \hat{P} \hat{A} + \hat{Q} & \hat{B}^\top \hat{P} \hat{A} \\ \hat{A}^\top \hat{P} \hat{B} & \hat{B}^\top \hat{P} \hat{B} + R \end{bmatrix}.$$

The policy improvement step is thereby given as,

$$u_k = -\hat{H}_{k_{22}}^{-1} \hat{H}_{k_{21}} \hat{\mathbf{x}}_k.$$

One major drawback of using Q-learning directly, especially for real-time tracking is that it takes many iterations to obtain a good policy that performs well. Since this algorithm involves an exploration phase, unless we choose appropriate excitation signals and samples, a large portion of the relevant areas of the state-space is unexplored and convergence to zero error takes longer to achieve.

3.10 Conclusions

The main contribution of this work is a data-guided target tracking scheme that aims to achieve performance improvement by augmenting the standard regulator in

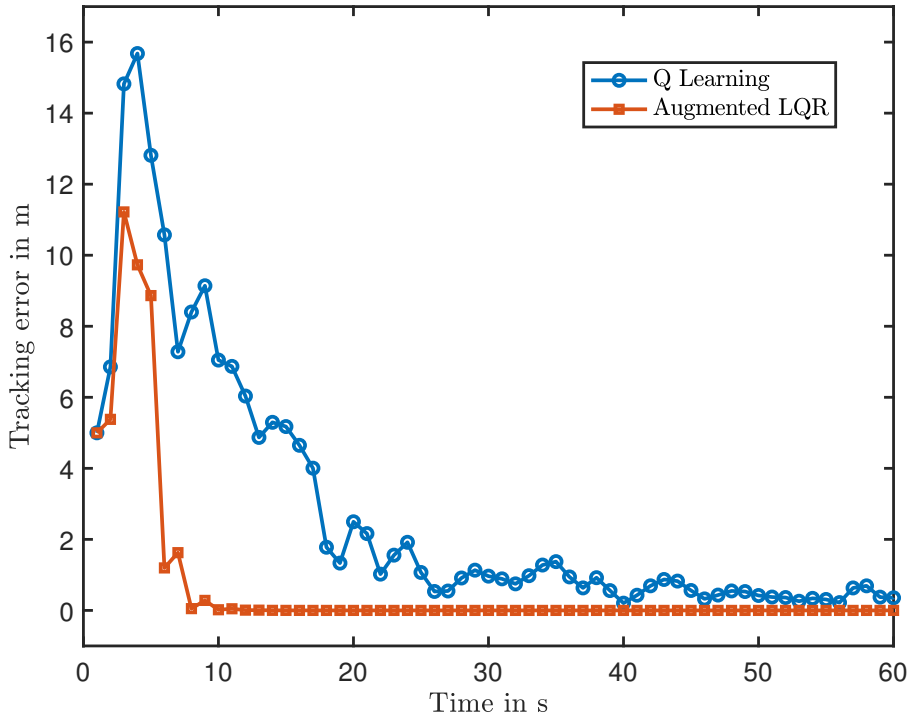


Figure 3.11: Tracking error convergence for a simple constant velocity case. Q-learning, although eventually performs satisfactorily, takes many iterations and much longer to converge to zero tracking error.

order to utilize past and current observations of the moving target. Inspired by the trajectory predicting capabilities of the Koopman operator, we used linear least squares and a quadratic program to build an artificial dynamical system from the target position data to create an augmented state system that tracks the target more closely than it does with position feedback alone. The simulation results showed this improved performance for certain classes of target systems. We then formally defined the quantity η that represents the relative performance improvement achieved using the data-guided approach. Subsequently, we derived bounds for this parameter and examined conditions under which additional data would not further improve target tracking. We then compared this "model-based" approach with one that learns a policy

directly from the tracking errors observed from a model-free Q-learning formulation. In our future work, we will extend these results to a nonlinear set up for the UAV and subsequently to a team of UAVs that cooperatively track a given target by running a consensus algorithm.

Chapter 4

TARGET TRACKING VIA MODEL-FREE REINFORCEMENT LEARNING

4.1 Real-time Q-learning for continuous state and action spaces

The idea behind using Q-learning for tracking has been adapted from Steven Bradtke's work on Incremental Dynamic Programming [27]. He explains - "Traditional Dynamic Programming (DP) methods are model-based methods that require the state transition function of the system in order to determine the optimal control law. Algorithms such as Q-learning directly learn a policy and do not require such a model. In a model-based approach, a system model is first developed from a training dataset and an optimal control problem is solved using the learned model. This relatively simplifies the problem than having to learn the control law directly. Another reason to take the model-based approach is that, once the model is formed, it may be used to solve a number of related control problems. There are scenarios where a model-free approach is more appropriate; it may be less expensive to find the optimal or at least an acceptable controller through direct interaction with the system than by building a model and deriving a controller [27]. A model of the state transition function may be easy to obtain for simple systems. But, for other more complex systems like financial markets, home heating or flight control, an accurate model is very difficult to obtain and are often idealized and inaccurate. Even if the exact model is known, computing an optimal controller may be computationally intractable. Q-learning does not use a system model to form its estimates of the optimal value function or the optimal policy. It is an online algorithm that tries to balance two competing goals: exploration and exploitation. Efficient exploration requires that actions should always be

chosen so that the average useful information gain per time step is maximized. However, since this is an online algorithm, it is also desirable to exploit what is currently known about the problem in order to find a solution. Most reinforcement learning (RL) results are limited to systems with both finite state and finite action sets and fail when applied to continuous systems and may become unstable or may converge to controllers that are destabilizing."

The DP-based reinforcement learning algorithm described in [27] provides convergence proofs for problems with continuous state and action spaces and is included here for completeness. The Adaptive Policy Iteration algorithm is firmly tied to the LQR theory. We will briefly describe this connection in the following sections and develop an approach to LQ-based tracking without using system models for the UAV or the target. Much of the concepts and some text is taken directly from [27] and applied to the augmented system described in Chapter 3.

4.2 Linear Quadratic Regulation

Recalling the LQR formulation, consider the linear, discrete-time, multivariable, dynamic system,

$$x_{k+1} = f(x_k, u_k) = Ax_k + Bu_k \tag{4.1}$$

with feedback control

$$u_k = Kx_k. \tag{4.2}$$

Here A, B are the system matrices as described in Section 2.1 and K is chosen so that spectral radius, $\rho(A + BK) < 1$, *i.e.*, $A + BK$ has all its eigenvalues strictly within the unit circle.

The one step cost is given as

$$r_k = \mathfrak{R}(x_k, u_k) = x_k^\top Q x_k + u_k^\top R u_k, \tag{4.3}$$

with $Q \in \mathbb{R}^{n\times n}$ symmetric positive semidefinite and $R \in \mathbb{R}^{m\times m}$ symmetric positive definite. The *total cost* of a state x_k under control policy K, $V_K(x_k)$, is defined as the discounted sum of all one step costs that will be incurred by using K from time k onward, *i.e.*, $V_K(x_k) = \sum_{i=0}^{\infty} \gamma^i r_{k+i}$, where $0 \leq \gamma \leq 1$ is the discount factor. This gives

$$V_K(x_k) = \mathfrak{R}(x_k, Kx_k) + \gamma V_K(x_{k+1}). \tag{4.4}$$

V_K is a quadratic function [54] and therefore can be expressed as

$$V_K(x_k) = x_k^\top P_K x_k, \tag{4.5}$$

where P_K is the $n \times n$ cost matrix for policy K. K^* denotes the policy that is optimal in the sense that the total discounted cost of every state is minimized. $P^* = P_{K^*}$ represents the cost matrix associated with K^*. If the matrices A, B, Q, R and K are known, then P_K can be found as the unique positive definite solution of the linear equation

$$P_K = \gamma(A + BK)^\top P_K(A + BK) + Q + K^\top RK. \tag{4.6}$$

P^* can be found as the unique positive definite solution to the Ricatti equation

$$P^* = A^\top \left[\gamma P^* - \gamma^2 P^* B \left(Q + \gamma B^\top P^* B\right)^{-1} B^\top P^*\right] A + Q. \tag{4.7}$$

The optimal policy is

$$K^* = -\gamma \left(R + \gamma B^\top P^* B\right)^{-1} B^\top P^* A. \tag{4.8}$$

This is simple but computationally expensive to derive even when the models of the system and cost function are available. DP-based reinforcement algorithms address how to define an adaptive policy that converges to K^* without access to such models.

4.3 Direct Estimation of Q-functions for tracking

The Q-function is a quantity that relates the quality of a state-action combination. The following derivation is outlined in [27] and is included here to show its relevance in the augmented system tracking that follows. The Q-function for an LQR problem can be given as

$$\begin{aligned}\mathcal{Q}_K(x,u) &= \mathfrak{R}(x,u) + Q_K(f(x,u), K(f(x,u))) \\ &= \mathfrak{R}(x,u) + \gamma V_K(f(x,u)) \qquad (4.9)\\ &= x^\top Q x + u^\top R u + \gamma(Ax+Bu)^\top P_K(Ax+Bu) \\ &= \begin{bmatrix} x, & u \end{bmatrix}^\top H_K \begin{bmatrix} x, & u \end{bmatrix}, \qquad (4.10)\end{aligned}$$

where

$$H_K = \begin{bmatrix} Q+\gamma A^\top P_K A & \gamma A^\top P_K B \\ \gamma B^\top P_K A & R+\gamma B^\top P_K B \end{bmatrix},$$

with $\begin{bmatrix} x, & u \end{bmatrix}^\top$ is the column vector of concatenation of x and u and H_K is a symmetric positive definite matrix of dimensions $(n+m)\times(n+m)$.

The Q-function defined in (4.10) can be directly estimated using recursive least squares without the need to identify the system model or the one-step cost function separately. Let us follow of the derivation of $\mathcal{Q}_K$ for the tracking problem, in which a UAV is flying at a fixed altitude $\boldsymbol{h}$, attempting to track a target with altitude $\boldsymbol{h}_t < \boldsymbol{h} - \delta$, where δ ensures that target is at a reasonable distance below the tracking agent. First, let us define a new state that represents the deviation of the UAV state from the desired trajectory as

$$\hat{x}_k = x_k - \tilde{x}_k,$$

with $\tilde{x}_k = L\tilde{y}_k$ is the observed state of the target at time step k. $\hat{x}_k$ represents the tracking error in 2D between the positions of the tracking agent (UAV) and the target.

At each time step k, the goal is to have the UAV be able to measure $\hat{x}_k$, and choose the appropriate control to regulate $\hat{x}$ to zero, without the need to know the system model A and B or the target dynamics. Define a vector $\bar{x}$ whose elements are the quadratic basis functions over the elements of x, *i.e.,*

$$\bar{x} = \left[x_1^2, \ldots, x_1 x_n, x_2^2, \ldots, x_2 x_n, \ldots, x_{n-1}^2, x_{n-1} x_n, x_n^2\right]^\top.$$

Let Θ be a function of square matrices, where $\Theta(P)$ is the vector whose elements are the n diagonal entries of P and the $n(n+1)/2 - n$ distinct sums $(P_{ij} + P_{ji})$. The elements of $\bar{x}$ and $\Theta(P)$ are ordered so that $x^\top P x = \bar{x}^\top \Theta(P)$. The Q-function can be written as

$$\mathcal{Q}_K(\hat{x}, u) = [\hat{x}, u]^\top H_K [\hat{x}, u] = \overline{[\hat{x}, u]}^\top \Theta(H_K). \tag{4.11}$$

We can rearrange (4.10) as

$$\begin{aligned}
r_k &= \mathfrak{R}(\hat{x}_k, u_k) \\
&= Q_K(\hat{x}_k, u_k) - \gamma Q_K(\hat{x}_{k+1}, K\hat{x}_{k+1}) \\
&= [\hat{x}_k, u_k]^\top H_K [\hat{x}_k, u_k] - \gamma [\hat{x}_{k+1}, K\hat{x}_{k+1}]^\top H_K [\hat{x}_{k+1}, K\hat{x}_{k+1}] \\
&= \overline{[\hat{x}_k, u_k]}^\top \Theta(H_K) - \gamma \overline{[\hat{x}_{k+1}, K\hat{x}_{k+1}]}^\top \Theta(H_K) \\
&= \phi_k^\top \theta_K
\end{aligned}$$

with

$$\phi_k = \overline{[\hat{x}_k, u_k]}^\top - \gamma \overline{[\hat{x}_{k+1}, K\hat{x}_{k+1}]}^\top,$$

and $\theta_K = \Theta(H_K)$. The recurrence relations for the recursive least squares are derived

Algorithm 3 Q-function based policy iteration algorithm for tracking

Initialize parameters $\hat{\theta}_1(0)$ and set $k = 0$, $j = 1$ and K_1 stabilizing.
Observe current relative state $\hat{x}_k = x_k - \tilde{x}_k$.
repeat forever
  Initialize $S_j(0) = S_0$.
  for $i = 1$ **to** N
    Set $e_k = \text{rand}(m)/\sqrt{j}$.
    $u_k = K_j \hat{x}_k + e_k$.
    Obtain new relative state $\hat{x}_{k+1}$ from applying u_k to the UAV system.
    Update Q-function parameters, $\hat{\theta}_j(i)$ from equations in (4.12).
    $k = k + 1$.
  end
  Compute matrix $\hat{H}_j$ from (4.11) that corresponds to parameter vector $\hat{\theta}_j$.
  Perform policy improvement as $K_{j+1} = -\hat{H}^{-1}_{j(22)} \hat{H}_{j(21)}$.
  Initialize parameters $\hat{\theta}_{j+1}(0) = \hat{\theta}_j$.
  $j = j + 1$.
end repeat

in [27] and the ith estimate of θ^*_j is given by

$$\begin{aligned}
\hat{\theta}_j(i) &= \hat{\theta}_j(i-1) + \frac{S_j(i-1)\phi_k(r_k - \phi_k^\top \hat{\theta}_j(i-1))}{1 + \phi_k^\top S_j(i-1)\phi_k} \\
S_j(i) &= S_j(i-1) - \frac{S_j(i-1)\phi_k\phi_k^\top S_j(i-1)}{1 + \phi_k^\top S_j(i-1)\phi_k} \\
S_j(0) &= S_0. \qquad (4.12)
\end{aligned}$$

$S_0 = \beta I$ for some large positive constant β.

The persistence of excitation condition [55] for ϕ_k such that the above algorithm converges asymptotically to the true parameters with $\hat{\theta}_j$ fixed is given by

$$\epsilon_0 I \leq \frac{1}{N} \sum_{i=1}^{N} \phi_{j-i}\phi_{k-i}^\top \leq \bar{\epsilon}_0 I, \quad \forall j \geq N_0 \text{ and } N \geq N_0, \qquad (4.13)$$

with N_0 and $\epsilon_0 \leq \bar{\epsilon}_0$ positive.

4.4 Adaptive policy iteration

In Q-learning, an agent learns a policy or a feedback law K, that minimizes or maximizes the value of the sum of recursive reward functions from observing the results of its own actions. Each policy iteration step consists of two phases: estimation of the Q-function for the current controller and policy improvement based on that estimate. Each estimation interval is N time steps long. For the jth policy iteration step, the estimate of the true parameter vector, $\theta_j^* = \Theta\left(H_{K_j}\right)$, is given by $\hat{\theta}_j = \hat{\theta}\left(N\right)$. The recursive least squares algorithm sets $P_j\left(0\right) = P_0$ and $\hat{\theta}_j\left(0\right) = \hat{\theta}_{j-1}\left(N\right)$. After identifying the parameters $\Theta\left(H_{K_j}\right)$ for N time steps, one policy improvement step is taken based on the estimate $\hat{\theta}_j$ which produces a new policy K_{j+1}. The outline of this algorithm is given in 3. The policy improvement process based on Q-functions and the ability to estimate H_K directly are the two key elements of the adaptive policy iteration algorithm.

Each of the K_j is not guaranteed to be stabilizing, since the kth policy improvement step is based on the estimate $\Theta\left(H_{K_j}\right)$ and the sequence K_j that may lead to K^* are not immediately apparent. The convergence of the adaptive policy iteration algorithm was established in [27] and is given below for completeness.

Theorem 1. ***(Convergence of adaptive policy iteration)*** *Suppose that $\{A, B\}$ is a controllable pair[1], that K_0 is a stabilizing control and that the vector ϕ_k is persistently excited. Then there exists an estimation interval $N < \infty$ so that the adaptive policy iteration generates a sequence $\{K_j,\ j = 1, 2, \ldots\}$ of stabilizing controls, converging so that*

$$\lim_{k\to\infty} \|K_j - K^*\| = 0,$$

where K^* is the optimal feedback control matrix.

Lemma 6. *If $\{A, B\}$ is controllable, K_1 stabilizing, and $K_2 = -\gamma\left(R + \gamma B^\top P_1 B\right)^{-1} B^\top P_1 A$,*

[1] *The pair $\{A, B\}$ is controllable if the $n \times nm$ matrix $\left[B, AB, A^2B, \ldots, A^{n-1}B\right]$ has linearly independent rows.*

then

$$P_1 - P_2 = \sum_{i=0}^{\infty} \gamma^i A_2^{i\top} \left[(K_1 - K_2)^\top \left(R + \gamma B^\top P_1 B \right) (K_1 - K_2) \right] A_2^i,$$

where $A_1 = A + BK_1$ *and* $A_2 = A + BK_2$.

Lemma 7. *If* $\{A, B\}$ *is controllable,* K_1 *stabilizing with associated cost matrix* P_1 *and* K_2 *is the result of one policy improvement step from* K_1*, i.e.* $K_2 = -\gamma \left(R + \gamma B^\top P_1 B \right)^{-1} B^\top P_1 A$*, then*

$$\Delta \|K_1 - K_2\|^2 \leq tr(P_1) - tr(P_2) \leq \delta \|K_1 - K_2\|^2,$$

where

$$0 < \Delta = \underline{\sigma}(R) \leq \delta = tr\left(R + \gamma B^\top P_1 B \right) \| \sum_{i=0}^{\infty} \gamma^{(i/2)} \left(A + BK_2 \right)^i \|^2,$$

and $\underline{\sigma}(\cdot)$ *denotes the minimum singular value of a matrix.*

This ends the listing of relevant results from [27].

4.5 Policy improvement for the augmented tracking system

We consider a single aerial vehicle tracking a moving target. The state of the UAV is augmented with the target reference state $\tilde{x}_k$ at the current time step k. The associated state space equation is

$$\hat{x}_{k+1} = \begin{bmatrix} A & 0 \\ 0 & \mathcal{A} \end{bmatrix} \hat{x}_k + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k, \tag{4.14}$$

where $\hat{x}_k = [x_k, \tilde{x}_k]^\top$, with the output equation given by

$$x_k - \tilde{x}_k = \begin{bmatrix} I_4 & -I_4 \end{bmatrix} \hat{x}_k. \tag{4.15}$$

Here, the output of the augmented system is the observed relative distance between the UAV and the target. Let the system matrices be denoted by $\hat{A}$, $\hat{B}$ and $\hat{C}$. We start with an initial arbitrarily selected stabilizing policy K_1 for the above system. This policy may be obtained with some trial and error using a coarse system model of the UAV. We used a random exploration signal generated from a normal distribution in order to induce persistent excitation of the vector ϕ_k. Let us define e_k as the "exploration" component of the control signal and initialize it to some small random value, which is the same dimension as the control. We will let e_k decrease in magnitude as more time is spent exploring to reduce its effects on tracking error gradually.

Let us define N as the horizon length, which is the number of time steps spent in the exploration phase. N is usually chosen to be twice the number of parameters of this system. The $\hat{A}$ and $\hat{B}$ matrices of (4.14) can contain a total of up to $2n^2 + nm$ independent parameters that must be estimated. On the other hand, the function Q_K for some policy K contains $(2n + m)(n + m + 1)/2$ independent parameters. The adaptive policy iteration algorithm [27] estimates fewer parameters than the model-based method whenever

$$\frac{(2n + m)(n + m + 1)}{2} < 2n^2 + nm.$$

It can be seen from the above inequality that the adaptive policy iteration algorithm requires estimating fewer parameters whenever $m < 2n - 1$ when compared with the model-based method that estimates the parameters of $\hat{A}$ and $\hat{B}$. The effect of the choice of N on the convergence of the adaptive policy iteration to the optimal policy is shown in Figure 4.1.

4.6 Convergence of policy iteration for the augmented system

Output controllability describes the ability of an external input to move the output of the system from any initial condition to any final condition in a finite time interval.

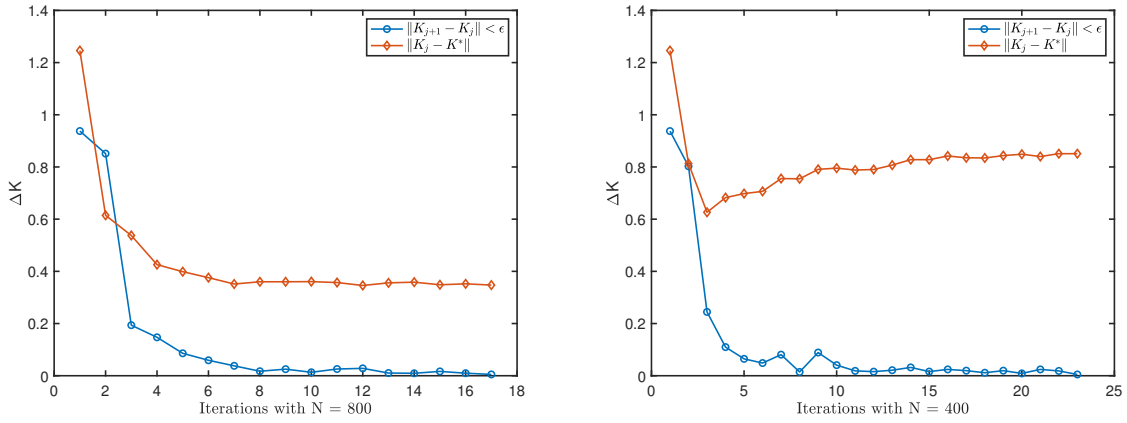


Figure 4.1: Effect of exploration horizon N on the algorithm convergence. Policy convergence for $N = 800$ and failure of policy improvement for $N = 400$. Terminating condition set to $\Delta K < \epsilon$, with $\epsilon = 0.005$.

Lemma 8. *The augmented system given in* (4.14) *is output controllable.*

Proof. For the linear continuous-time system (4.14), the $m \times (n+1)\, r$ output controllability matrix

$$\begin{bmatrix} \hat{C}\hat{B} & \hat{C}\hat{A}B & \hat{C}\hat{A}^2\hat{B} & \cdots & \hat{C}\hat{A}^{n-1}\hat{B} & D \end{bmatrix}, \tag{4.16}$$

has full row rank (i.e. rank m) if and only if the system is output controllable. It can be seen that the above output controllability matrix of the augmented system reduces to the state controllability matrix of the UAV system,

$$\begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix},$$

which has full row rank, $m = n$. □

Lemma 9. *The stabilizing controller produced by the adaptive policy iteration algorithm for the augmented system learns the feedback and feedforward components of the stabilizing controller.*

Proof. Following through the derivation of the Q-function, Q_K for the augmented system described above, the policy improvement step yields a new stabilizing policy, K_{j+1}, from which a new control law can be computed. The control law at time k for the augmented system in (4.14) is

$$
\begin{aligned}
u_k &= \underbrace{-(R+\hat{B}^\top \hat{P}\hat{B})^{-1}\hat{B}\hat{P}\hat{A}}_{K_{j+1}}\hat{x}_k \\
&= -\left(R + \begin{bmatrix} B^\top & 0 \end{bmatrix} \begin{bmatrix} P & P_{12} \\ P_{12}^\top & P_{22} \end{bmatrix} \begin{bmatrix} B \\ 0 \end{bmatrix}\right)^{-1} \cdots \\
&\qquad \begin{bmatrix} B^\top & 0 \end{bmatrix} \begin{bmatrix} P & P_{12} \\ P_{12}^\top & P_{22} \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & \mathcal{A} \end{bmatrix} \begin{bmatrix} x_k \\ \tilde{x}_k \end{bmatrix} \\
&= -\begin{bmatrix} \left(R+B^\top P B\right)^{-1} B^\top A & \left(R+B^\top P B\right)^{-1} B^\top P_{12}\mathcal{A} \end{bmatrix} \begin{bmatrix} x_k \\ \tilde{x}_k \end{bmatrix}
\end{aligned}
$$

From the lemma given in (3.17), the above expression separates out the feedback and feedforward terms and can be represented as

$$u_k = \begin{bmatrix} K_{j+1}^{fb} & K_{j+1}^{ff} \end{bmatrix} \hat{x}_k. \tag{4.17}$$

Here K_j^{fb} denotes the feedback component of the estimated controller and K_j^{ff} captures the feedforward properties of the controller. □

Since Q-learning also allows for a direct estimation of the Q-function, there is no need to identify the system matrices A, B and $\mathcal{A}$, the cost matrices P and P_{12} or the one step cost separately. Assuming a quadratic basis from the elements of the state vector $\hat{x}_k$, we can compute a linear relationship between the reward and the Q-function parameters. Solving this, yields a kernel matrix, $\hat{H}_k$ that corresponds to

a parameter matrix,

$$\hat{H}_k = \begin{bmatrix} \hat{A}^\top \hat{P} \hat{A} + \hat{Q} & \hat{B}^\top \hat{P} \hat{A} \\ \hat{A}^\top \hat{P} \hat{B} & \hat{B}^\top \hat{P} \hat{B} + R \end{bmatrix}.$$

The policy improvement step is thereby given as,

$$\tilde{K}_{j+1} = -\hat{H}_{k_{22}}^{-1} \hat{H}_{k_{21}}.$$

This feedback policy $\tilde{K}_{j+1}$ is by definition a stabilizing policy. A new Q-function can be assigned to this policy and the policy improvement procedure can be repeated until desirable control properties of the system are achieved.

Lemma 10. *Given the augmented system* $\left\{\hat{A}, \hat{B}, \hat{C}\right\}$ *that is controllable,* K_1 *stabilizing with associated cost matrix* P_1 *and* K_2 *is the result of one policy improvement step from* K_1*, i.e.* $K_2 = -\gamma \left(R + \gamma B^\top P_1 B\right)^{-1} B^\top P_1 A$*, then*

$$\Delta \| K_1^{ff} - K_2^{ff} \|^2 \leq tr\left(P_{1_{(22)}} - P_{2_{(22)}}\right) \leq \delta \| K_1^{ff} - K_2^{ff} \|^2,$$

where

$$0 < \Delta = \underline{\sigma}\left(R\right) \leq \delta = tr\left(R + \gamma B^\top P_1 B\right) \| \sum_{i=0}^{\infty} \gamma^{(i/2)} \left(A + B K_2\right)^i \|^2,$$

and $\underline{\sigma}\left(\cdot\right)$ *denotes the minimum singular value of a matrix.*

Proof. By Lemma 6, we know that

$$\begin{aligned} P_1 - P_2 &= \sum_{i=0}^{\infty} \gamma^i \left(A + B K_2\right)^{i\top} \left[\left(K_1 - K_2\right)^\top \left(R + \gamma B^\top P_1 B\right) \left(K_1 - K_2\right)\right] \left(A + B K_2\right)^i \\ &\geq \left(K_1 - K_2\right)^\top R \left(K_1 - K_2\right), \end{aligned}$$

since all the sum terms are positive. Taking trace on both sides,

$$\begin{aligned}
\mathrm{tr}\,(P_1) - \mathrm{tr}\,(P_2) &= \mathrm{tr}\,(P_1 - P_2) \\
&\geq \mathrm{tr}\left(\left((K_1 - K_2)^\top \left(R + \gamma B^\top P_1 B\right)(K_1 - K_2)\right)\right) \\
&\geq \underline{\sigma}\,(R)\,\|\,(K_1 - K_2)\,\|^2 \\
&= \underline{\sigma}\,(R)\,\|\left(\begin{bmatrix} K_1^{fb} & K_1^{ff} \end{bmatrix} - \begin{bmatrix} K_2^{fb} & K_2^{ff} \end{bmatrix}\right)\|^2 \\
&= \underline{\sigma}\,(R)\,\|\left(\begin{bmatrix} K_1^{fb} - K_2^{fb} & K_1^{ff} - K_2^{ff} \end{bmatrix}\right)\|^2 \\
&\geq \underline{\sigma}\,(R)\,\|\left(K_1^{ff} - K_2^{ff}\right)\|^2.
\end{aligned}$$

Noting that R is positive definite, we get

$$0 < \mathrm{tr}\,(P_1) - \mathrm{tr}\,(P_2) \geq \Delta\|\left(K_1^{ff} - K_2^{ff}\right)\|^2.$$

Similarly,

$$P_1 - P_2 \leq G^\top (K_1 - K_2)^\top \left(R + \gamma B^\top P_1 B\right)(K_1 - K_2)\, G,$$

where $G = \left(\sum_{i=0}^{\infty} \gamma^{(i/2)}\,(A + BK_2)\right)$. Taking trace on both sides,

$$\begin{aligned}
\mathrm{tr}\,(P_1) - \mathrm{tr}\,(P_2) &= \mathrm{tr}\,(P_1 - P_2) \\
&\leq \mathrm{tr}\left(G^\top (K_1 - K_2)^\top \left(R + \gamma B^\top P_1 B\right)(K_1 - K_2)\, G\right) \\
&\leq \mathrm{tr}\left(R + \gamma B^\top P_1 B\right)\|G\|^2 \|K_1 - K_2\|^2 \\
&= \mathrm{tr}\left(R + \gamma B^\top P_1 B\right)\|G\|^2 \|\left(\begin{bmatrix} K_1^{fb} - K_2^{fb} & K_1^{ff} - K_2^{ff} \end{bmatrix}\right)\|^2 \\
&\leq \mathrm{tr}\left(R + \gamma B^\top P_1 B\right)\|G\|^2 \left(\|K_1^{ff} - K_2^{ff}\|^2 - \|K_1^{fb} - K_2^{fb}\|^2\right) \\
&\leq \mathrm{tr}\left(R + \gamma B^\top P_1 B\right)\|G\|^2 \|K_1^{ff} - K_2^{ff}\|^2.
\end{aligned}$$

Since $\left(R+\gamma B^\top P_1 B\right)$ is positive definite, we get

$$0 < \operatorname{tr}(P_1) - \operatorname{tr}(P_2) \leq \delta \|K_1^{ff} - K_2^{ff}\|^2.$$

Also,

$$\begin{aligned}
\operatorname{tr}(P_1) - \operatorname{tr}(P_2) &= \operatorname{tr}(P_1 - P_2) \\
&= \operatorname{tr}\left(\begin{bmatrix} P_1^{fb} - P_2^{fb} & P_{1_{(12)}} - P_{2_{(12)}} \\ P_{1_{(12)}}^\top - P_{2_{(12)}}^\top & P_{1_{(22)}} - P_{2_{(22)}} \end{bmatrix}\right) \\
&= \operatorname{tr}\left(P_1^{fb} - P_2^{fb}\right) + \operatorname{tr}\left(P_{1_{(22)}} - P_{2_{(22)}}\right) \\
&\geq \operatorname{tr}\left(P_{1_{(22)}} - P_{2_{(22)}}\right).
\end{aligned}$$

□

Theorem 2. *The stabilizing controller $\hat{K}$ estimated for the augmented system converges to the optimal controller K^* with a bias, such that*

$$\lim_{k\to\infty} \|\hat{K}_k - K^*\| = \kappa.$$

Proof. From Theorem 1, the series of stabilizing controllers, $K_k,\ k = 1, 2, 3\ldots$ converge to the optimal feedback control matrix. Define the control matrix for the augmented system,

$$\hat{K}_k = \begin{bmatrix} K_k & K_k^{ff} \end{bmatrix},$$

where K_k^{ff} represents the feedforward component of the control matrix. Let us define another augmented matrix

$$\bar{K} = \begin{bmatrix} K^* & 0 \end{bmatrix},$$

which has the same number of columns as $\hat{K}_k$. Consider

$$\begin{aligned}
\|\hat{K}_k - \bar{K}\| &= \| \begin{bmatrix} K_k & K_k^{ff} \end{bmatrix} - \begin{bmatrix} K^* & 0 \end{bmatrix} \| \\
&= \| \begin{bmatrix} K_k - K^* & K_k^{ff} \end{bmatrix} \| \\
&= \| \begin{bmatrix} K_k - K^* & 0 \end{bmatrix} + \begin{bmatrix} 0 & K_k^{ff} \end{bmatrix} \| \\
&\leq \|K_k - K^*\| + \|K_k^{ff}\| \\
&= \|K_k - K^*\| + \| \left(R + B^\top P B\right)^{-1} B^\top P_{k_{(12)}} \mathcal{A}\|.
\end{aligned}$$

Letting k go to infinity, for some estimation interval $N < \infty$,

$$\begin{aligned}
\lim_{k\to\infty} \|\hat{K}_k - \bar{K}\| &= \lim_{k\to\infty} \left(\|K_k - K^*\| + \| \left(R + B^\top P B\right)^{-1} B^\top P_{k_{(12)}} \mathcal{A}\| \right) \\
&= \lim_{k\to\infty} \|K_k - K^*\| + \lim_{k\to\infty} \| \left(R + B^\top \bar{P} B\right)^{-1} B^\top P_{k_{(12)}} \mathcal{A}\| \\
&= \kappa_k,
\end{aligned}$$

where $\kappa_k = \| \left(R + B^\top \bar{P} B\right)^{-1} B^\top P_{k_{(12)}} \mathcal{A}\|$. For a large enough estimation interval, for targets with constant velocity, $\kappa_k = \bar{\kappa}$. For a more dynamic target with time-varying input, κ_k will tend to slightly vary due to changes in $\mathcal{A}$ and $P_{k_{(12)}}$. □

4.7 Tracking an unknown target with an unmanned aerial vehicle

Figure 4.2 shows a scenario where the augmented system described in (4.14), may be deployed in order to learn the optimal controller that tracks a given target. The target velocity is assumed to be constant for the initial assessment. The Q-functions can be computed through direct interactions with the augmented system which involves exploration around the observed trajectory of the target. The simulation parameters are given in Table 4.1.

Setting a value for N, we follow Algorithm 3 to obtain an acceptable sub-optimal controller. The goal here is to learn a control policy that minimizes the relative

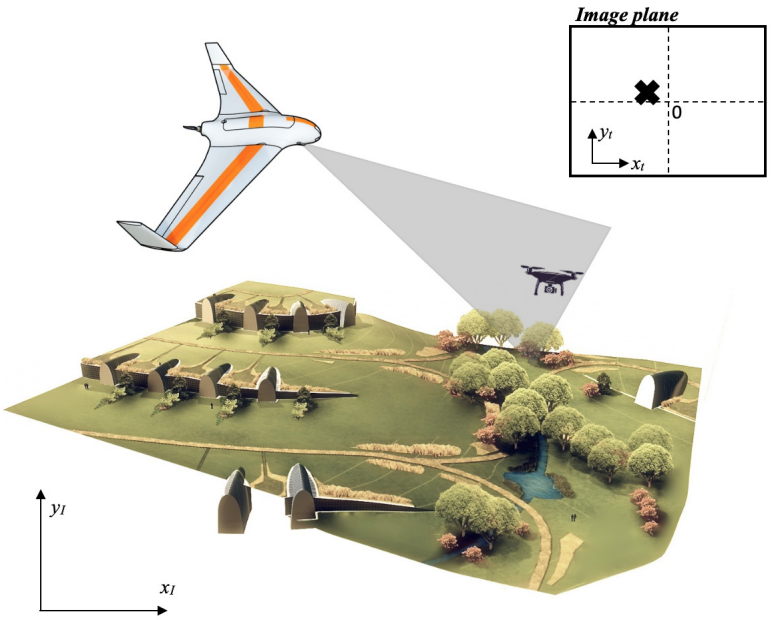


Figure 4.2: Sketch of the target tracking problem.

distance of the aerial vehicle with respect to the observed target. The numerical simulations for a sample trajectory is shown in Figure 4.3. The initial policy K_0 is arbitrarily selected such that it stabilizes the system in (4.14). As mentioned earlier, this initial stabilizing policy may be chosen based on a simple model of the aerial vehicle and iterated over until it stabilizes the system for basic flight conditions.

A random exploration signal is generated from a normal distribution in order to induce persistent excitation of the vector ϕ_k according to the condition in (4.13). Errors induced by the imperfect initial controller and the additional excitation signal drives the cost high during the exploration phase. We choose the exploration phase horizon $N = 80$, about twice the number of parameters in our augmented system. Figure 4.5 shows the norm of the difference between the current controller and the optimal LQR controller at each policy iteration step. The bias seen in the controller convergence is due to the presence of the feed-forward term in the controller for the augmented system. After 8 policy iteration steps, the algorithm has converged enough

Parameter	Description	Value	Units
T_s	Sampling Period	1	s
$\boldsymbol{h}$	UAV altitude	50	m
T	Total Simulation time	500	s
ϵ	Policy convergence threshold	0.0001	-
$\bar{\mathbf{v}}_a$	Max UAV speed	10	m/s
$\bar{\mathbf{v}}_t$	Max target speed	8	m/s

Table 4.1: Simulation Parameters

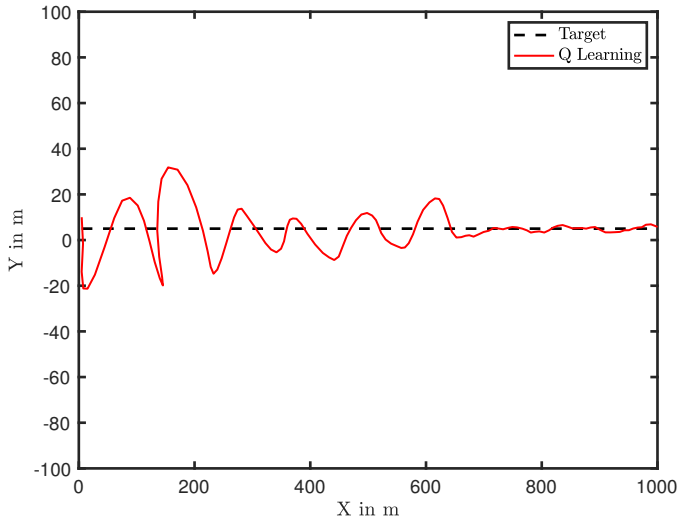


Figure 4.3: Tracking with the augmented system using adaptive policy iteration. The augmented system learns the Q-functions by directly interacting with the environment while maximizing a cost function that penalizes the deviations of the augmented system output from zero. The exploration around the desired trajectory is shown here.

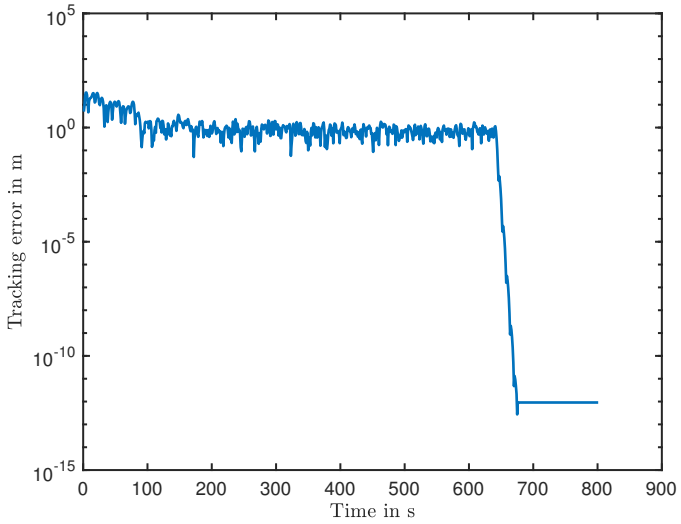


Figure 4.4: The output of the augmented system which is the tracking error. The effect of the persistent excitation is seen in the initial phase of the exploration. Once the policy convergence has been achieved, the learned controller takes over and the error is driven to zero.

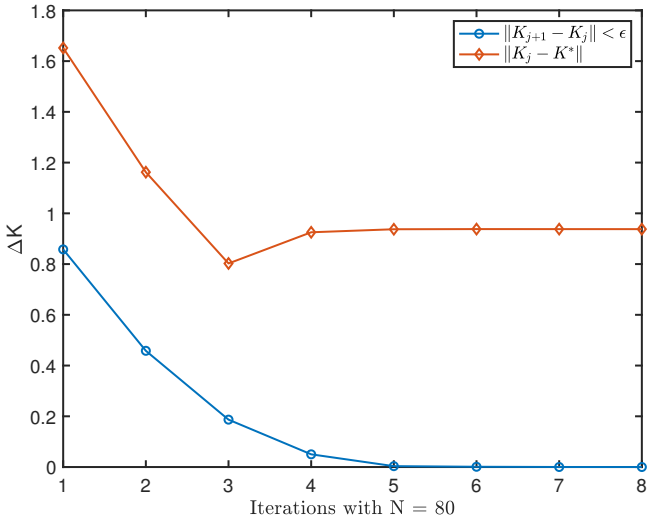


Figure 4.5: Policy iteration convergence. The augmented system eventually learns the optimal controller K^* to drive the tracking error to zero. The bias seen here in the $\|K_j - K^*\|$ is due to the presence of the feedforward component in the learned controller. The policy improvement is also shown to demonstrate the terminating condition.

to see no policy improvement. A new controller is implemented after every N time steps. Each controller in this series of policy improvement iterations is stabilizing and the feedback component of the overall policy eventually converges to the optimal controller. The time-varying component is continuously learned as more samples become available. However, for constant velocity target, a time-invariant controller is learned for this portion as well.

The estimation interval N has a significant effect on the convergence rate and the policy improvement achieved. Figure 4.6shows that with a low value for N, the algorithm does not have enough samples to achieve policy improvement. In fact, in such under-sampled scenarios, these policy iteration steps lead to the divergence of the controller and amplification of the tracking errors. The same effect is seen if the persistent excitation is not maintained. A good strategy to follow here is to slowly cease the additional excitation signal added to the augmented system, so that eventually the effect of only the learned controller is seen. After a few policy improvement steps, in order to exploit the performance of the learned controller, we choose to let the excitation signal decay by selecting an excitation vector of the form, $e_k = \text{rand}(m)/\sqrt{j}$, that diminishes with every iteration j. Due to this, if the value of N is not sufficiently large, the exploratory signal does not provide enough persistent excitation in the subsequent iterations, leading to a destabilizing controller K_{j+1}.

4.8 Conclusions

This chapter focused on the extending the model-free dynamic programming based reinforcement learning approach in the continuous state and action spaces to the problem of tracking with linear quadratic controllers. The adaptive policy iteration algorithm proposed in [27] provided a baseline on which controllers for a tracking system were established that achieved close to zero tracking errors even when models for the tracker system (UAV) and the target were unavailable. The efficacy of directly computing Q-functions was examined against target trajectories with constant and

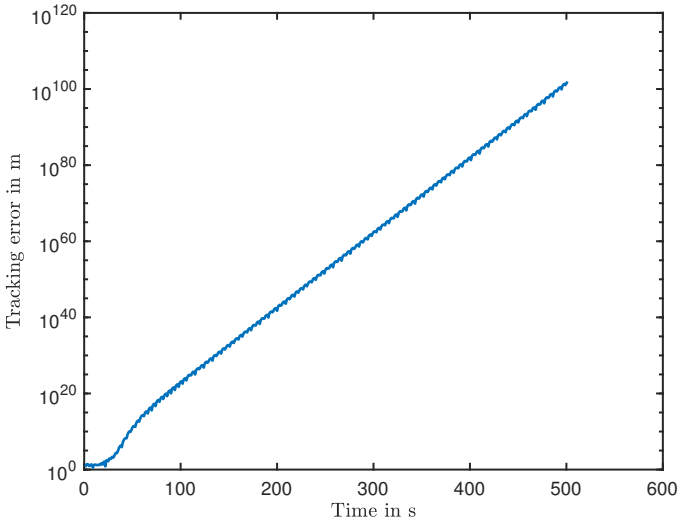


Figure 4.6: Failed convergence and instability when the estimation interval N was set to 25 time steps for a system with 40 system parameters.

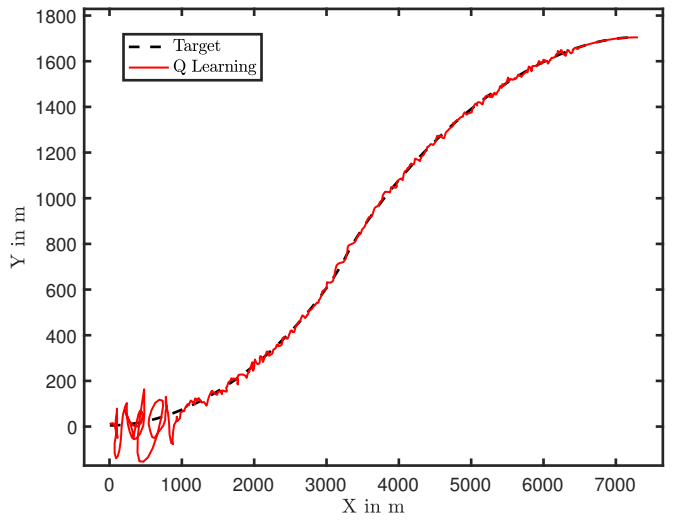


Figure 4.7: Tracking a target with time varying input with the augmented system using adaptive policy iteration. This case needed a more aggressive exploration signal around the desired trajectory to capture the varying nature of the target dynamics.

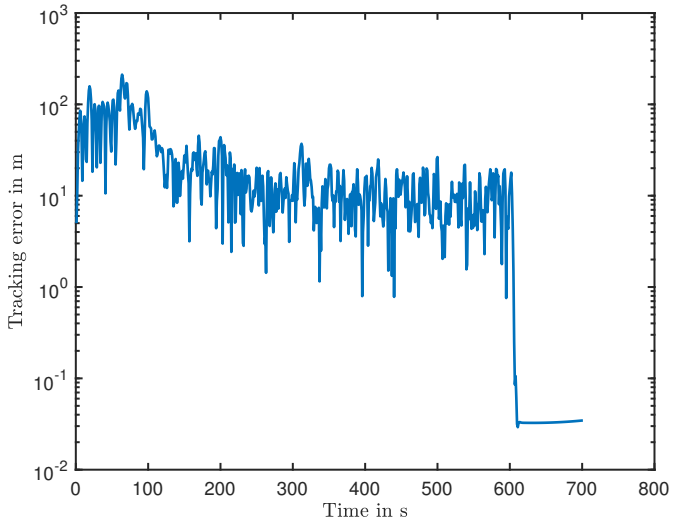


Figure 4.8: Tracking error for time varying target input. The effect of the persistent excitation is seen in the initial phase of the exploration. Once the policy convergence has been achieved, the learned controller takes over and the error is driven closer to zero, although not as closely as the constant velocity case.

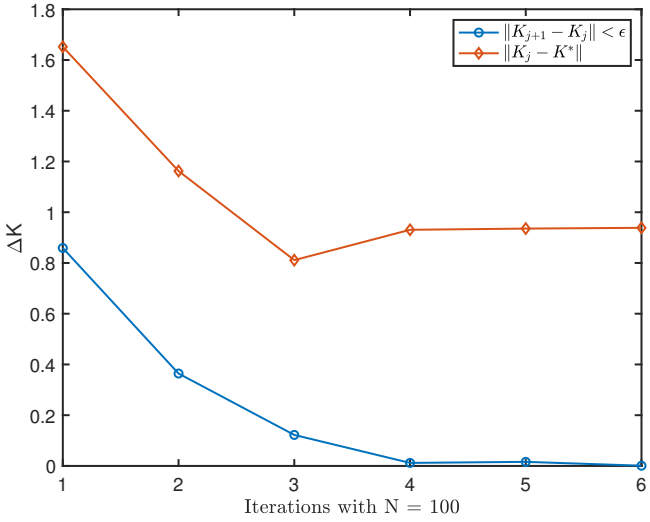


Figure 4.9: Policy iteration convergence. The augmented system eventually learns the sub-optimal controller to drive the tracking error to zero. The bias seen here in the $\|K_j - K^*\|$ is due to the presence of the feedforward component in the learned controller. The terminating condition is relaxed here to be $\epsilon = 0.01$.

time-varying inputs to produce a series of stabilizing controllers that improved tracking performance with each iteration step. The influence of the estimation interval and the persistent excitation signal were evaluated in simulation and the scenarios where the algorithm failed to achieve convergence were also presented. By setting a terminating condition that may yield a sub-optimal controller that performs satisfactorily for various operating conditions, significant computational savings may be achieved. Future work ideas include extending this setup to a distributed framework, along the lines of [56], where the authors explore a distributed Q-learning formulation to design feedback controllers and show that it converges to the optimal LQR controller.

Chapter 5

DISTRIBUTED TRACKING THROUGH DATA-GUIDED CONSENSUS

5.1 Introduction

Complex and dynamic environments like mobile sensor networks, swarms of unmanned aerials, transportation networks and load scheduling are expected to actively adapt to environment and objective changes. Each agent in a network, a single aerial vehicle in a UAV swarm, for example, usually has a limited field of view (FOV) which by itself limits the mission length and control overhead for a task of area surveillance. As a result, efficient motion control of the agent is critical for tasks like target tracking, formation control and area mapping. Nonholonomic constraints like speed limits, turning radius and other commonly enforced constraints increases the complexity of the problem. Moreover, tracking becomes an especially challenging when the target is in certain environments when the onboard UAV's sensor line of sight (LOS) to the target might be occluded by the presence of obstacles such as buildings. In such scenarios in which target tracking can not be maintained by a single UAV, two or more UAVs may be required to accomplish the task.

Prior work in this area includes Model Predictive Control (MPC) [14, 15], vision-based tracking [57, 58], adaptive fuzzy formation control [59], optimal circular flight [60], optimal UAV coordination [7], and active sensing based cooperative tracking [61]. However, these approaches mainly focus on signal processing or when a preview or model of the target trajectory is known. We, however, attempt to leverage the observed target position information in to the control law design by posing a dynamic agreement problem.

We present a framework for a team of two UAVs that perform vision-based data-driven consensus on a mobile ground target's trajectory using a learning-based control strategy. Each agent attempts to locally learn the target's behavior from visual sensor data collected on board and then computes and exchanges target model information with other agents to track the target's motion better. This increased knowledge about the target achieves improved tracking performance, even accounting for occasional occlusions and cases with limited visibility. Our focus here is to find an optimal flight path for each UAV to minimize the time of loss during the moving target tracking.

In this direction, we explore the algorithmic solutions described in the tutorial to solve the dynamic average consensus problem [62]. Here the authors explain - "The algorithm is desired to be

- scalable, so that the amount of computations and resources required on each agent does not grow with the network size,

- robust to the disturbances present in practical scenarios, such as communication delays and packet drops, agents entering/leaving the network, noisy measurements, and

- correct, meaning that the algorithm converges to the exact average or, alternatively, a formal guarantee can be given about the distance between the estimate and the exact average."

It is also indicated that to achieve agreement, there has to be frequent information exchange between the agents. It is not feasible to achieve zero tracking error for aggressive signals as the information of each agent takes some time to propagate through the network. However, if there is some prior information about how the target is expected to behave, a suitable compensating signal may be synthesized. In the problem we want to address, it is assumed that this information is unavailable during control design but may be derived using the data-guided scheme developed in

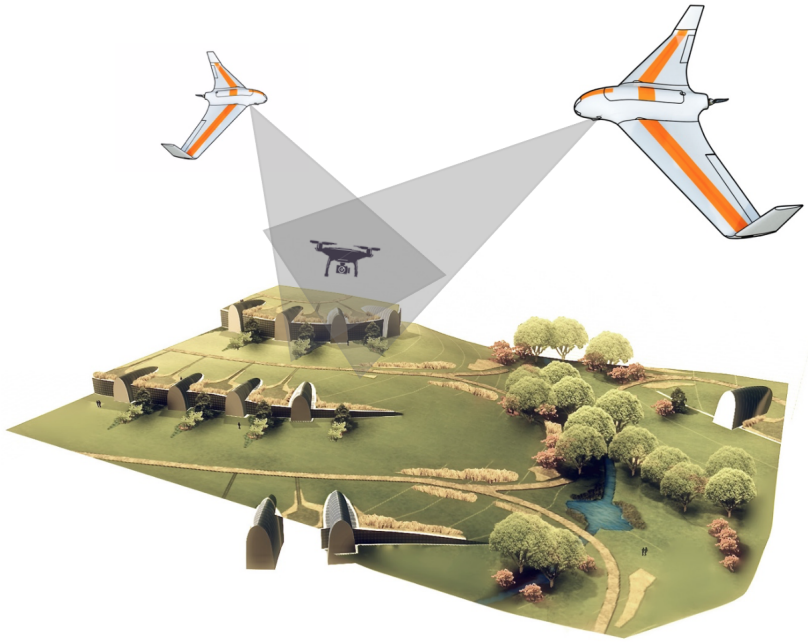


Figure 5.1: Cooperative tracking

Chapter 3. The convergence guarantees of dynamic average consensus algorithms are exploited to achieve distributed tracking of unknown targets.

5.2 Problem Description

Assume homogenous aerial vehicles as described in (2.1), where $j = 1, 2, \ldots, N_v$. We consider the performance of the UAV when flying at the constant height $\boldsymbol{h}_j$. In order to avoid collision with each other the UAVs are set to have different altitudes, such that $d_{safe} < \|\boldsymbol{h}_i - \boldsymbol{h}_j\|_2 < d_r$, for some reasonable d_{safe} in meters that is within the communication range d_r of the UAVs.

Assume target altitude $\boldsymbol{h}_t < \min\left(\boldsymbol{h}^j\right),\ j = 1, 2 \ldots, Nv$. If the sensor field of view (FOV) angle of the UAV j is defined to be 2θ degrees, then the search area covered

by the camera on board the UAV can be represented by the linear function $c(\boldsymbol{h}_j)$,

$$c(\boldsymbol{h}_j) = \begin{cases} 0 & \text{if } \boldsymbol{h}_j < \boldsymbol{h}_0 \\ \boldsymbol{h}_j \tan\theta & \text{if } \boldsymbol{h}_0 \leq \boldsymbol{h}_j \leq \boldsymbol{h}_{max} \\ 0 & \text{if } \boldsymbol{h} > \boldsymbol{h}_{max} \end{cases} \tag{5.1}$$

where $(\boldsymbol{h}_0, \boldsymbol{h}_{max})$ is the altitude range where visual based sensing is viable and yields reasonable resolution of the target for the image processing software to make target position estimates with low error covariances. Alternatively, the projected planar position of the UAV j, x_j^k, at time step k is at the center of the circle with radius $c(\boldsymbol{h}^j)$. When the relative planar distance between the UAV and the target, $d_k^j = \left\|x_k^j - \tilde{x}_k\right\|_2 = 0$, the UAV j is directly above the target and the measurement error is the smallest. The target is "visible" to the UAV, if $d_k^j < c(\boldsymbol{h}^j)$. As $d^j \to c(\boldsymbol{h}^j)$, the associated error covariance grows and becomes significantly elongated in the viewing direction. The UAV j attempts to minimize d^j, which is essentially the tracking error.

Mathematically, each UAV tries to minimize the quadratic cost,

$$V(T, x_0^j) = \sum_{k=0}^{T-1} (x_k^j - \tilde{x}_k)^\top Q (x_k^j - \tilde{x}_k) + u_k^{j\top} R u_k^j + (x_T^j - \tilde{x}_T) P_T^j (x_T^j - \tilde{x}_T),$$

subject to its dynamics in (2.1) for some finite horizon T. Following the data-guided approach described in Chapter 3, we build the artificial dynamical system that captures the dominant modes of the target motion.

5.3 Learning target motion

The data-guided approach developed before directly embeds raw observation data with no prior assumptions about the target model. Each UAV j samples the current position of the target every T_s seconds (sampling period) which is appropriately chosen based on the processing power and memory available on board each agent. The

measurement vector is given as $\bar{y}_k^j = y_{g_k} + w_k,\ w_k \sim \mathcal{N}\left(0, \sigma_j^2\right),\ j = 1, 2$, and creates datasets X^j and Y^j as described in (3.1),

$$\begin{aligned} X_k^j &= \begin{bmatrix} \rho^{k-1}\bar{y}_1^j & \rho^{k-2}\bar{y}_2^j & \cdots & \bar{y}_{k-1}^j \end{bmatrix} \\ Y_k^j &= \begin{bmatrix} \rho^{k-1}\bar{y}_2^j & \rho^{k-2}\bar{y}_3^j & \cdots & \bar{y}_k^j \end{bmatrix} \end{aligned} \tag{5.2}$$

for some scaling factor $\rho,\ 0 < \rho \leq 1$ and some time $k = m$ number of snapshots of measurement data. The UAV then computes the linear operator $\mathcal{A}_k^j$, such that

$$Y_k^j = \mathcal{A}_k^j X_k^j$$

by minimizing the quantity

$$\left\| Y_k^j - \mathcal{A}_k^j X_k^j \right\|_F^2$$

subject to a stability constraint as described in Algorithm 1. We employ the rank-1 updates described in [2] in order to update this approximation. The update equations for $\mathcal{A}_k^j$ when a new snapshot pair becomes available are

$$\mathcal{A}_{k+1}^j = \mathcal{A}_k^j + \gamma_{k+1}\left(\bar{y}_{k+1}^j - \mathcal{A}_k^j \bar{y}_k^j\right) \bar{y}_k^{j\top} \tilde{P}_k^j,$$

where

$$\begin{aligned} \tilde{P}_{k+1}^j &= \frac{1}{\rho}\left(\tilde{P}_k^j - \gamma_{k+1}\tilde{P}_k^j \bar{y}_k^j \bar{y}_k^{j\top} \tilde{P}_k^j\right), \\ \gamma_{k+1} &= \frac{1}{1 + \bar{y}_k^{j\top} \tilde{P}_k^j \bar{y}_k^j}. \end{aligned} \tag{5.3}$$

This update ensures we gradually "forget" older snapshots due to the weighting parameter ρ.

5.4 Target system consensus

The aerial vehicles each collect target position data for a fixed number of time steps through the vision system that records target location in each image frame. This data is subject to the uncertainty that comes with the respective visual sensors. In order to improve our knowledge about the target, we can perform consensus through information fusion on one of these three levels.

1. Data level (Measurements): We can obtain a combined dataset by averaging over the datasets that each UAV collects and exchanges and then compute the linear operator. This, however, involves exchanging a vector of m snapshots, which increases space complexity.

2. Model level (Parameters): Once each agent computes the linear operator from the measurement data, the UAVs can perform consensus on the linear operator to obtain a combined estimate of the operator. This is a 2×2 matrix exchange, that encodes the target information.

3. Outcome level (Estimates): Once each UAV obtains the operator from the data, a preview of the immediate future trajectory may be exchanged and fused to perform reference tracking of this combined prediction of the future target path.

There has been extensive work in the first and the third levels of information fusion. We explore the second option here to examine how well performing consensus on the learned model fares as compared with each UAV tracking the target separately. The agents compute their own local approximation of the target model from the observation data that were collected on board and perform updates to this approximation at every time step as new data become available. This is a time-varying parameter signal over which the agents would ideally wish to reach an agreement. One simple way is to combine the two linear operators $\mathcal{A}_k^{(i)}$ and $\mathcal{A}_k^{(j)}$ computed by two UAVs i

and j using a simple weighted average for identical sampling rates and is given by,

$$\hat{\mathcal{A}}_k = \frac{1}{\lambda_i + \lambda_j}(\lambda_i \mathcal{A}_k^{(i)} + \lambda_j \mathcal{A}_k^{(j)}), \tag{5.4}$$

where $\lambda_i > 0$ and $\lambda_j > 0$ are weights that represent the relative importance of each UAV's estimate. As described in [62,63], this dynamic agreement problem corresponds to *dynamic average consensus*. The dynamic average consensus problem is for a group of agents to cooperate in order to track the average of locally available time-varying reference signals, where each agent is only capable of local computations and communicating with local neighbors. We will explore this idea in the context of learned model parameters which may be time-varying in nature.

5.5 Dynamic Average Consensus

The tutorial on dynamic average consensus [62] iterates that the difficulty in the dynamic average consensus problem is that the information is distributed across the network. In contrast, the centralized approach gathers all data, computes averages and redistributes them to the different nodes of the network. However, this centralized scheme suffers from robustness issues, scalability problems, time delays subject to the size of the network, security concerns and the existence of a single point of failure in the network. The dynamics average consensus problem only involves local interactions and decisions among the agents.

The *static average consensus* problem, as described in [62], is one in which the agents seek to agree on a specific combination of fixed quantities. This has been extensively studied in the literature [64–67], and several simple and efficient distributed algorithms exist with exact convergence guarantees. Instead of using a static average consensus algorithm between sampling times, which will require fast convergence within the specified sampling times, the dynamic average consensus algorithm keeps a memory of past actions and produces a better tracking response than the static

Algorithm 4 Discrete time dynamic average consensus. Adapted from [62].

1: Input: ℓ_k^i and $\eta_k^{ij},\ j \in \mathcal{N}_{out}(i)$
2: Output: $\mathcal{A}_{k+1}^i,\ \ell_{k+1}^i,\ \eta_{k+1}^{ij}$
3: $\mathcal{A}_{k+1}^i \leftarrow c^i\left(\ell_k^i, \eta_k^{ij}\right)$
4: Generate ℓ_{k+1}^i and η_{k+1}^i
5: Broadcast η_{k+1}^i

algorithm initialized at each sampling time with the current values.

The dynamic average consensus problem consists of designing an algorithm that allows individual agents to track the time-varying average of the reference signals, given by

$$u_k^{avg} = \frac{1}{N}\sum_{i=1}^{N} u_k^i,$$

where each agent computes the signal u_k^i which may be the output of a sensor located on the agent, or it could be the output of another algorithm that the agent is running. In the problem of our interest, the agents compute new updates for $\mathcal{A}_k^j$ at each k from local observations. We seek a command or an update that accomplishes $\mathcal{A}_t^i \to \mathcal{A}_k^{avg}$ as $t \to \infty$, where t is the sampling interval $(k, k+1)$ that only depends on local computation ℓ_k^i and the information obtained from the agents' neighbors $\eta_k^{ij},\ j \in \mathcal{N}_{out}(i)$.

The consensus update (5.4) may be performed every few time steps or whenever the UAVs are able to exchange information. The update horizon, N_h may have an effect on the tracking accuracy, which we'll examine with numerical simulation on some synthetic data. This approach is summarized as the distributed data-guided consensus algorithm in Algorithm and the improvement in the tracking performance is evaluated in Section 5.7.

Algorithm 5 Distributed data-guided consensus algorithm for cooperative tracking.

1: For each agent j, initialize discounted data matrices $\tilde{X}^j$ and $\tilde{Y}^j$, from m target position snapshots data with variance σ_j^2 and some value of ρ and update horizon N.
2: Compute linear operator $\mathcal{A}_k^j$ from the quadratic program (3.9) subject to constraints (3.13), such that $Y_k^j \approx \mathcal{A}_k^j X_k^j$.
3: **for** time $k = m + 1$ **to** end
4: Obtain combined $\hat{\mathcal{A}}_{avg}$ from if $k = \alpha N_h,\ \alpha \in \mathbb{N} = \{1, 2, \ldots\}$.
5: Compute control laws K^j and $\hat{K}^j$ and optimal control u_k^{j*} for the augmented state system in (3.22).
6: Obtain new UAV state x_{k+1}^j from u_k^{j*} and the new target snapshot $\bar{y}_{k+1}^j$.
7: Update $\mathcal{A}_k$ with the new snapshot using the rank-1 update.
8: **end for**

5.6 Convergence Analysis

Assumption: The agents are represented as the nodes of a strongly connected undirected graph.

5.6.1 Static average consensus

The agents start their agreement state, $\mathcal{A}_k^i$ with their own local approximation of the target from noisy measurements. They then adjust this parameter based on some weighted linear feedback which takes into account the difference between their agreement state and of their adjacent neighbors. This leads to algorithms of the following form

$$\mathcal{A}_{k+1}^i = \mathcal{A}_k^i - \sum_{j=1}^{N} a_{ij} \left(\mathcal{A}_k^i - \mathcal{A}_k^j \right), \tag{5.5}$$

for $i \in \{1, 2, \ldots, N\}$ and $\boldsymbol{A} = [a_{ij}]_{N \times N}$ is the adjacency matrix of the communication graph. The graph Laplacian matrix is

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A},$$

where $\boldsymbol{D}$ is the degree matrix. The problem in (5.5) can be compactly written as

$$\mathbb{A}_{k+1} = -\boldsymbol{L}\mathbb{A}_k, \tag{5.6}$$

with $\mathbb{A}_0$ is initial approximations of $\mathcal{A}_0^i$ stacked together. The following theorem arises for the static average consensus problem.

Theorem 3. *[62] As $k \to \infty$, every agreement state $\mathcal{A}_k^j,\ j \in \{1, 2, \ldots, N\}$, of the static average consensus problem converges to $\mathcal{A}^{avg}$ with an exponential rate no worse than $\varrho \in (0, 1)$, provided the Laplacian matrix satisfies $\varrho = \|\boldsymbol{I}_N - \boldsymbol{L} - \mathbf{1}_N\mathbf{1}_N^\top/N\|_2 < 1$.*

Since the reference signals enter the static average consensus algorithms (5.5) as initial conditions, they cannot track time-varying parameters. The dynamic average consensus formulation continuously injects the signals as inputs into the dynamical system. This allows the system to naturally respond to changes in the signals without any need for re-initialization.

5.6.2 Dynamic average consensus

The time-varying reference signals enter the formulation in (5.6) as an external input given as,

$$\mathbb{A}_{k+1} = -\boldsymbol{L}\mathbb{A}_k + \mathbb{U}_k, \quad \mathcal{A}_0^i = \mathcal{U}_0^i, \tag{5.7}$$

which can also be written as

$$\mathcal{A}_{k+1}^i = \mathcal{A}_k^i - \sum_{j=1}^{N} a_{ij}\left(\mathcal{A}_k^i - \mathcal{A}_k^j\right) + \mathcal{U}_k^i,$$

$$\mathcal{A}_0^i = \mathcal{U}_0^i,\ i \in \{1, 2, \ldots, N\}. \tag{5.8a}$$

We assume that the communication graph is constant, connected, and undirected. The Laplacian matrix is then symmetric and therefore has real eigenvalues. Since the graph is connected, the smallest eigenvalue is $\lambda_1 = 0$ and all the other eigenvalues are

Parameter	Description	Value	Units
T_s	Sampling Period	2	s
$\boldsymbol{h}$	UAV altitude	$(60, 70, 90)$	m
T	Total Simulation time	60	s
m	Number of snapshots	8	-
σ_t^2	Sensor Noise Variance	$(1^2, 2^2, 3^2)$	m^2
$\mathbf{v}_a$	Max UAV speed	20	m/s
$\mathbf{v}_t$	Max target speed	18	m/s

Table 5.1: Simulation Parameters

strictly positive, in other words, $\lambda_2 > 0$. The discretized dynamic average consensus algorithm has the iterations

$$p_{k+1}^i = p_k^i + k_I \sum_{j=1}^{N} a_{ij} \left(\mathcal{A}_k^i - \mathcal{A}_k^j \right), \; p_0^i \in \mathbb{R}^{\dim(\mathcal{A}_k)}, \; i \in \{1, \ldots, N\},$$

$$\mathcal{A}_k^i = \mathcal{U}_k^i - p_k^i, \tag{5.9a}$$

where k_I is the step size.

Theorem 4. *[62] Let $\mathcal{G}$ be a connected, undirected graph. The agreement states $\mathcal{A}_k^i, \; i \in \{1, \ldots, N\}$ converge to $\mathcal{A}^{avg}$ exponentially with rate $\varrho = \dfrac{\lambda_N - \lambda_2}{\lambda_2 + \lambda_N}$ with $k_I = \dfrac{2}{\lambda_2 + \lambda_N}$ and the algorithm is initialized such that the average of the initial integrator states is zero, that is, $\sum_{i=1}^{N} p_0^i = 0$.*

5.7 Simulation results

Consider three homogenous agents that can be represented by an undirected graph as shown in Figure 5.2. The system equations for each agent is described in (2.1) and it is assumed that they maintain flying altitudes, $\boldsymbol{h}^j, \; j = 1, \ldots, 3$. Let us begin by considering a target moving at a constant velocity. Each UAV obtains different noisy measurements of the same target and performs consensus on the learned linear oper-

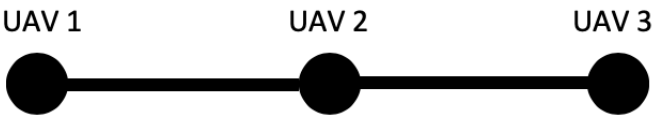


Figure 5.2: A simple path graph showing the communication network for the UAVs.

ator. The trajectories and tracking errors are presented in Figure 5.7. The consensus update is performed every $N = 5$s to keep a realistic scenario under consideration. The value of N may vary based on the strength of the communication link between the agents. The approach in Algorithm 5 was applied to a target trajectory with 100 different values of random, normally distributed sensor noise and different update lengths. As expected, infrequent updates to the target model information increases the average combined tracking error over time, but the error remains lower than what is achievable with the two UAVs separately.

For a small time-varying control input, this data-guided approach with model consensus still outperforms the one without, making a case for combining the locally computed linear operators to obtain better tracking. The tracking error vs. update horizon N for this is shown in Figure 5.9 along with the target velocity profile. For very aggressive trajectories or higher noise variance, however, the performance significantly degrades due to the amplification of the error. One factor that influences this behavior is the values of ρ. $\rho = 1$ remembers all snapshots which, for aggressive trajectories and very noisy data, could lead to bad estimates of the linear operator. An appropriate value of ρ considers recent snapshots and weights old snapshots considerably lower.

As Figure 5.9 shows, the average tracking error remains on the lower side compared

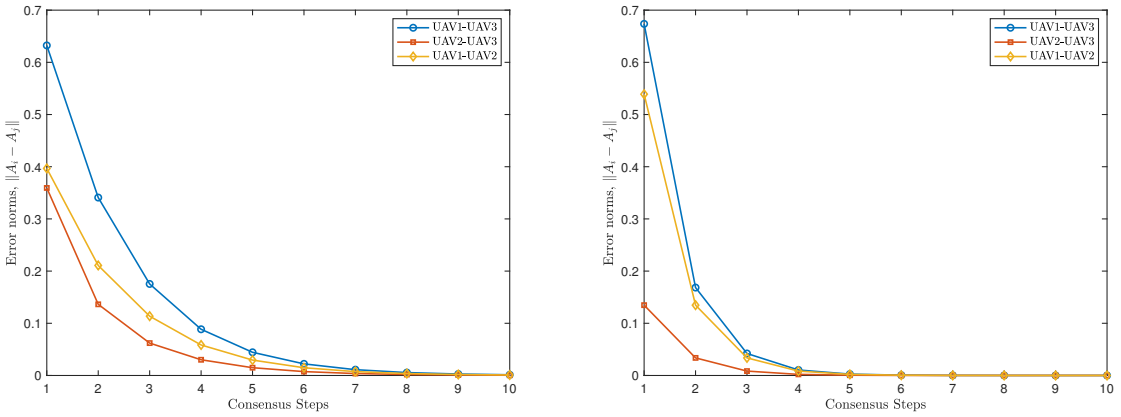


Figure 5.3: Convergence of the linear operator for three agents with noisy measurements with variance of $2m^2$, $3m^2$ and $4m^2$ respectively with $k_I = 1$ and 1.5 .

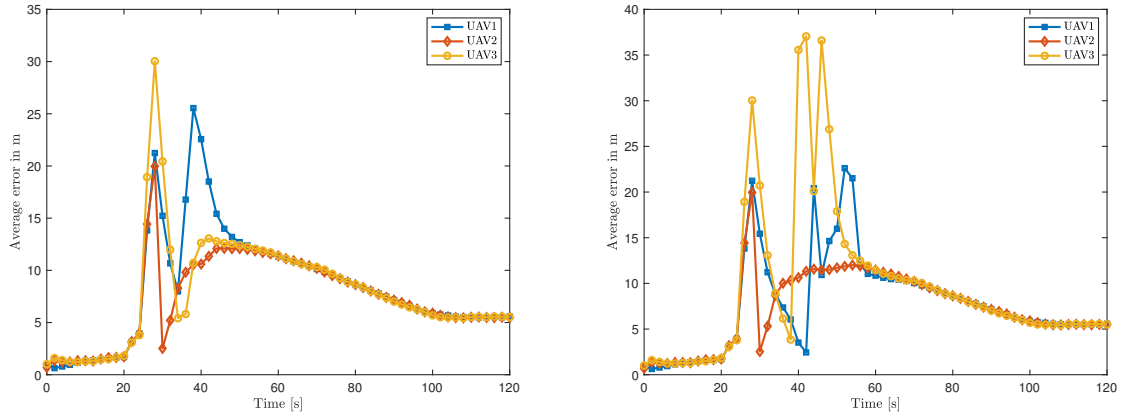


Figure 5.4: Convergence of the tracking errors three agents with noisy measurements with variance of $2m^2$, $3m^2$ and $4m^2$ respectively with $k_I = 1$ and 1.5 .

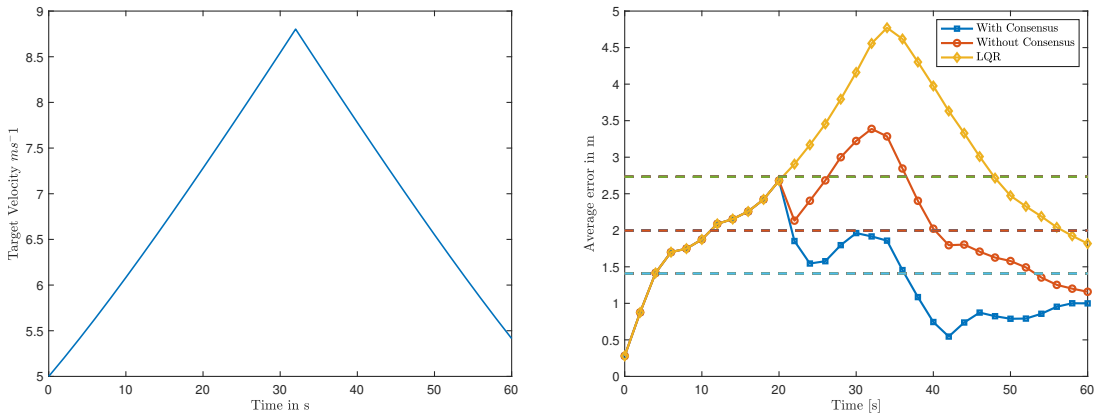


Figure 5.5: Average minimum error for a slow target over 100 trials.

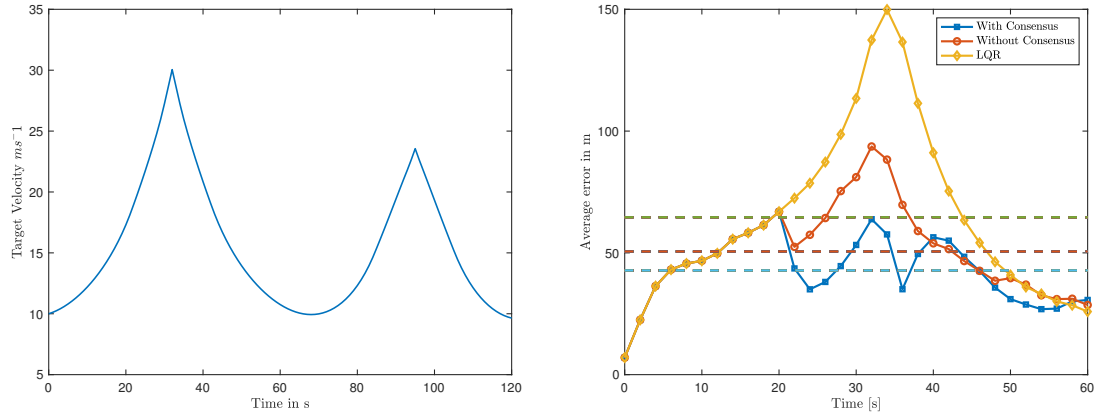


Figure 5.6: Average minimum error for a more aggressive target over 100 trials.

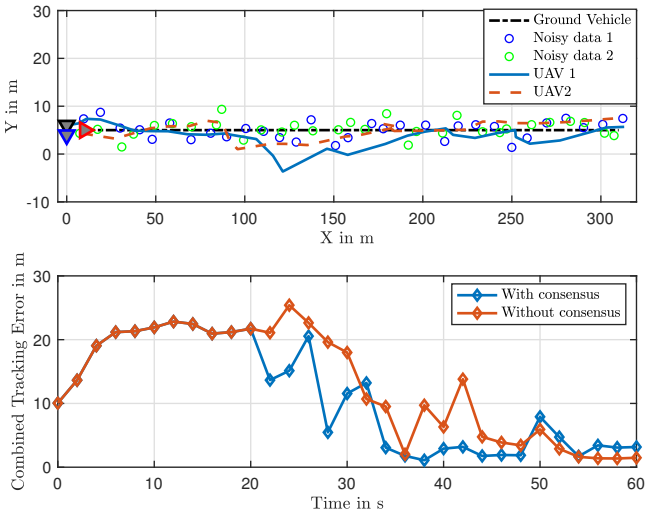


Figure 5.7: Two agents tracking a target traveling with a constant velocity with measurement variance of $2m^2$ and $3m^2$ respectively. Consensus horizon, $N_h = 5$s.

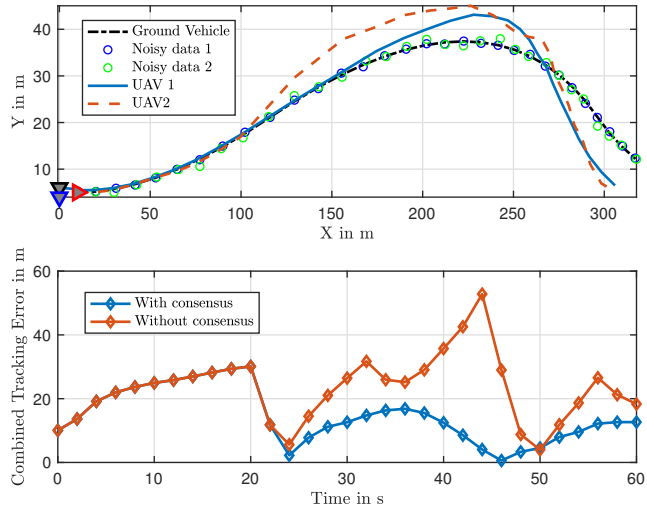


Figure 5.8: Two agents tracking a target traveling at some control input with measurement variance of 2m and 3m respectively. Consensus horizon, $N = 5$s.

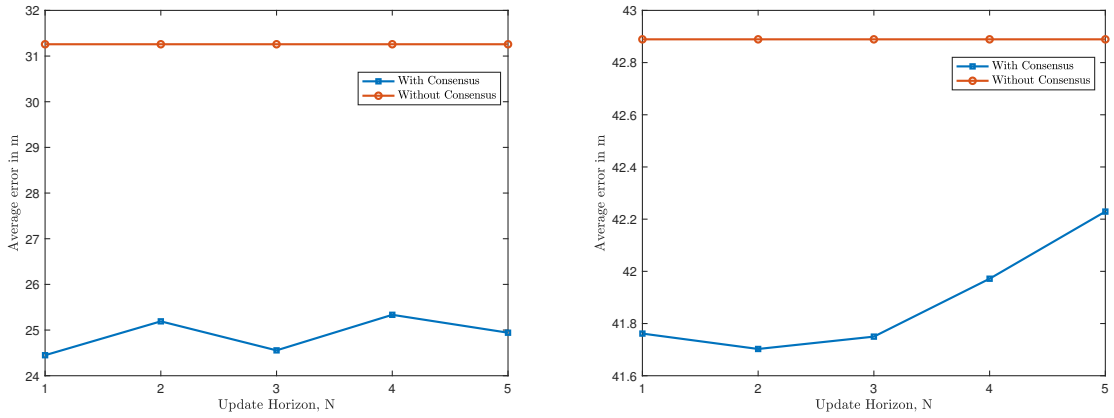


Figure 5.9: Average minimum error over 100 trials for different update horizons, N for the slow moving and aggressive targets.

to the scenario with no information exchange. For slower targets, the average error varies very little and remains fairly constant. On the other hand, the case with more aggressive targets, infrequent updates eventually affects tracking performance, but this degradation still produces better outcome than without consensus. In order to quantify this relationship between the consensus update horizon and the tracking performance, we can start by looking to the consensus error bounds proposed in [62]. This further detail will be explored in a future extension of this work.

5.8 Conclusions

This chapter explored a novel approach for information fusion for a group of aerial vehicles cooperatively tracking an unknown target. The information exchanged is the set of parameters that each agent estimates for the observed target from data with different levels of noise. These estimates are inherently time-varying in nature due to the target motion. The agents perform dynamic average consensus on this time-varying signal and agree on an average set of parameters that each agent then continues to refine. The convergence rate for a simple path graph was analyzed.

The tracking errors for the case where each agent operates independently and while exchanging information were presented and it was shown that exchanging target model parameters reduces the average tracking error by a significant by over 20-50% for slow trajectories compared with the data-guided approach presented in Chapter 3 and standard LQR. These numbers were about 10%-30% for more aggressive trajectories. Another interesting outcome was the benefit of infrequent consensus updates, since the errors were lower even with delaying exchange of information. A possible future extension would be to evaluate sensitivity of this algorithm for further delays in updates and potential dropping off of some agents.

Chapter 6

CONCLUSIONS

6.1 Summary

This work bridges several disciplines including optimal control theory, dynamical systems and statistical learning. Where conventional methods fail or perform less than satisfactorily, these algorithms provide a way to embed observed or measured data in to the decision-making process to enhance performance of these systems. As we embark on a more data-rich world, these methods are a good step towards improving safety, performance and robustness of the systems that may one day become more ubiquitous.

The main take-aways from this work are as follows:

1. This research work explored the use of learned local linear models for the target output data for aerial tracking. The design may be applicable to a small fixed wing UAV with a gimbaled visual sensing system. This data-guided approach was cast as a standard LQ tracking problem, where the target information is unavailable, but for the noisy position measurements at each time step from the on board cameras.

2. This result was extended to a distributed setup with multiple aerial vehicles tracking a target. For this, by means of dynamic average consensus, the different agents exchange the parameters of the local target models that capture the target's behavior, contrary to what most consensus-based methods do, which is exchange of measurements or estimates.

3. A model-free formulation for learning policy parameters directly from interactions with the target was explored to track the target without any knowledge of UAV system and the target. The continuous-time Q-learning formulation, where an adaptive policy iteration approach learns an optimal controller, was applied to the augmented UAV system.

6.2 Future Directions

This preliminary investigation opens up a lot of avenues for theoretical and experimental possibilities for aerial robotics. For instance, we could scale the distributed framework presented in Chapter 5 for multiple UAVs AND multiple targets within a region of interest. This will require a data association method to classify the object in the current image frame. Video-based multiple object tracking algorithms may be leveraged to include the moving camera platform. The layout of the targets may be observed at each time step to determine the formation topology. Once that is achieved, we can attempt to track the centroid of this formation. A potential application is wildlife or wildfire tracking. A federated learning setup could be explored where each agent learns only a part of the target state through observations from limited sensing capabilities. A distributed estimation algorithm may be developed that learns a better model from the fragmented sub-models.

BIBLIOGRAPHY

[1] S. M. Siddiqi, B. Boots, and G. J. Gordon, "A Constraint Generation Approach to Learning Stable Linear Dynamical Systems," *Tech report CMU-ML-08-101*, 2008.

[2] H. Zhang, C. W. Rowley, E. A. Deem, and L. N. Cattafesta, "Online dynamic mode decomposition for time-varying systems," *arXiv preprint arXiv:1707.02876*, 2017.

[3] H. J. Kim, D. H. Shim, and S. Sastry, "Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 5, pp. 3576–3581, IEEE, 2002.

[4] G. R. Rodríguez-Canosa, S. Thomas, J. del Cerro, A. Barrientos, and B. MacDonald, "A real-time method to detect and track moving objects (datmo) from unmanned aerial vehicles (uavs) using a single camera," *Remote Sensing*, vol. 4, no. 4, pp. 1090–1111, 2012.

[5] J. Hu, L. Xie, K. Y. Lum, and J. Xu, "Multiagent Information Fusion and Cooperative Control in Target Search," *IEEE Transactions on Control Systems Technology*, vol. 21, pp. 1223–1235, July 2013.

[6] S. Zhu and D. Wang, "Cooperative Ground Target Tracking with Input Constraints," in *Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on*, pp. 1051–1056, Dec 2010.

[7] S. A. P. Quintero, F. Papi, D. J. Klein, L. Chisci, and J. P. Hespanha, "Optimal UAV Coordination for Target Tracking using Dynamic Programming," in *49th IEEE Conference on Decision and Control (CDC)*, pp. 4541–4546, Dec 2010.

[8] F. Lin, X. Dong, B. M. Chen, K. Y. Lum, and T. H. Lee, "A Robust Real-Time Embedded Vision System on an Unmanned Rotorcraft for Ground Target Following," *IEEE Transactions on Industrial Electronics*, vol. 59, pp. 1038–1049, Feb 2012.

[9] İ. Güvenç, O. Ozdemir, Y. Yapici, H. Mehrpouyan, and D. Matolak, "Detection, localization, and tracking of unauthorized uas and jammers," in *2017*

IEEE/AIAA 36th Digital Avionics Systems Conference (DASC), pp. 1–10, Sep. 2017.

[10] J. Schiff, "Automated intruder tracking using particle filtering and a network of binary motion sensors," in *in IEEE International Conference on Automation Science and Engineering*, pp. 1–2, 2006.

[11] Y. Wu, Y. Sui, and G. Wang, "Vision-based real-time aerial object localization and tracking for uav sensing system," *IEEE Access*, vol. 5, pp. 23969–23978, 2017.

[12] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, "Vector Field Path Following for Miniature Air Vehicles," *IEEE Transactions on Robotics*, vol. 23, pp. 519–529, June 2007.

[13] E. W. Frew, D. A. Lawrence, C. Dixon, J. Elston, and W. J. Pisano, "Lyapunov Guidance Vector Fields for Unmanned Aircraft Applications," in *2007 American Control Conference*, pp. 371–376, July 2007.

[14] S. A. P. Quintero, D. A. Copp, and J. P. Hespanha, "Robust UAV Coordination for Target Tracking using Output-Feedback Model Predictive Control with Moving Horizon Estimation," in *2015 American Control Conference (ACC)*, pp. 3758–3764, July 2015.

[15] Z. He and J. Xu, "Moving target tracking by uavs in an urban area," in *2013 10th IEEE International Conference on Control and Automation (ICCA)*, pp. 1933–1938, June 2013.

[16] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[17] E. A. Wan and R. Van Der Merwe, "The Unscented Kalman Filter for Nonlinear Estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pp. 153–158, Ieee, 2000.

[18] J. N. Juang and R. S. Pappa, "An Eigensystem Realization Algorithm for Modal Parameter Identification and Model Reduction," *Journal of Guidance, Control, and Dynamics*, vol. 8, pp. 620–627, 1985.

[19] J. N. Juang, M. Phan, L. G. Horta, and R. W. Longman, "Identification of Observer/Kalman Filter Markov Parameters: Theory and experiments," *Technical Memorandum 104069, NASA*, 1991.

[20] S. Arora, E. Hazan, H. Lee, K. Singh, C. Zhang, and Y. Zhang, "Towards provable control for unknown linear dynamical systems," 2018.

[21] R. Boczar, N. Matni, and B. Recht, "Finite-data performance guarantees for the output-feedback control of an unknown system," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 2994–2999, IEEE, 2018.

[22] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu, "On the sample complexity of the linear quadratic regulator," *arXiv preprint arXiv:1710.01688*, 2017.

[23] M. Hardt, T. Ma, and B. Recht, "Gradient descent learns linear dynamical systems," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 1025–1068, 2018.

[24] E. Hazan, K. Singh, and C. Zhang, "Learning linear dynamical systems via spectral filtering," in *Advances in Neural Information Processing Systems*, pp. 6702–6712, 2017.

[25] M. Fazel, R. Ge, S. M. Kakade, and M. Mesbahi, "Global convergence of policy gradient methods for the linear quadratic regulator," *arXiv preprint arXiv:1801.05039*, 2018.

[26] S. Tu, R. Boczar, A. Packard, and B. Recht, "Non-asymptotic analysis of robust control from coarse-grained identification," *arXiv preprint arXiv:1707.04791*, 2017.

[27] S. J. Bradtke, "Incremental Dynamic Programming for On-Line Adaptive Optimal Control," *PhD thesis*, 1994.

[28] B. O. Koopman, "Hamiltonian Systems and Transformation in Hilbert Space," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 7, p. 315, 1931.

[29] I. Mezić, "Spectral Properties of Dynamical Systems, Model Reduction and Decompositions," *Nonlinear Dynamics*, vol. 41, pp. 309–325, 2005.

[30] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, "On Dynamic Mode Decomposition: Theory and Applications," *Journal of Computational Dynamics*, Nov 2013.

[31] S. T. Dawson, M. S. Hemati, M. O. Williams, and C. W. Rowley, "Characterizing and Correcting for the Effect of Sensor Noise in the Dynamic Mode Decomposition," *Experiments in Fluids*, vol. 57, pp. 1–19, 2016 2016.

[32] L. Ma and N. Hovakimyan, "Cooperative target tracking in balanced circular formation: Multiple uavs tracking a ground vehicle," in *2013 American Control Conference*, pp. 5386–5391, June 2013.

[33] D. Kingston, "Decentralized control of multiple uavs for perimeter and target surveillance," 2007.

[34] C. Peterson and D. A. Paley, "Multivehicle coordination in an estimated time-varying flowfield," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 1, pp. 177–191, 2011.

[35] S. Kim, H. Oh, and A. Tsourdos, "Nonlinear model predictive coordinated standoff tracking of a moving ground vehicle," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 2, pp. 557–566, 2013.

[36] B. D. O. Anderson and J. B. Moore, *Optimal Control-Linear Quadratic Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1990.

[37] K. Pereida, R. R. P. R. Duivenvoorden, and A. P. Schoellig, "High-precision trajectory tracking in changing environments through L1 adaptive feedback and iterative learning," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 344–350, 2017.

[38] S. Vasisht and M. Mesbahi, "A Data-Driven Approach for UAV Tracking Control," *ASME Dynamic Systems and Control Conference*, Oct 2017.

[39] A. Mauroy and I. Mezić, "Global stability analysis using the eigenfunctions of the koopman operator," *IEEE Transactions on Automatic Control*, vol. 61, pp. 3356–3369, Nov 2016.

[40] Y. Lan and I. Mezić, "Linearization in the large of nonlinear systems and koopman operator spectrum," *Physica D: Nonlinear Phenomena*, vol. 242, no. 1, pp. 42 – 53, 2013.

[41] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A Data–driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition," *Journal of Nonlinear Science*, vol. 25, no. 6, pp. 1307–1346, 2015.

[42] P. J. Schmid, "Dynamic Mode Decomposition of Numerical and Experimental Data," *Journal of Fluid Mechanics*, vol. 656, pp. 5–28, 2010.

[43] M. H. Gutknecht, *A Brief Introduction to Krylov Space Methods for Solving Linear Systems*, pp. 53–62. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[44] T. C. Hsia, *System identification*. Lexington Books, 1977.

[45] D. M. Hawkins, "The problem of overfitting," *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[46] S. L. Lacy and D. S. Bernstein, "Subspace identification with guaranteed stability using constrained optimization," *IEEE Transactions on Automatic Control*, vol. 48, pp. 1259–1263, July 2003.

[47] P. J. Olver, "Numerical analysis lecture notes," 2005.

[48] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[49] M. S. Hemati, M. O. Williams, and C. W. Rowley, "Dynamic mode decomposition for large and streaming datasets," *Physics of Fluids*, vol. 26, no. 11, p. 111701, 2014.

[50] J. Sherman and W. J. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *Ann. Math. Statist.*, vol. 21, pp. 124–127, 03 1950.

[51] W. Hager, "Updating the inverse of a matrix," *SIAM Review*, vol. 31, no. 2, pp. 221–239, 1989.

[52] A. Mauroy and J. Goncalves, "Linear Identification of Nonlinear Systems: A Lifting Technique based on the Koopman Operator," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 6500–6505, Dec 2016.

[53] J. Skaf and S. Boyd, "Analysis and Synthesis of State-Feedback Controllers With Timing Jitter," *IEEE Transactions on Automatic Control*, vol. 54, pp. 652–657, March 2009.

[54] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 2nd ed., 2000.

[55] G. Goodwin and K. Sin, *Adaptive filtering prediction and control*. Prentice-Hall information and system sciences series, Prentice-Hall, 1984.

[56] S. Alemzadeh and M. Mesbahi, "Distributed q-learning for dynamically decoupled systems," *arXiv preprint arXiv:1809.08745*, 2018.

[57] Z. Li, N. Hovakimyan, V. Dobrokhodov, and I. Kaminer, "Vision-based target tracking and motion estimation using a small uav," in *49th IEEE Conference on Decision and Control (CDC)*, pp. 2505–2510, Dec 2010.

[58] M. Zhang and H. H. T. Liu, "Vision-based tracking and estimation of ground moving target using unmanned aerial vehicle," in *Proceedings of the 2010 American Control Conference*, pp. 6968–6973, June 2010.

[59] V. Gazi and B. Fidan, "Adaptive formation control and target tracking in a class of multi-agent systems: Formation maneuvers," in *2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)*, pp. 78–85, Oct 2013.

[60] J. Kim and Y. Kim, "Optimal circular flight of multiple uavs for target tracking in urban areas," 2009.

[61] Lin Wang, Fei Su, Huayong Zhu, and Lincheng Shen, "Active sensing based cooperative target tracking using uavs in an urban area," in *2010 2nd International Conference on Advanced Computer Control*, vol. 2, pp. 486–491, March 2010.

[62] S. S. Kia, B. V. Scoy, J. Cortés, R. A. Freeman, K. M. Lynch, and S. Martínez, "Tutorial on dynamic average consensus: the problem, its applications, and the algorithms," *CoRR*, vol. abs/1803.04628, 2018.

[63] M. Zhu and S. Martínez, "Discrete-time dynamic average consensus," *Automatica*, vol. 46, no. 2, pp. 322 – 329, 2010.

[64] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, pp. 215–233, Jan 2007.

[65] W. Ren and R. W. Beard, *Distributed Consensus in Multi-vehicle Cooperative Control: Theory and Applications*. Springer Publishing Company, Incorporated, 1st ed., 2007.

[66] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *IEEE Control Systems Magazine*, vol. 27, pp. 71–82, April 2007.

[67] W. Ren and Y. Cao, *Distributed Coordination of Multi-agent Networks: Emergent Problems, Models, and Issues*. Springer Publishing Company, Incorporated, 2013.