Team 07: Car-Tooning

ESE 451 Senior Design Final Report

Authors

Alexandre Amice: EE and Math
amice@seas.upenn.edu
Sophia Moses: EE and Finance
sophiamo@seas.upenn.edu
Sergio Roman: EE and Math
seroman@seas.upenn.edu
Matteo Sciolla: CMPE
matteos@seas.upenn.edu

Advisor

Manfred Morari: ESE Faculty Fellow
morari@seas.upenn.edu

April 29, 2020

Contents

I Executive Summary

Traffic delays cost Americans an estimated $45 billion of productivity per year. Additionally, idling in traffic causes significant wear and tear on vehicles and as well as substantially increases the likelihood of being involved in an accident. Traffic is getting worse, increasing by more than 40 hours per year during peak travel times in America's most congested cities [1]. These statistics reflect a growing problem of gridlock in major cities as global populations rise. A recent Fehr and Peers study [2] demonstrated that one reason for this increase in congestion around the world is the rise of ride-sharing platforms such as Uber and Lyft. In a race to meet the global demand of mobility on call, these companies have encouraged a surplus of drivers to operate on the roads. Such a strategy stands to only worsen traffic as these companies leverage the emerging technology of autonomous vehicles to reduce wait times and costs.

Unfortunately, public attempts to rectify this growing traffic catastrophe have failed to properly curb the problem. Many cities do not have the money or resources to significantly improve infrastructure by either expanding roadways or through the adoption of smart-city technologies. Additionally, the latter technologies raise serious privacy concerns about mass surveillance and have met public opposition. Therefore, public efforts have largely come in the form of managed lane strategies which attempt to encourage specific driving behaviors. The most popular of these is the high occupancy vehicle (HOV) lane, which reserves lanes for vehicles with a minimum occupancy. Unfortunately, such strategies have been shown to worsen traffic in many cases due to the underutilized road surface [3]. Other strategies such as congestion pricing misalign incentives by commoditizing traffic rather than eliminating it [4].

Autonomous vehicles promise to solve the issue in a way that HOV lanes have failed through the formation of platoons. A platoon is a group of vehicles which maintain constant communication to safely travel at high speeds with very small inter-vehicle space. In addition to improvements in roadway throughput via this behavior, platoons improve the aerodynamic properties and fuel efficiency of the vehicles involved, reducing pollution [5]. The theory of platooning is well studied and promises to reduce traffic. However, these idealized methods studied in theory typically fail in realistic scenarios where the unreliability of wireless networks and the natural actuation delay present in control systems degrades the performance relative to the idealized model. In more adversarial conditions, the extra delays caused by realistic wireless networks and the slow response time of vehicles within the platoon can lead to instability within the network. This instability manifests itself as ever-growing oscillations in speed and inter-vehicle distances, leading to even worse traffic and increased fuel consumption [6].

Research on robustifying classical control schemes against the aforementioned issues with implementing platooning on real systems is quite common. However, few studies attempt to design a control scheme to handle as broad a range of challenges as Car-Tooning. Guided by the goal of implementing the theorized controller on a real, specific, low quality system, Car-Tooning is a controller designed without abstracting away any of the challenges of online networked control. Phase one of the project was to design the Car-Tooning controller and gain experience in platooning through software simulation. Phase two of the project attempted to implement this scheme on low quality cars and networks and modify the controller to perform in realistic conditions.

II Overview

Traffic is a society-wide problem which costs billions of dollars a year in lost productivity and environmental pollution. Vehicular platooning is a well studied method that promises to significantly reduce traffic by

allowing networks of cars to coordinate their motions more fluidly. However, current state of the art methods fail to safely enable platooning in realistic situations such as unreliable networks and dynamic road conditions. Car-Tooning is a distributed platooning controller that seeks to be safer and robust enough to operate in these conditions through the implementation of a predictive control scheme designed with realistic systems in mind. Car-Tooning enables even low quality hardware to platoon efficiently. This project advances the theory of platoons in autonomous vehicles thus reducing traffic.

III Technical Description

3.1 Introduction

The primary objective of platooning is to coordinate the velocity and distance of a set vehicles satisfying spacing requirements [7]. Satisfying this time-headway specification is trivially achieved using simple feedback controllers if zero time delays exist within the system [8]. However, the control of a distributed network of agents invariably suffers from communication delays, introducing an overall delay in the networked control system and thus inducing the possibility of instability [9]. Therefore, the primary objective of the vehicular platooning problem is to design a controller which achieves string stability [10] in the presence of time-varying network delays. When one can control both the design of the network and the controller, this leads to a codesign problem which can be solved in a variety of ways as illustrated in [11].

Unfortunately, the codesign of a controller and a network can be quite challenging as typically networks are public infrastructures and the limitations of network throughput are governed by physical hardware and bandwidth limitations which are difficult and costly to improve [12]. Therefore, it is beneficial to design controllers which accept the network as a static fixture which cannot be improved. Rather than codesigning a network and a controller, Car-Tooning seeks to design a controller and a minimal information flow topology which will enable the efficient platooning of vehicles subject to realistic network delays. This project designs a model of the necessary information which must be exchanged between vehicles to achieve efficient, string-stable platooning in the presence of inevitable, uncontrollable network delays on very low quality hardware. The purpose of using poor hardware is to demonstrate the potential of this controller to perform well even in the worst case scenario. We are further motivated by using this minimalist approach as the future adoption of autonomous vehicles will result in heterogeneous operating conditions for the network as various quality and cost of vehicular hardware will be available to consumers.

This section is primarily concerned with the technical developments necessary to synthesize our controller. We begin by outline a set of specifications governing our design in section 3.2. Next, in section 3.3 we describe the trade-offs in selecting different design paradigms for the Car-Tooning controller and justify our selection of an optimization-based, discrete-time, model predictive controller. We follow with a complete description of the controller in section 3.4. The description is left in its abstract formulation enabling its implementation on a wide range of systems. In section 3.5, we describe the system in which we chose to attempt to test the Car-Tooning controller. This includes a description of the hardware components as well as the necessary software developments to enable the minimum level of autonomy necessary to test the platoon.

Notation: Lowercase and uppercase boldface letters represent column vectors ($\boldsymbol{x}$) and matrices ($\boldsymbol{X}$) respectively with the i^{th} vector and matrix of a collection denoted by $\boldsymbol{x}^{(i)}$ and $\boldsymbol{X}^{(i)}$. Lowercase letters (x) will denote scalars, with the i^{th} entry of a vector $\boldsymbol{x}$ denoted by x_i and the i, j entry of a matrix $\boldsymbol{X}$ denoted by X_{ij}. $\boldsymbol{X}_{i:k,j:l}$ will be used to denote the submatrix of $\boldsymbol{X}$ constructed i to k^{th} rows and j to l^{th} columns of the matrix $\boldsymbol{X}$ with : denoting the entire set of indices $0 : n$ of that dimension.

Sets will be denoted using calligraphic letters ($\mathcal{X}$). $\|\boldsymbol{x}\|_k$ and $\|\boldsymbol{X}\|_k$ will be used to denote the k-norm

and k-entrywise norm respectively where k is assumed to be 2 when omitted. $|\mathcal{X}|$ will be used to denote the cardinality of $\mathcal{X}$. Finally $\boldsymbol{x} \succ k$ ($\boldsymbol{x} \succeq k$) will denote the elementwise greater than (or equal to) of the vector (i.e. $x_i > k$ for $i = 1, \dots n$) while for a symmetric matrix $\boldsymbol{X} \succ 0$ ($\boldsymbol{X} \succeq 0$) will be used to denote positive (semi)-definite matrix with $\mathcal{X} \succeq \mathcal{Y}$ denoting the partial ordering in the positive semi-definite cone. We write $\lambda_{\min}$ and $\lambda_{\max}$ to denote the minimum and maximum eigenvalues of a matrix respectively.

Finally, we denote by $vstack\left(\left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{n}\right)$ ($vstack\left(\left\{\boldsymbol{X}^{(i)}\right\}_{i=1}^{n}\right)$) the vectors (matrices) constructed by stacking the vectors (matrices) into a column i.e. $\begin{bmatrix} \boldsymbol{x}^{(1)} \\ \boldsymbol{x}^{(2)} \\ \vdots \\ \boldsymbol{x}^{(n)} \end{bmatrix}$. Similarly $hstack\left(\left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{n}\right)$ stacks the vectors horizontally, i.e. $\begin{bmatrix} \boldsymbol{x}^{(1)} & \boldsymbol{x}^{(2)} & \dots & \boldsymbol{x}^{(n)} \end{bmatrix}$.

3.2 Specifications

Platooning is enabled by two state of the art technologies: autonomous vehicles (AVs) and wireless communications. In section X we outline various standards relevant to informing the development of a platooning controller. Due to the safety-critical nature of autonomous platooning, it is paramount that our controller have sufficient margin for redundant, fail-safe control informed by these standards. This sections serves to translate those standards into a list of concrete specifications for our controller design. The results of this section are succinctly summarized in table 3.2.

We begin by assuming our platoon will operate within a 3G network. 4G communication networks are capable of communicating at 100Mbps with a delay variation of 2-40ms and 0.1% packet drop rate [13]. Meanwhile, the 3G specifications allow for a mere 348kbps of transmitted data for moving vehicles [14]. Though typical operation will occur in a 4G network, we choose to constrain ourselves to the latter specification of 348kbps providing us with a factor of over 200 safety margin for our communication speed. This specification does not translate into longer communication delays, but rather a constraint on the maximal data each vehicle can transmit to other vehicles in the platoon. As we will see next, our controller will be constrained to output control decisions within 20ms, leaving us with 80ms of time to gather data. Therefore, a maximum of 27.8kb can be transmitted by a local chain in the platoon.

As autonomous vehicles output control decisions on 10Hz control loops, the speed of output decisions is paramount. As we plan to use an optimization based approach to control synthesis, we expect allow ourselves 20ms to compute a control decision. This allows the vast majority of the system compute time to be spent on state estimation which is typical of autonomous systems.

As the control of AVs is safety-critical, the testing and validation of the scheme of paramount importance. We therefore constrain ourselves to using a model-based, convex optimization driven solution. This ensures that at all possible system states, the controller outputs a unique solution or a provable certificate of infeasibility. This differs from a learning based approach which typically cannot generate such guarantees. Such a technique is what allows us to ensure that our controller can generate certificates of stability.

Finally, we focus on generating an abstract, parametric solution that is scalable to an arbitrary size. The former requirement ensures that our design remain relevant as communication and AV technology improves. The latter ensures that our design remain relevant as the market of AVs grows and can indeed aid in city-wide traffic reduction.

Criterion	Standard	Specification
Communicated Data Efficiency	IMT-2000 [14]	27.8kbits or fewer communicated within the platoon
Control Speed	IEC 61508 [15]	20ms computation of control loop
Decision Reliability	2050-2018 IEEE [16]	Deterministic, model based solution methods
Robustness and Stability	IEC 61508 [15]	Provable certificates of stability
Scalability	N/A	Support arbitrarily large platoon
Next Generation Versatility	USDOT [17] and FCC Roadmaps [18]	Abstract solution capable of integrating next-gen communication and control standards

Table 1: Summary of desired controller specifications and relevant standards

3.3 Design Considerations

When designing a novel networked controller, several different design paradigms need to be considered each with their own trade offs. In this section we discuss three paradigms considered and how this choice influenced our design.

There exist two primary methods for designing feedback controllers. The classical approach is to design a proportional-integral-derivative (PID) feedback loop where in an error is computed based on the deviation of a set-point value from the process value and the corrective input is applied based on a proportional, integral, and derivative evolution of this error. A typical PID loop can be seen in figure 3.3. The design burden of such controller is placed on the tuning of the PID gains which though challenging can be guided by a variety of established methods [19].

Due to its simplicity, PID control can output decisions in microseconds on modern chips and can be vigorously tested in a wide variety of conditions. Unfortunately, this method typically employs fixed gains for each term which can be proven to not be string stable [20]. String stability is discussed in further detail in section 3.4.5.

By contrast, the modern approach to control synthesis is based on the theory of convex optimization. In this framework, the process value is driven to a set point by trading off deviations from the set point in the future versus control energy in the present. The maturity of the field of optimization enables the natural incorporation of physical actuation limits, safety constraints, and a wide variety of other conditions [21]. Additionally, through the use of the available system model, this framework can employ foresight by planning for the dynamics in the future rather than only the current deviation.

However, this framework shifts the design burden from tuning controller gains to designing the appropriate cost function to achieve the desired behavior which can be difficult due to the high dimension of most cost penalties. Additionally, the online evaluation of potentially high-dimensional optimization problems can be challenging to achieve quickly.

Car-Tooning chooses to employ the latter framework nonetheless as it does not compute a static feedback controller, but rather varies its gain over the entirety of the allowable system state (see figure 3.3). The

non-static nature of the controller enables it to overcome the short-comings of PID control.

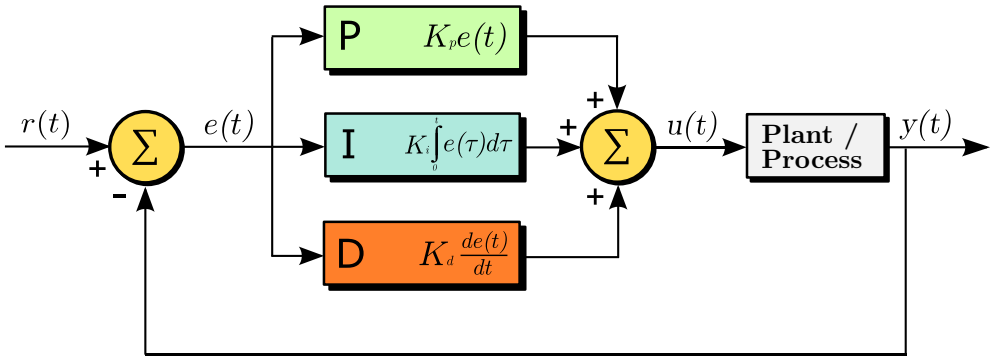

Figure 1: PID control tunes static gains in order to achieve the desired behavior

The next design consideration is whether to design a continuous or discrete time controller. Continuous time controllers have the advantage of most faithfully modeling the physical world. As such, continuous time controls should be designed at the lowest level of the control stack possible particularly for very sensitive systems where fine grained control is necessary.

However, designing continuous time controllers in an optimization based framework can be quite challenging. The continuous nature of the dynamics typically results in semi-infinite programs [22] which are difficult to analyze and solve. Moreover, the digital nature of most computing environments necessitates the discretization of the control inputs even if the problem admits a closed form solution.

By contrast, discrete time controllers are easily implemented onto modern digital computers and the finite nature of the resulting optimization problems enables the use of a wide-variety of both general and specialized optimization solvers. The use of a sufficiently fine discretization and long enough time horizons can typically eliminate most of the draw backs of discrete time control and thus Car-Tooning employs a discrete time strategy after appropriate discretization (see section 3.4.2).

Finally, we must decide whether to us a distributed or centralized scheme for generating control decisions throughout the platoon. In a centralized control scheme, a central agent collects information and then distributes control decisions to each agent. Such a scheme has the clear advantage in theory as it can generate the unique, globally optimal solution as it has full state information of the entire platoon. However, in practice as the platoon grows the increase in physical distances leads to slower communication speed as well as larger system state. The former degrades performance as control actions are performed on old information, while the latter degrades performance as the growing system state incurs larger overhead when solving the programs which are synthesizing the controller.

By contrast, in a distributed scheme, agents share information with other nearby agents and each agent generates local control decisions based on the observed information. Such a scheme generally outputs control

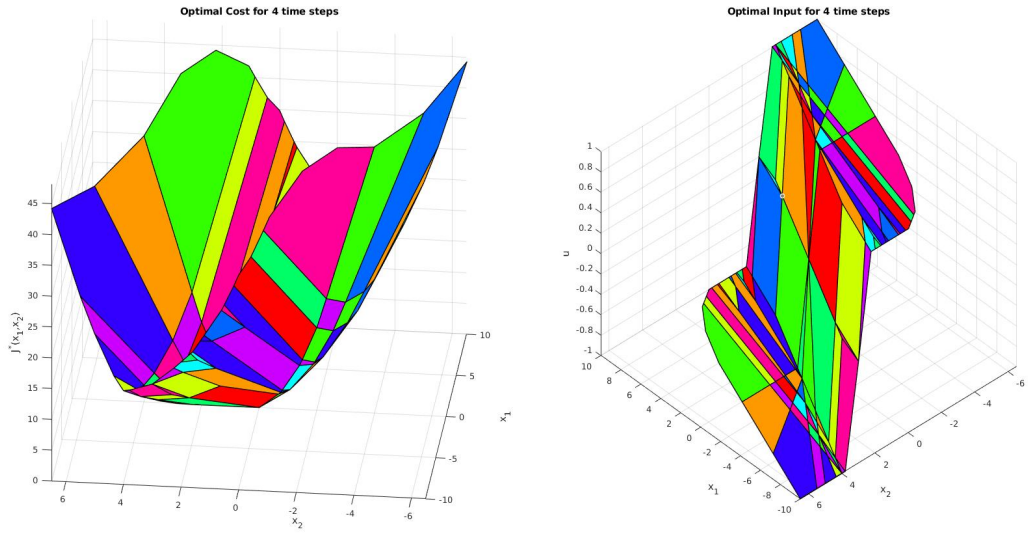

Figure 2: Modern control designs focuses on tuning cost functions to achieve state-dependent control

decisions which are suboptimal with respect to the global cost function, yet does not suffer from the increased communication and computation times as the platoon grows, allowing the system to scale to arbitrary size.

Car-Tooning is therefore designed as a discrete-time, predictive, distributed controller.

Remark: Note that our choice of control scheme was primarily driven by technical considerations and not by societal, economic, and environmental concerns. This is because platooning technology is enabled by AV technology which already is a high-cost technology with significant compute. The computational burden of platooning software will not require a more sophisticated hardware suite than will already be available on such vehicles and thus incur no additional economic cost. Moreover, the model-based design approach does not require massive computation before deployment as is typical of learning-based approaches and can emit as much carbon as five vehicles in their lifetime [23].

3.4 Car-Tooning Controller

3.4.1 Control of Chain Graphs

We consider the generalized control of a chain of agents according to the framework proposed in [24]. The authors from [24] give a general method of providing locally optimal control of the distributed system:

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} \boldsymbol{A}^{00} & 0 & 0 & \dots & 0 \\ \boldsymbol{A}^{10} & \boldsymbol{A}^{11} & 0 & \dots & 0 \\ 0 & \boldsymbol{A}^{21} & \boldsymbol{A}^{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 \ddots & \boldsymbol{A}^{N,N-1} & \boldsymbol{A}^{NN} & \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} \boldsymbol{B}^{(0)} & 0 & 0 & \dots & 0 \\ 0 & \boldsymbol{B}^{(1)} & 0 & \dots & 0 \\ 0 & 0 & \boldsymbol{B}^{(2)} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \boldsymbol{B}^{(N)} \end{bmatrix} \boldsymbol{u}(t) \tag{1}$$

subject to the linear quadratic regulator (LQR) cost:

$$J = \boldsymbol{x}(t_f)^T \boldsymbol{P} \boldsymbol{x}(t_f) + \int_{t_0}^{t_f} \boldsymbol{x}(t)^T \boldsymbol{Q} \boldsymbol{x}(t) + \boldsymbol{u}(t)^T \boldsymbol{R} \boldsymbol{u}(t) dt \tag{2}$$

where $\boldsymbol{P} \succeq 0$, $\boldsymbol{Q} \succeq 0$ and $\boldsymbol{R} \succ 0$ assuming the network is experiencing no time delays. Local optimality does not imply global optimality which is generally intractable for very large versions of this problem.

If the networked control system given in (1) can be controlled with a centralized control strategy, it is well known [25] that the unconstrained optimal control strategy is given by:

$$\boldsymbol{u}^*_{cen} = -\boldsymbol{K} \boldsymbol{x}(t) \tag{3}$$

$$\boldsymbol{K} = \boldsymbol{R}^{-1} \boldsymbol{B}^T \boldsymbol{P} \tag{4}$$

$$\dot{\boldsymbol{P}} = \boldsymbol{P} \boldsymbol{B} \boldsymbol{R}^{-1} \boldsymbol{B}^T \boldsymbol{P} - \boldsymbol{A}^T \boldsymbol{P} - \boldsymbol{P} \boldsymbol{A} - \boldsymbol{Q} \tag{5}$$

This centralized policy is impractical due to the large delays incurred by centralizing the information, solving the very large problem, and then universally distributing the controller. Thus the authors [24] derive a decentralized, locally optimal algorithm which is replicated in appendix 1.1 for reference. This solution is globally asymptotically stable provided each coupled subsystem consisting of vehicles $\boldsymbol{x}^{(i-1)}(t)$ and $\boldsymbol{x}^{(i)}(t)$ are stable for all times t. We note that this asymptotic stability is not sufficient to guarantee string stability which is a more important notion of stability for platoons. A discussion of string stability is deferred until section 3.4.5.

The incorporation of state and input constraints can be naturally introduced on the optimization problem. This prevents obtaining a closed form solutions described in algorithm 1, instead requiring iterative optimization techniques. We address the solution of this problem after discussing the plant discretization in 3.4.2.

3.4.2 Networked Control Subject to Time Delays

Given a continuous time, networked system such as the one described in (1), the authors in [26] describe a method for incorporating uncertain network time delays longer than the plant discretization through a semi-definite programming (SDP) approach. Consider a linear time invariant (LTI), continuous time system with sampling time T_s, input actuation delay $\tau' < T_S$ caused by the network, and input control constraints $u_{min} \leq \left\| \boldsymbol{u}(t) \right\|_\infty \leq u_{\max}$, i.e. the continuous time model

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}^C \boldsymbol{x}(t) + \boldsymbol{B}^C \boldsymbol{u}(t - \tau') \tag{6}$$

$$\text{subject to } u_{min} \leq \left\| \boldsymbol{u}(t) \right\|_\infty \leq u_{\max} \tag{7}$$

The plant discretization with sampling time T_s and indexes k is achieved [27] via:

$$\boldsymbol{x}((k+1)T_s) = \exp(\boldsymbol{A}^C T_s) \boldsymbol{x}(kT_s) + \int_{kT_s+\tau'}^{(k+1)T_s} \exp\left(\boldsymbol{A}^C((k+1)T_s - s)\right) \boldsymbol{B}^C \boldsymbol{u}(kh) ds + \int_{kT_s}^{kh+\tau'} \exp\left(\boldsymbol{A}^C((k+1)T_s - s)\right) \boldsymbol{B}^C \boldsymbol{u}((k-1)h) ds \tag{8}$$

To simplify notation, we introduce the following matrices:

$$\begin{aligned} \boldsymbol{A}^D = \exp(\boldsymbol{A}^C T_s), \quad \boldsymbol{B}^D_+(\tau') = \int_{T_s-\tau'}^{T_s} \exp\left(\boldsymbol{A}^C(r)\right) \boldsymbol{B}^C dr \\ \boldsymbol{B}^D_-(\tau') = \int_0^{T_s-\tau'} \exp\left(\boldsymbol{A}^C(r)\right) \boldsymbol{B}^C, \quad \boldsymbol{z}(kh) = \boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{x}(kh) \\ \boldsymbol{u}((k-1)h) \end{bmatrix} \end{aligned} \tag{9}$$

where the change of variable $r \coloneqq (k+1)T_s - s \Rightarrow dr = -ds$ is made in the integrations. This results in the discretized plant with augmented state:

$$\boldsymbol{z}_{k+1} = \begin{bmatrix} \boldsymbol{A}^D & \boldsymbol{B}_+^D(\tau') \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \boldsymbol{z}_k + \begin{bmatrix} \boldsymbol{B}_-^D(\tau') \\ \boldsymbol{I} \end{bmatrix} \boldsymbol{u}_k \tag{10}$$

$$\boldsymbol{z}_{k+1} = \boldsymbol{A}(\tau')\boldsymbol{z}_k + \boldsymbol{B}(\tau')\boldsymbol{u}_k \tag{11}$$

In order to handle cases where the delay is $\tau = (d-1)T_S - \tau'$ for $d \in \{1, 2 \dots\}$ and $0 \leq \tau' < T_S$, we perform the same discretization, but augment the state further. The discrete time system becomes:

$$\begin{bmatrix} \boldsymbol{x}((k+1)T_s) \\ \boldsymbol{u}((k+1-d)T_s) \\ \vdots \\ \boldsymbol{u}((k-1)T_s) \\ \boldsymbol{u}(kT_s) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}^D & \boldsymbol{B}_+^D(\tau') & \boldsymbol{B}_-^D(\tau') & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \dots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(kT_s) \\ \boldsymbol{u}((k-d)T_s) \\ \boldsymbol{u}((k-d-1)T_s) \\ \vdots \\ \boldsymbol{u}((k-2)T_s) \\ \boldsymbol{u}((k-1)T_s) \\ \boldsymbol{u}(kT_s) \end{bmatrix} \tag{12}$$

$$\boldsymbol{z}_{k+1} = \begin{bmatrix} \boldsymbol{A}(\tau') & \boldsymbol{B}(\tau') & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \boldsymbol{z}_k \tag{13}$$

We are interested in understanding the effect that a variable τ' will have on the control system. Suppose that τ is drawn from some probability distribution such that $\tau_{\min} \leq \tau \leq \tau_{\max}$. Let $\tau_{avg} = \frac{\tau_{\max} + \tau_{\min}}{2}$. We decompose the time delay as:

$$\tau = \tau_{avg} + \frac{\tau_{\min} - \tau_{\max}}{2}\delta \tag{14}$$

where $|\delta| \leq 1$. This allows for the structural decomposition of the matrices $\boldsymbol{B}_+^D(\tau)$ and $\boldsymbol{B}_-^D(\tau)$ into a constant part and an uncertain part:

$$\boldsymbol{B}_-^D(\tau) = \int_0^{T_s - \tau_{avg}} \exp\left(\boldsymbol{A}^C(r)\right) \boldsymbol{B}^C dr + \int_{T_s - \tau_{avg}}^{T_s - \tau} \exp\left(\boldsymbol{A}^C(r)\right) \boldsymbol{B}^C dr \triangleq \boldsymbol{B}_-^D(\tau) + \Delta\boldsymbol{B}_-^D(\tau) \tag{15}$$

$$\boldsymbol{B}_+^D(\tau) = \int_{T_s - \tau}^{T_s - \tau_{avg}} \exp\left(\boldsymbol{A}^C(r)\right) \boldsymbol{B}^C dr + \int_{T_s - \tau_{avg}}^{T_s} \exp\left(\boldsymbol{A}^C(r)\right) \boldsymbol{B}^C dr \triangleq \boldsymbol{B}_+^D(\tau_{avg}) + \Delta\boldsymbol{B}_+^D(\tau) \tag{16}$$

resulting in the polytopic uncertain model:

$$\boldsymbol{z}_{k+1} = \left(\boldsymbol{A}(\tau_{avg}) + \begin{bmatrix} \boldsymbol{0} & \Delta\boldsymbol{B}_+^D(\tau) \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}\right) \boldsymbol{z}_k + \left(\boldsymbol{B}(\tau_{avg}) + \begin{bmatrix} \Delta\boldsymbol{B}_-^D(\tau) \\ \boldsymbol{0} \end{bmatrix}\right) \boldsymbol{u}_k \tag{17}$$

$$\boldsymbol{z}_{k+1} = \left(\boldsymbol{A}(\tau_{avg}) + \Delta\boldsymbol{A}(\tau)\right) \boldsymbol{z}_k + \left(\boldsymbol{B}(\tau_{avg}) + \Delta\boldsymbol{B}(\tau)\right) \boldsymbol{u}_k \tag{18}$$

assuming $\tau \leq T_S$. The case where $\tau > T_S$ is straightforward to handle.

3.4.3 Efficient Optimal Control of Time Delayed System

We consider the optimal control of the system in (18) subject to the discrete LQR cost with constrained inputs. That is, we consider the optimization problem:

$$\begin{aligned} J_N(\boldsymbol{z}_i)^* = \min_{\mathcal{U}} \ & \boldsymbol{z}_{i+N}^T \boldsymbol{P} \boldsymbol{z}_{i+1} + \sum_{k=i}^{N+i-1} \boldsymbol{z}_k^T \boldsymbol{Q} \boldsymbol{z}_k + \boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k \\ \text{subject to } & \left(\boldsymbol{A}(\tau_{avg}) + \Delta \boldsymbol{A}(\tau)\right) \boldsymbol{z}_k + \left(\boldsymbol{B}(\tau_{avg}) + \Delta \boldsymbol{B}(\tau)\right) \boldsymbol{u}_k \\ & u_{\min} \leq \|\boldsymbol{u}\|_\infty \leq u_{\max} \end{aligned} \tag{19}$$

where $\mathcal{U} = \{\boldsymbol{u}_k\}_{k=i}^{N-1}$. This problem is a constrained, finite time optimal control (CFTOC) problem with polytopically uncertain system identification. For a brief review of the control of discrete, constrained finite time optimal control (CFTOC) the reader is referred to Appendix 1.2.2.

To solve (19), we must obtain norm bounds on $\Delta \boldsymbol{A}(\tau)$ and $\Delta \boldsymbol{B}(\tau)$ [26] so that the machinery from [28] can be applied. Tight norm bounds can be found by solving two linear matrix inequalities (LMI) to obtain the solutions to the generalized eigenvalue problems [29]

$$\begin{aligned} & \min \ t && \min \ s && (20) \\ \text{subject to } & \boldsymbol{Q}_1 \succ 0 && \text{subject to } \boldsymbol{Q}_2 \succ 0 && (21) \\ & \boldsymbol{Q}_1 = \boldsymbol{Q}_1^T \prec t\boldsymbol{I} && \boldsymbol{Q}_2 = \boldsymbol{Q}^T \preceq s\boldsymbol{I} && (22) \\ & \Delta \boldsymbol{A}(\tau_{\max})^T \Delta \boldsymbol{A}(\tau_{\max}) \prec \boldsymbol{Q}_1 && \Delta \boldsymbol{B}(\tau_{\max})^T \Delta \boldsymbol{B}(\tau_{\max}) \prec \boldsymbol{Q}_2 && (23) \\ & \Delta \boldsymbol{A}(\tau_{\min})^T \Delta \boldsymbol{A}(\tau_{\min}) \prec \boldsymbol{Q}_1 && \Delta \boldsymbol{B}(\tau_{\min})^T \Delta \boldsymbol{B}(\tau_{\in}) \prec \boldsymbol{Q}_2 && (24) \end{aligned}$$

The solution of this SDP allows for a conservative, constant approximation of the system dynamics with polytopic uncertainty. After solving the SDP to obtain the matrices Q_1 and Q_2, we solve the program (25).

$$\begin{aligned} J_N(\boldsymbol{z}_i)^* = \min_{\mathcal{U}} \ & \boldsymbol{z}_{i+N}^T \boldsymbol{P} \boldsymbol{z}_{i+1} + \sum_{k=i}^{N+i-1} \boldsymbol{z}_k^T \boldsymbol{Q} \boldsymbol{z}_k + \boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k \\ \text{subject to } & \left(\boldsymbol{A}(\tau_{avg}) + \boldsymbol{Q}_1\right) \boldsymbol{z}_k + \left(\boldsymbol{B}(\tau_{avg}) + \boldsymbol{Q}_2\right) \boldsymbol{u}_k \\ & u_{\min} \leq \|\boldsymbol{u}\|_\infty \leq u_{\max} \end{aligned} \tag{25}$$

As this problem is time invariant, it is well known to admit parametric solutions allowing for efficient $\mathcal{O}(\log_2(K))$ online evaluation of the control law via binary search for some constant K once the controller is synthesized offline by solving the parametric program (see [30], [21], [29], or Appendix 1.2.2).

3.4.4 Controller Design

In this section, we synthesize the previous sections into a complete description of the control of a platoon of vehicles subject to bounded network delays which can be implemented on a digital (and therefore discrete) computer. We begin by constructing the state space model corresponding to the continuous time system in

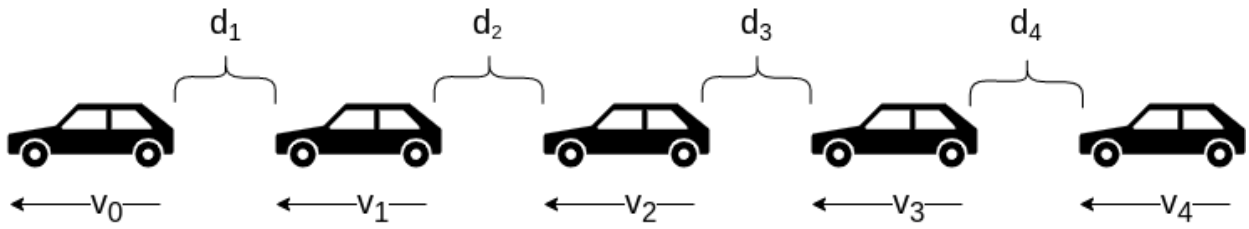

Figure 3: Five vehicle platoon diagram

(1). Consider a set of $n+1$ vehicles with indices $\{0, 1, \ldots, n\}$. We construct the first order model:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ldots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v^{(0)} \\ d^{(1)} \\ v^{(1)} \\ d^{(2)} \\ \vdots \\ v^{(n-1)} \\ d^{(n)} \\ v^{(n)} \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix} \begin{bmatrix} u^{(0)} \\ u^{(1)} \\ u^{(2)} \\ \vdots \\ u^{(n)} \end{bmatrix} \tag{26}$$

This matrix has a clear triangular block structure enabling the local decomposition of the i^{th} subsystem as:

$$\dot{\boldsymbol{x}}^{(i)} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v^{(i-1)}(t) \\ d^{(i)}(t) \\ v^{(i)}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u^{(i)}(t-\tau) = \boldsymbol{A}^C \boldsymbol{x}(t) + \boldsymbol{B}^C u^{(i)}(t-\tau) \tag{27}$$

The discretization of this continuous time system is contingent on an understanding of the physical system's sampling rate T_s and network delay τ which we defer until the next semester. For now we proceed assuming these can be measured or selected appropriately.

We now seek to design a cost which will encourage efficient platooning. In typical platooning systems, the desired reference distance $d^{(i,ref)}$ is proportional to the current velocity [7]. That is:

$$d^{(i,ref)} = \beta^{(i)} v^{(i)} \tag{28}$$

for $\beta^{(i)}$ constant. Our performance criterion is the minimization of the squared deviations of the velocities in the subsystem $\left(v^{(i-1)} - v^{(i)}\right)^2$ as well as the squared deviation from the reference distance $\left(d^{(i)} - \beta^{(i)} v^{(i)}\right)^2$. Furthermore, we wish to minimize our actuation energy $u^{(i)}$. Consider that we construct our augmented, discrete time, state vector $\boldsymbol{z}_k$ according to the formulation in section 3.4.2, accounting for a potentially long network delay relative to the sampling period. This results in a state vector

$$\boldsymbol{z}_k^{(i)} = \begin{bmatrix} v^{(i-1)}(kT_s) \\ d^{(i)}(kT_s) \\ v^{(i)}(kT_s) \\ u^{(i)}((k-d)T_s \\ \vdots \\ u^{(i)}(kT_S) \end{bmatrix} \tag{29}$$

The cost structure of the discretized plant over an N step time horizon is given by

$$
\begin{aligned}
J_N(u^{(i)}) = \min_{\mathcal{U}} \sum_{k=0}^{N} \left(\boldsymbol{z}_k^{(i)}\right)^T
&\left[\begin{array}{ccc|cccc}
w_{\Delta v}^{(i)} & 0 & -w_{\Delta v}^{(i)} & 0 & 0 & \dots & 0 \\
0 & w_{\beta^{(i)}}^{(i)} & -\beta w_{\beta^{(i)}}^{(i)} & 0 & 0 & \dots & 0 \\
-w_{\Delta v}^{(i)} & -\beta w_{\beta^{(i)}}^{(i)} & \left(\beta^{(i)}\right)^2 w_{\beta^{(i)}}^{(i)} + w_{\Delta_v}^{(i)} & 0 & 0 & \dots & 0 \\
\hline
0 & 0 & 0 & R^{(i)} & 0 & \dots & 0 \\
0 & 0 & 0 & 0 & R^{(i)} & \dots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & 0 & \dots & R^{(i)}
\end{array}\right] \boldsymbol{z}_k^{(i)} \\
&\text{subject to } u_{\min}^{(i)} \leq u_k^{(i)} \leq u_{\max}^{(i)}, \text{ and the discretized dynamics} \\
= \min_{\mathcal{U}} \sum_{k=0}^{N} \left(\boldsymbol{z}_k^{(i)}\right)^T
&\left[\begin{array}{c|cccc}
Q & 0 & 0 \dots & 0 & \\
\hline
0 & R^{(i)} & 0 & \dots & 0 \\
0 & 0 & R^{(i)} & \dots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \dots & R^{(i)}
\end{array}\right] \boldsymbol{z}_k^{(i)} \\
&\text{subject to } u_{\min}^{(i)} \leq u_k^{(i)} \leq u_{\max}^{(i)}, \text{ and the discretized dynamics}
\end{aligned}
\tag{30}
$$

The weights of $w_{\Delta v}^{(i)}$, $w_{\beta^{(i)}}^{(i)}$, and $R^{(i)}$ are chosen as relative penalties on velocity deviations, time-space headway deviations, and input actuation cost. Higher w and lower R will tend to lead to more aggressive system responses as input actuation will be encouraged in order to maintain the steady state of the platoon. We anticipate that we will tune our controllers in such a manner.

Remark: It is worth noting that the construction of the multiparametric solution described in section 3.4.3 is only feasible when there is low input dimension and relatively short time horizon. As the dimension of the state and the time horizon increase, the number of partitions on the piecewise affince controller computed via the multiparametric program grows exponentially [28]. The state of our proposed problem is small enough that we will not suffer from this exponential growth and thus will be able to compute and store the entire controller offline. This enables the efficient online evaluation of the control law via search rather than online optimization.

3.4.5 Analysis of Stability: Specifications of a Control System

Typically, control systems are concerned with a notion of stability known as global asymptotic stabilitiy (GAS). This notion of stability concerns controlling a dynamic system such as (6) such that $\lim_{t\to\infty} \boldsymbol{x}(t) = 0$ [30]. The authors of [24] demonstrate that the construction of section 3.4.1 results in a globally asymptotically stable controller if each subsystem achieves an asymptotically stable solution locally. A similar analysis is immediate for the controller described in section 3.4.4 if the bounds on the delay's effect is not significant. We do not derive this as GAS is not the primary form of stability for which platooning systems are concerned.

String stability was introduced by Swaroop in [10], which intuitively describes the uniform boundedness of all the states in the system for all time. This form of stability is particularly important in platoons as failure to achieve string stability results in the exponential growth of errors in the platooning system which in turns leads to system failure such as disengagement from the platoon or crashes [31, 6].

The definition of string stability put forward by Swaroop can be difficult to analyze, particularly when closed form solutions of the control synthesis are not available. In [24], the authors propose to use a surrogate

measure of string stability based on the transfer function with the upstream vehicle's velocity as the input, and the downstream vehicle as the output. This transfer function relationship is given by:

$$G(s) = \frac{V_i(s)}{V_{i-1}(s)} = \frac{-p\frac{u_k}{v_{i,k}}s - p\left(\frac{u_k}{d_{i,k}}\right)^2}{s^2 + p\frac{u_k}{v_{i-1,k}}s - p\left(\frac{u_k}{d_{i,k}}\right)^2} \tag{31}$$

where p in the above is the parameter dictating converting engine torque to engine force.

By maintaining that $\max_s \left|G(s)\right| \leq 1-\gamma$, we ensure string stability as defined in [24] provided $\gamma \in [-1, 0]$. A γ closer to -1 implies that there is no output response in response to sinusoidal disturbances of $V_i(s)$, while γ close to 0 implies a larger, but still stable response.

This results in the final form of the Car-Tooning controller:

$$\begin{aligned}
\min_{u_k} \quad & x_N^T P x_N + \sum_{k=0}^{N-1} \left(x_k^T Q x_k + u_k^T R u_k\right) \\
\text{subject to} \quad & x_{k+1} = A_d(\tau_a)x_k + D_1 x_k + B_d(\tau_a)u_k + D_2 u_k \\
& \max_s \left| \frac{-p\frac{u_k}{v_{i,k}}s - p\left(\frac{u_k}{d_{i,k}}\right)^2}{s^2 + p\frac{u_k}{v_{i-1,k}}s - p\left(\frac{u_k}{d_{i,k}}\right)^2} \right| \leq 1 + \gamma \\
& -1 \leq \gamma \leq 0
\end{aligned} \tag{32}$$

3.5 Autonomous Platooning Testing Suite

Typical autonomous platooning simulation software focuses on studying the macro behavior that various platooning algorithms can have on large-scale traffic interactions [32]. Conversely, high-fidelity physics simulators such as Gazebo [33] typically do not have realistic network simulators built in. Furthermore, simulators often fail to capture the requisite randomness of network delays typical in outdoor environments. It was therefore determined that the best test bed would be implementation onto a small platoon of rudimentary autonomous vehicles. This section serves to outline the efforts made in the development of such vehicles and the ultimate results. We begin with a description of the hardware platform and continue with a description of the developed software stack as outlined in figure 4. All software can be found in the repository at https://github.com/AlexandreAmice/SeniorDesign

3.5.1 Hardware Description

In this section we discuss the hardware chosen to test our control scheme. This hardware was selected on the basis of low cost and a belief that the low performance would enable Car-Tooning to show dramatic improvements over less sophisticated control techniques. Our physical test-bed was five vehicles repurposed from the University of Pennsylvania ESE 421 course on the control of autonomous vehicles. The vehicles contain two compute units.

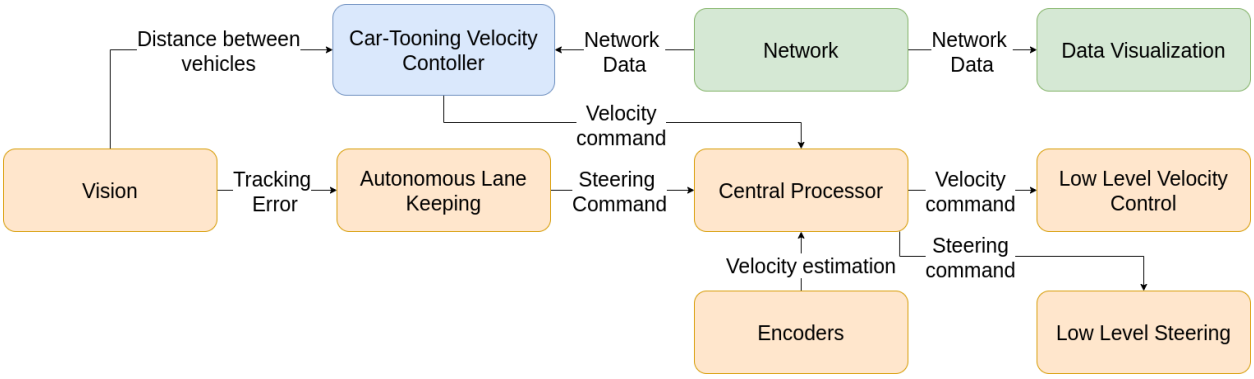

Figure 4: A block diagram of the developed autonomy software stack. The orange blocks represent the core autonomy meant to enable autonomous lane following. The green blocks represent data communication blocks. Car-Tooning in blue serves the role of the velocity motion planner.

The first is a Raspberry Pi 3 which is a complete, general purpose computer which runs a varient of the Debian operating system which supports limited GPIO functionality. The Raspberry Pi comes with built in Wifi and Bluetooth capabilities. The general purpose nature of the Raspberry Pi enables the choice of any high-level programming language. For this project we chose Python for its high-prototyping speed, strong linear algebra libraries, and established use for computer vision.

The second is compute unit is the Arduino Mega 2560 which is a dedicated microcontroller supporting a wide variety of plug in sensors and real-time operation. This unit only supports the Arduino programming language which is a derivative of the C/C++ family of programming languages. The two compute units communicate via a dedicated I^2C bus where the Raspberry Pi acting as the master and the Arduino the slave. We note that the two compute units do not share a common clock and so are completely asynchronous from each other.

The Arduino Mega interfaces with the majority of the onboard hardware. The front wheels are jointly connect to a single TowerPro SG-5010 servo which is used to guide the wheels into position for steering. These wheels are not driven. The back wheels are each driven by their own DC geared motor. The intent was to install encoders onto the rear wheels to enable velocity estimation, but logistical issues with the department impaired this venture. Onboard state-estimation is achieved via three attached Adafruit HC-SR04 ultrasonic sonar distance sensors and Adafruit FXOS8700+FXAS21002 nine degree of freedom, inertial measurement unit (IMU) both of which interface with the Arduino Mega as well as a Raspberry Pi Camera Module V2 which connects to the Raspberry Pi and enables the use of visual information.

The system was powered via a 12 volt battery simultaneously connected to the Arduino via its the DC power jack and to the Raspberry Pi via the micro-USB port.

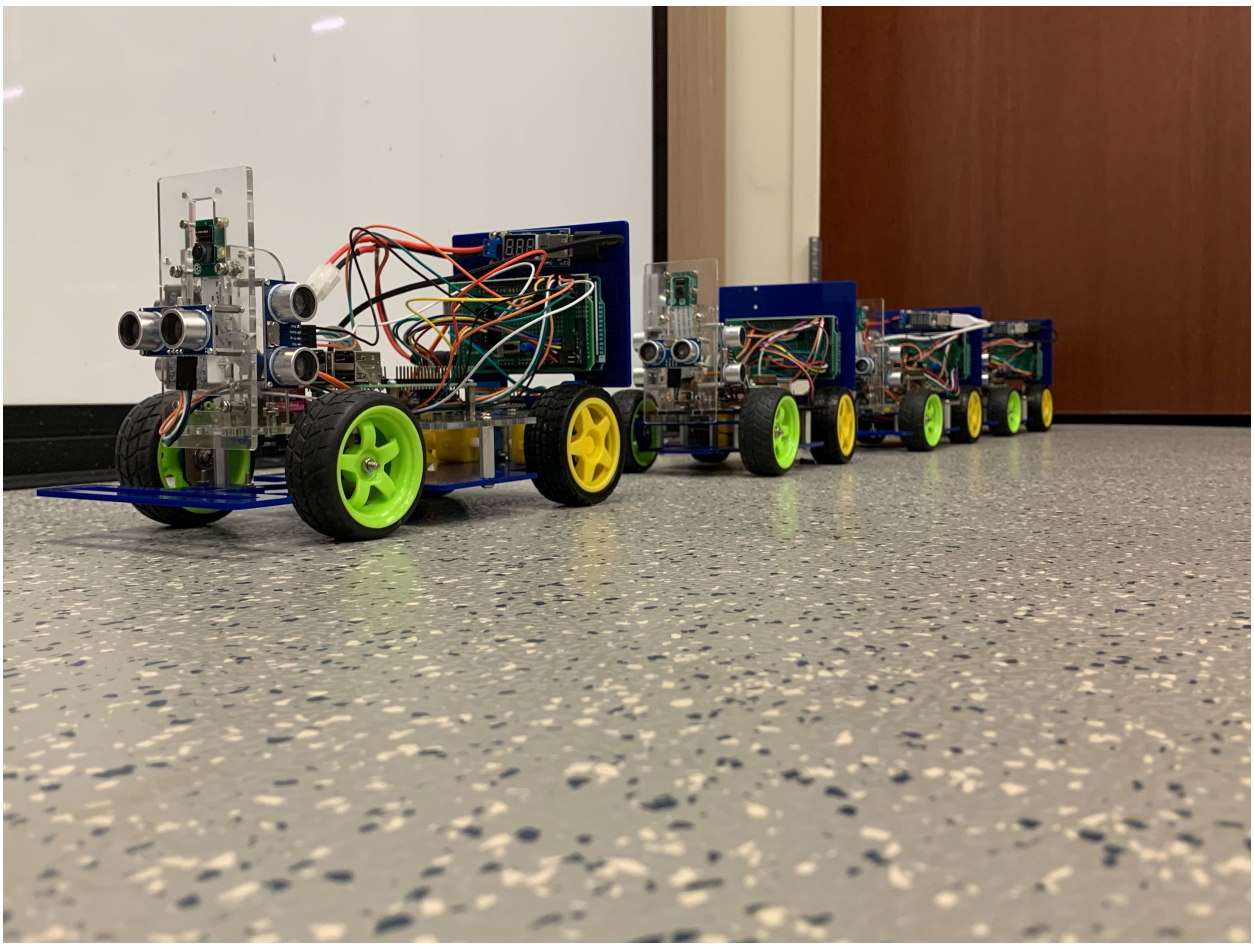

Figure 5: Fully assembled platooning test bed

3.5.2 Software Architecture

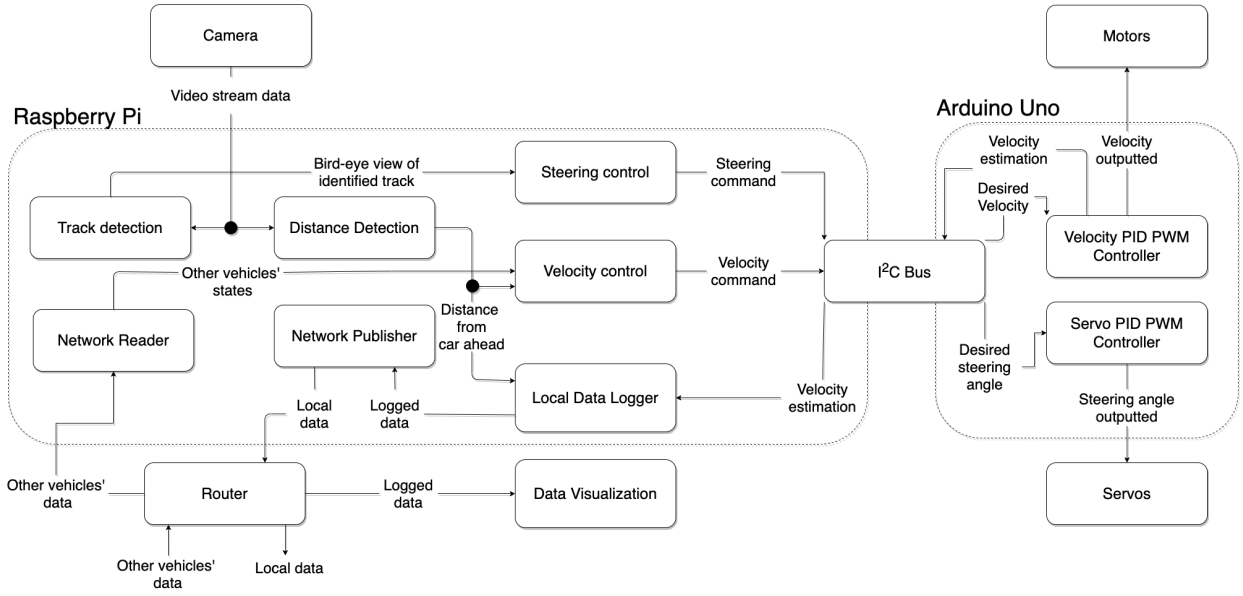

Figure 6: Internal Architecture of the Autonomy Stack

This section serves to give an overview of the software architecture developed to enable rudimentary autonomous driving on the above vehicles. We begin by describing the responsibility of each compute unit, the data which is sent between the two, and finally describe the internal flow of information within each compute unit.

The Raspberry Pi contains a 4 core ARM Cortex-A53 CPU with 1GB of DDR RAM. The ability to run the Debian operating system enables the use of standard higher level program languages such as Python. However, as Debian is does not meet the requirements a real-time operating system (RTOS), it is ill-suited for real time control of most hardware peripheries. By contrast, the Arduino Mega is a microcontroller with only 8KB of SRAM which only runs the Arduino programming language. Few libraries exist for this language beyond those created specifically to interface with hardware plugins.

These different strengths lead to a natural division of labor of software components in the two computes. In the Raspberry Pi, we implement all higher level planning such as determining the appropriate steering and velocity set points and network interfacing. The Arduino is used to control the motors and and steering servo to meet these desired set points.

We begin by describing the internal architecture of the Raspberry Pi. The Raspberry Pi has five main functions: camera state estimation, velocity and steering set-point computation, data logging, network communication, and I^2C bus management. These functions need to run simultaneously at very high frequency with large amounts of exchanged data as seen in figure 6. Due to the high volume of exchanged information, it was desirable that all tasks share a common memory space leading us to use a thread-driven concurrency model rather than a process driven one. Unfortunately, due to Python's Global Interpretter Lock (GIL) concurrency model, only one thread can be executed at a time in Python, preventing the leveraging of all cores on the CPU. In order to prevent resource hogging by a single thread, threads were manually scheduled

such that as many threads as possible would complete within a 100ms time window. Thus, the use of threads trades reduced latency introduced by frequent information sharing through inter-process communication and increased latency due to the Python GIL.

The five aforementioned tasks were divided into five modules. In the first module, the camera was run in continuous video mode. As frames entered memory, they were processed in order to detect the distance to the vehicle in front as well as deviation from a desired track. This process is described in further detail in section 3.5.3. The next module handled reading and writing to the network. As the vehicle state changed, this triggered periodic broadcasts to the network of the most current vehicle state. This module also accepted networks messages addressed to the current vehicle's specific IP address and forwarded the necessary information to the control modules. This is described in further detail in section 3.5.5.

The control module was responsible for determining the set-point of the desired velocity and steering angle based on the available camera data, information from other vehicles collected via the network module, and the velocity information from the Arduino. A variety of simple controllers were implemented in this framework including the ability to control the vehicle via keyboard commands, a controller which exactly mimicked the state of another vehicle, and a distance-based velocity feedback controller.

A data logging module was also implemented in order to keep track of key variables. These included desired steering angle and velocity over time, measured distance from the vehicle in front, and offset from the desired trajectory. This local data was saved to disk as well as transmitted over a network to a central aggregator capable of logging all the data from all vehicles simultaneously.

Finally, as the I^2C bus spanned both devices each device needed its own interface to the hardware. As the master, the Raspberry Pi controlled all read and write requests to which the Arduino responded. As the I^2C bus is only capable of communicating bytes directly with no native encoding, the Raspberry Pi implemented an encoding scheme to control the access of information. All data was packaged into lists of 8 bytes. The first byte served as a command byte, instructing the Arduino to either accept the following data as a specific variable, trigger a specific method on the Arduino, or have the Arduino respond with requested information. The remaining 7 bytes were used to transmit information that the Arduino should store if desired.

As the Raspberry Pi did not have direct access to the Arduino's memory registers, virtual registers were implemented on the Arduino's I^2C bus software module. Two C-style array buffer were maintained by the Arduino. The first stored information that the Raspberry Pi had sent to the Arduino such as the desired velocity and steering angle. The second stored information which the Raspberry Pi could access by requesting it from the Arduino. On the receipt of a message from the Pi, the Arduino parsed the first byte to determine whether to write to its memory buffers or prepare itself to send information to the Raspberry Pi. It then proceeded to either read the remaining 7 bytes or respond with the desired information.

As the Arduino is not a general purpose computer, it does not support multi-threading and multi-processing. Concurrent behavior is achieved via protothreads which are event-driven, stackless threads which enables the concurrent execution of code while listening for triggers or interrupts from peripheries. As such, the architecture of the Arduino code is not truly encapsulated into modules beyond the I^2C bus and the main code. Two methods drive the Arduino's behavior which are run in sequentially fashion in a 10ms loop. The steering control is modulated by a PID loop where the desired steering angle is read from the register where the Raspberry Pi writes to. By using the gyroscopic rates from the IMU, the actual turning rate of the vehicle can be measured and the servo angle command can be adjusted in order to achieve the desired turning behavior. The speed of the motors was controlled via PWM rather than direct control of their velocity. Thus, measurements were taken enabling the fitting of model converting velocity to desired PWM inputs. The velocity was then controlled in an open-loop fashion as no encoders were ever placed

onto the cars. If this had occurred logistically, we would have implemented a PID feedback on the motors to ensure adherence to the desired velocity.

3.5.3 Vision-Based State Estimation

Vision based state estimation was used for two tasks: inter-vehicle distance estimation and track localization. Both vision tasks were run sequentially in the same thread as each acquired image required similar processing and the overhead introduced by creating an additional thread would not have led to a decrease in overall processing time.

Incorporating the vehicles into a motion capture system would have introduced a significant labor overhead as the team was not familiar with such systems. Moreover, accurate distance estimation is an integral part of realistic autonomy and therefore testing our system with a true estimated distance rather than receiving the distance measurements over the network within micrometer accuracy provided a more realistic test bed.

Accurate inter-vehicle distance estimation was achieved through the use of April Tags. April tags are a visual fiducial system developed by the APRIL group at the University of Michigian. The tags are easy to detect and process with a camera. After appropriate camera calibration, Python libraries exist for giving high accuracy pose estimation of the tag in the camera-relative frame.

The method by which pinhole cameras project three dimensional information onto the two dimensional image plane introduce two major types of distortion: radial and tangential. Radial distortion causes straight lines to appear curved with the effect becoming more pronounced as we approach the edges. Tangential distortion is introduced as the plane of the image lens is typically not perfectly aligned with the plane of the imaging plane. This causes parts of the image to appear closer than they are. The two effects cannot be corrected simultaneously. These two distortion effects are parametrized by five distortion coefficients:

$$\begin{pmatrix} k_1 & k_2 & k_3 & p_1 & p_2 \end{pmatrix}$$

Radial distortion can be corrected via the transformation

$$x_corrected = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_corrected = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Tangential distortion can be corrected via:

$$x_corrected = x + 2p_1 xy + p_2(r^2 + 2x^2)$$
$$y_corrected = y + p_1(r^2 + 2y^2) + 2p_2 xy$$

where $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$ and (x_c, y_c) is the center pixel.

Additionally,4 intrinsic camera parameters need to estimated. The first two are the focal lengths f_x and f_y which measure how strongly the camera pinhole converges light onto the imaging plane in the lateral and longitudinal direction. The second is the optical center (c_x, c_y) which is the center of the projection transformation or the stationary point of the $3D$ to $2D$ projection.

By following the tutorial at Camera Calibration, we were able to automate this process using the OpenCV library. By taking 10 pictures of a printed chessboard pattern with 7 interior corners and tile size 3cm from a distance of 1m away, we obtained the above 9 coefficients. The precision of these coefficients was sufficient to obtain 1cm accuracy on the distance of our AprilTag estimation.

The second vision task was relative offset from a desired path. Our objective was to have the platoon drive in a circle as a closed loop trajectory would enable the experiment to run for as long as we desired.

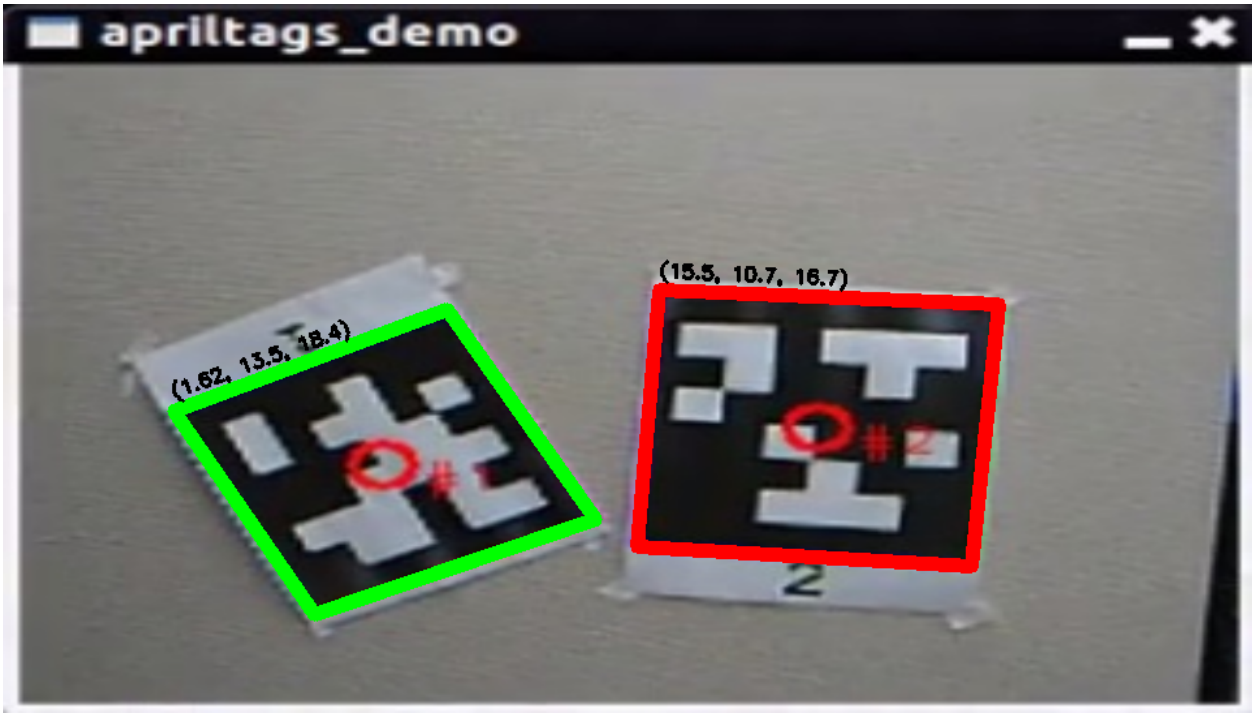

Figure 7: Accurate April Tag detection with 3D pose in black text above the picture. The green outline indicates detection of the desired tag while the red denotes tag detection, but not the tag we are seeking

To this end, we created a yellow-taped track on a red floor to create a high-contrast desired path as seen in the top at figure 8. By using color thresholding in the HSV [34] and LAB [35] colorspaces, very high fidelity track detection could be achieved as shown in figure 8 where the detected track can be seen in the lower image highlighted in green.

After detection of the track was achieved, a plausible offset and relative orientation to the track was computed. This was visualized in figure 9 where the objective is to align the red line to the green i.e. the track should appear in the center of our field of view and straight. With this detected vector in hand, we collected two points on the vector (x_1, y_1) and (x_2, y_2) and translate them to world coordinates. The angle the vector creates on the screen was translated to a relative heading offset via the formula:

$$\psi_{\text{rel}} = \arctan \frac{y_2 - y_1}{x_2 - x_1} - \pi/2 \tag{33}$$

The offset from the nearest point on the track is computed via the formula:

$$d = \frac{|y_1 - x_1|}{m^2 + 1}$$

where $m^2 = \frac{y_2 - y_1}{x_2 - x_1}$

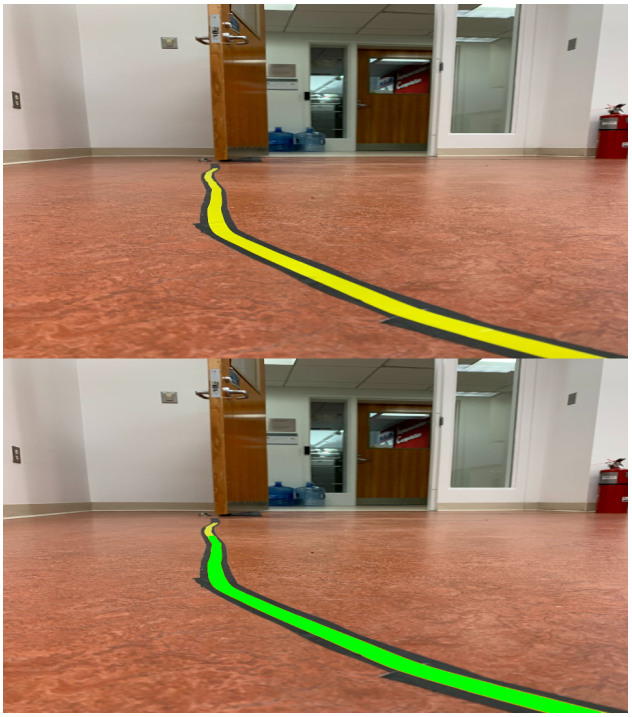

Figure 8: The top photo is the original unaltered track. In the lower photo, the detected track is highlighted in green. Note that the picture is cropped during detection to avoid artifacts in the horizon, hence the failure to detect the later track.

The translation of a point in pixel coordinates (u, v) to world coordinates (x, y) is achieved via [36]:

$$x = \frac{f_v h}{v - c_v} \tag{34}$$

$$y = \frac{y_w(u - c_u)}{f_u} \tag{35}$$

Where f_v and f_u are the v and u focal lengths and c_u and c_v are the u and v focal centers.

3.5.4 Steering Feedback Control

Once the relative offset and heading from the desired track were computed, a simple feedback controller was implemented in order to steer our vehicle to the desired track shown in figure 10. A gain of $K = 0.3$cm/rad was chosen, with the vehicle length being 30cm. This gain was chosen based on its performance on the velocity range of $[10, 40]$cm/s which is the anticipated range of velocities the vehicle would be tested on. In figure **??** you can see a characteristic plot of the vehicle's response when initially place 10cm off center from the track and with 0 radians of heading error. As you can see, the vehicle rapidly recovers to 0 track offset and heading error.

3.5.5 Network Communication

The above portions of the autonomy stack were used to enable the autonomous navigation of a single vehicle around a closed track. To enable platooning behavior, a wireless communication infrastructure needed to

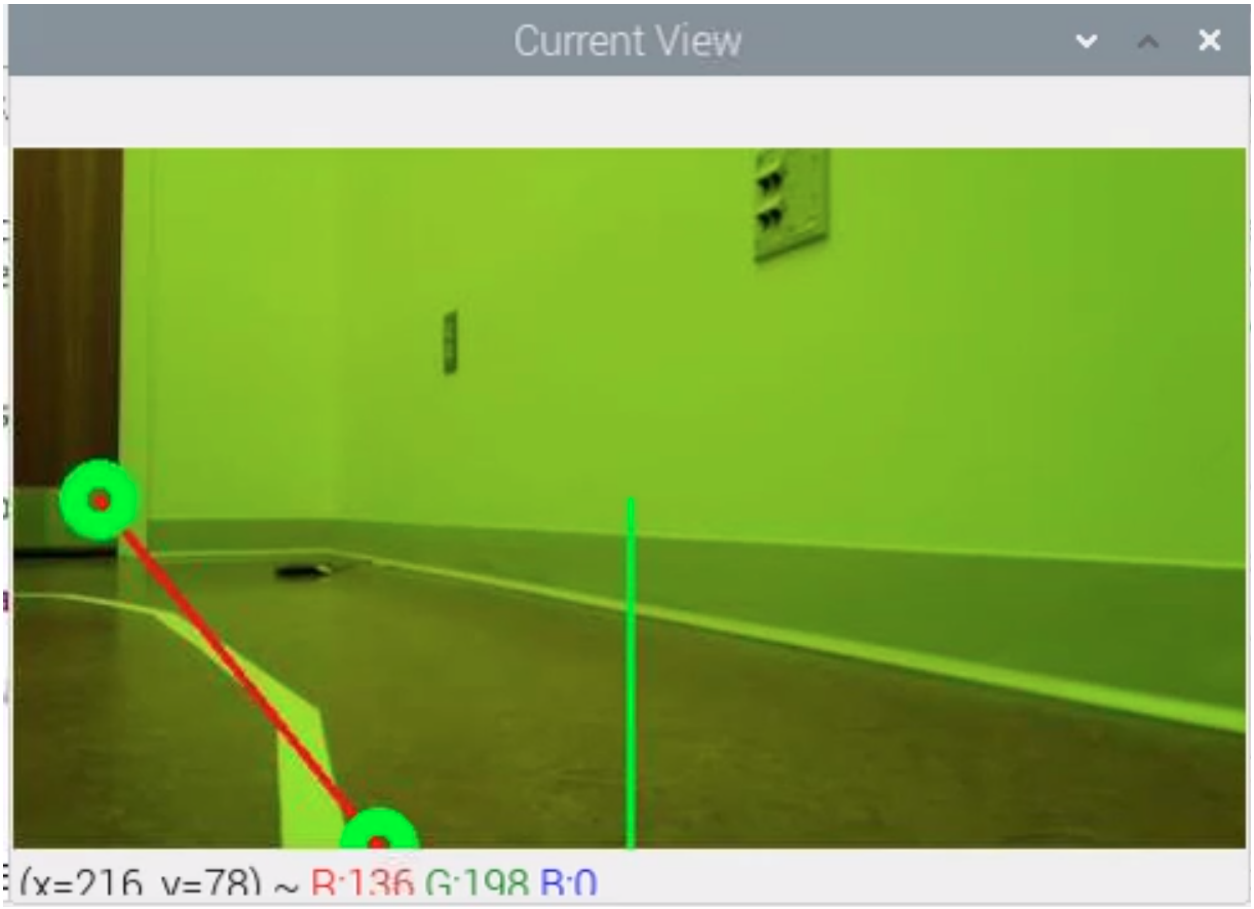

Figure 9: Relative heading and offset of track in camera coordinates

be implemented. In the network module, two separate threads were maintained: a network reader and a network writer. All communication was implemented over an available IEEE 802.11 Wifi network. The IPv4 standard was used as the network layer protocol. The UDP transport layer protocol was used as it introduces minimal overhead and focuses on fast rather than reliable transmission of the data stream.

The network reader thread was set to read information addressed to it's IP with a buffer size of 1024 bytes. Information was read as JSON files when available or otherwise as raw strings. After reading in the information, the latest information received by the network reader was appended to a shared memory address where it could be accessed by the controller and other relevant modules.

The network writer periodically broadcast information to all IP addresses to which was it was configured to write to. By design, it published JSON packets containing the current vehicle's ID number, IP address, velocity, distance from the vehicle ahead, and steering angle. This shared information would give the Car-Tooning controller all information it needed to output decisions.

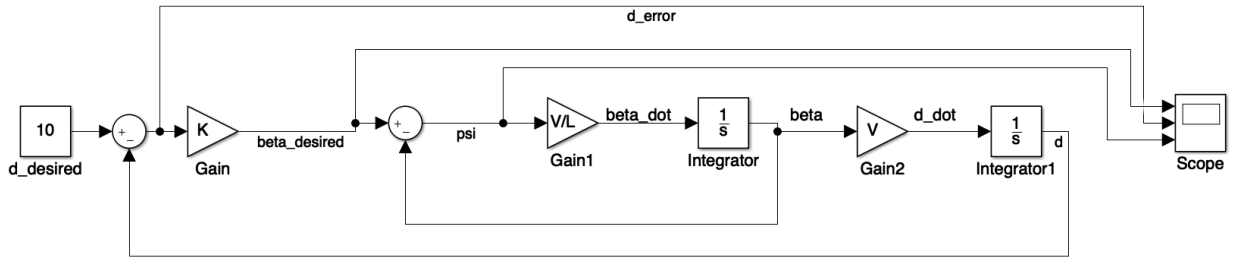

Figure 10: Block diagram of the steering controller

3.5.6 Data Visualization

The last piece of the autonomy stack consisted of a simple data visualizer that could output any property of the car onto a graph that was updated in real time. This was done by creating a data logger which periodically logged the outputs of the controller and packaged them into a JSON that could then be read by an animated plotter that was being updated with each new data point. This data could also be saved onto disk for analysis, although this was never used due to the coronavirus outbreak cutting our progress short.

3.6 Results

In this section we discuss the results of the technical developments of both the Car-Tooning controller and the autonomy stack. In section 3.4, we illustrate a method of designing controllers robust against randomized communication time delays for the purpose of platooning. A distributed controller was developed where each vehicle generates locally optimal acceleration commands based on the available information. By constraining this program to output commands which maintains the pairwise input-output stability of each vehicle, our controller is provably string stable. As the scheme optimizes against the worst case time delay of the network (within reason) we ensure that string stability is maintained for all possible network states.

In section 3.5 we describe the development of a rudimentary autonomous vehicle onto which we intended to test Car-Tooning. We outline the hardware specifications, software developments, and theoretical background necessary to enable the vehicle to navigate around a circular track demarcated by yellow tape on a red floor. Unfortunately, it was determined after testing that the hardware was of far too poor quality to accomplish even this simply task. The printed frame of the vehicle resulted in very poor granularity in the steering control making it quite difficult to steer the vehicle. Furthermore, the power draw from the motors of the vehicle was enough to reduce the voltage to both compute stacks. This severely degraded the camera's frame rate, making it very difficult to plan for curvature of the track and reduced the computational speed of the Raspberry Pi. The lack of power to the Raspberry Pi resulted in it automatically shutting off Wifi communication to preserve power for what it deemed to be critical resources and reduced the rate at which the Raspberry Pi could update the Arduino on the new desired steering commands.

These issues were epitomized by our attempts to control the vehicle manually using keyboard commands sent over the Wifi network. The lack of responsiveness of the onboard vehicle prohibited even manual, human control from guiding the vehicle along the desired track. This obstacle suggested the task that a model-based solution to the track following problem was infeasible for the given platform.

The net result was an inability to continue with the intended test bed. The decision to transition to a simulation environment distributed across multiple computers was made right before Spring break. As the team was prohibited from returning to campus due to the coronavirus outbreak, the experiment was discontinued due to lack of computing resources and access to a shared network.

3.7 Technical Conclusion

In this section we described the technical developments required to understand the Car-Tooning controller. The solution is shown to be globally asymptotically stable and locally optimal. We further outlined a realistic autonomy stack developed in order to enable the testing of the Car-Tooning controller. Both the hardware and software infrastructures were described as well as the network communication capabilities. Finally, the results we were able to gather were discussed. As the hardware performed much worse than anticipated, with large delays throughout the system, we were unable to successful manufacture an autonomous platoon in the given time frame. We had hoped to continue testing in a simulation, yet due to the coronavirus outbreak, we were unable to conclude testing the Car-Tooning controller as we lost access to a shared network and testing hardware.

IV Self Learning

4.1 Tools and Theory

Due to the advanced theoretical nature of our project, a significant amount of effort was spent understanding the state of the art. Controller design and model predictive control was the primary subject matter both of which Alexandre has experience through course work and research. Sophia's extensive controls coursework and Sergio's mathematics background have been important in digesting the many research papers involved in the development of the project. The reference [7] has been particularly helpful for all team members in acquiring further knowledge on the mathematics behind networked control.

Additionally, our group attempted to learn and understand the state of the art traffic and platoon simulators to assist us in understanding the challenges in platooning. We chose to use Plexe [32], an open-source platooning simulator which incorporates the state of the art traffic simulator SUMO (Simulation of Urban MObility) [37] and the Veins network simulator [38]. As this simulator is built on C++, many team members needed to learn the language. As Matteo supported the team's simulator development efforts, he also needed to improve his knowledge of Unix operating systems and C++ build systems. Furthermore, as Plexe integrates multiple programming languages and paradigms, Matteo had to learn how to administer a large and complex build system.

Once the controller was designed, we shifted to hardware and to the development of an autonomy stack to realistically test our design. We decided to use the ESE 421 cars because Alexandre and Sophia have both taken the course and were familiar with them. The development of the autonomy stack was done in Python and designed mainly by Alexandre, with his experience with autonomous vehicles from a summer at Uber particularly helpful. The autonomy stack incorporated computer vision-aided track and vehicle detection, for which we used openCV. Only Alexandre had previous experience with it but all team members got familiar with it, with Matteo calibrating the vehicle detection while Sophia and Sergio worked on the track detection side.

4.2 Classes

The main classes that have helped with this project have been the controls classes that Sophia and Alexandre have taken at Penn: ESE 421, ESE 500, ESE 505, ESE 512, ESE 619 and ESE 680 (learning for controls). These classes have directly helped with the design of the controller and ESE 421 helped greatly with the hardware, while other classes have helped in a more indirect manner. For example, Matteo's exposure to hardware and embedded systems (ESE 350) helped with the cars, and Sergio and Alexandre's exposure to mathematics (as their second major) have helped in understanding the mathematics of platooning. Sophia has taken linear systems theory and nonlinear dynamics which help in the modeling of a platooning system. In addition she has taken controls (ESE 505) and developed a final project that attempted to have faulty hardware platoon. Finally, Sergio has experience with signal processing and the stability of signals (ESE 531). Alexandre's exposure to computer vision through CIS 680, his Uber internship, and research proved helpful with later vision tasks. Other than these classes it's clear that Alexandre's exposure to research have made him well prepared to understand most of the papers quickly and understand how they fit into the picture of our project.

V Ethical and Professional Responsibilities

Our controller provides a method for the automatic control of vehicles and thus the ethics of our project are fundamentally tied to many of the issues surrounding the the general adoption of autonomous vehicles. Platooning promises to make the adoption of autonomous vehicles highly attractive due to the potential to significantly reduced traffic time. However, autonomous vehicles additionally are expected to be prohibitively expensive to the majority of the population with some estimates putting the cost of an autonomous vehicle at over $100,000 more than a conventional vehicle [39]. The potential for high density platoons runs the risk of creating two classes of citizens on the roads, those with autonomous vehicles capable of high throughput and those relegated to slower lanes of traffic. Furthermore, a poorly implemented platooning algorithm could in fact make traffic worse through large inter-vehicle distances or unstable oscillations contributing to traffic [6].

Conversely, a well implemented platooning algorithm could in fact reduce traffic for all road users. Those capable of platooning would occupy the road surface much more densely, freeing up road surface for other users and enabling higher speeds. It is incumbent on cities to ensure the best method for permitting and deploying platoons on their roads. This project seeks only to empower cities to make this decision.

Another important ethical consideration is the loss of employment due to the advent of driverless vehicles. 3% of Americans operate a vehicle as a living [40] and truck driving is the most common job in 29 states [41]. The ability to operate vehicles autonomously and have these vehicles platoon is likely to lead to reduction in employment in these sectors. As Car-Tooning expects to enable platooning more easily, it is possible we contribute to unemployment in this area. This is a societal issue that must be addressed by regulatory bodies to ensure equitable access to the prosperity brought on by autonomous vehicles.

Finally another issue that our team is working on is the business aspect of our design. We have researched congestion pricing in order to align incentives between drivers and users of these platoon lanes. With this type of pricing model, it would accurately match how much drivers value the time that they save in the platoon lane with the cost of the lane itself.

VI Meetings

We met as a team every Monday from 9am - 1pm. During this time, we worked on the project in K-Lab and later in the third floor of Moore. The reason we had this large block of time to work as a group on the project was because the hardware required two or three people working together. This block was very helpful and allowed us to start our weeks on the right foot. We also meet Monday afternoons with our advisor, Dr. Morari. These meetings were structured to review weekly progress updates and a discussion of upcoming deliverables and next steps. Due to the nature of our project, some of our work could be done independently with constant communication over Slack. Furthermore, since different people had more expertise in different subjects of our project, it wasn't uncommon for two people to meet without the rest of the team there. For example, Matteo was in charge of calibrating the car detection, but this was done with Sergio as well, and at the same time Sophia and Alexandre were working on the filters for the actuation. More frequent meetings were held in advance of a presentation and deadlines, where the whole team would discuss on how to conduct the presentation and then separate the work on the presentation.

Overall coordination for this was done pretty effectively and the communication channel was constantly in use and a critical tool for the project. Sophia and Alexandre both also had occasional meetings with experts in controls such as Professors Pappas and Kothmann as well as networks through meeting with Professor Ribeiro. These were one-off meetings asking for advice rather that consistent meetings.

VII Schedule and Milestones

Our milestones shifted throughout the semester as we found the limitations of hardware to be impassable. The first milestone of the spring was to make sure the cars were in functioning condition so we could start to push our code onto it. This was completed in late January. The second milestone was to have a functioning bare bones autonomy stack where cars could communicate with each other, and this was also finished in late January thanks to Alexandre taking a head start during winter break. During the month of February we hoped to implement the vision aspect of the cars. The car detection and distance estimation were done in the first week of February using april tags. For the track detection, our initial attempts used ping sensors on the sides and a track made from pool noodles. Due to the unreliability of the ultrasonic sensors, we rapidly realized that this wasn't a robust approach, so we transitioned to vision. We laid down yellow tape on the floor and managed to get the cars to detect the tape on the floor. This meant we were done with the vision aspect on time, yet still needed the car to actually follow the line. This is where the hardware began to break down. The response times were so slow we couldn't even manually control the cars to follow the line. By early March, we realized that hardware would not be feasible, and by this date we decided to shift to simulation.

The good thing is that we could use the same autonomy stack to realistically simulate the network aspect of the controller by having multiple computers, each acting as one car. So once we decided to switch to simulation, two of the three main milestones we had left were still intact: finishing the autonomy stack and implementing the controller in code. The only major milestone that changed is that now instead of having to finish the hardware aspect (the track following, the state estimation), we needed to build a simulator.

In the first week of march we managed to finish the autonomy stack by implementing the data logging and visualization portion; this would've been invaluable to the analysis of the controller later on. Unfortunately, due to the coronavirus outbreak the project was halted and we weren't able to implement the controller and the simulation.

We did much better at staying on top of our milestones this semester, and although we spent a lot of time on a fruitless part of the project (the hardware), the way we had initially built the project meant the transition to simulation was smooth and didn't require any overhead. This is also due in part to the fact that we learned from our mistakes and worked much better as a team in the spring.

VIII Discussion of teamwork

We set up a work-session from 9am - 5pm every Monday where we would all be working together (whenever not in class). This was the best day and time for us to host such a work-session as we always had at least 2 group members available at any time during that time slot. These sessions initially took place in Ketterer Lab and then moved to Moore 316 where we decided to set up our track. During that time slot we also had weekly meetings with Professor Morari at 4:30pm, which occurred either in his office or in Moore 316 depending on whether we needed to give him a demonstration or not. During these meetings we presented any completed work, set up new tasks for the upcoming week and assigned group members to work on those upcoming tasks. This system enabled us to keep people accountable for their work and ensured that every team member was aware of the current progress of the project. Furthermore, discussing the tasks as a group guaranteed that every group member approve of our next set of goals.

The weekly tasks were assigned to team members based on their strengths and availability.

Thanks to his strong background in control theory and embedded systems, Alexandre was the undisputed leader of our group. He worked on setting up the realistic autonomy stack and implementing Car-Tooning in software. With regards to the former, he designed the autonomy infrastructure and API allowing for easy integration of different controllers and additional modules. Further, he helped Sophia with the implementation of low-level controllers for steering and velocity and worked alongside Matteo on the vision-based track detection.

Sophia worked on the low-level feedback controllers for the steering and velocity. These were necessary because of the unreliability of the hardware and the need to correct any actuator error in the response to a given command. In addition, Sophia led our team's efforts in finding the right motivation and business model for our product. She organized a meeting with Professor Saikat Chaudhuri, that helped her come up with a flexible country-dependent business and pricing model. Sophia also coordinated many of the organizational efforts including setting up meetings, booking rooms, and setting short term goals in order to meet class deliverables.

Sergio dedicated a significant amount to the Track Detection system, which used the camera to recognize a track in front of it. He also deployed the data visualization, which retrieved the velocity data shared by vehicles on the network and dynamically plotted it in real-time. Sergio and Alexandre also spent a considerable amount of time formatting as best as possible any Latex documents our group submitted (including the poster and this report).

Matteo dedicated a significant amount of time to testing and fixing the sensors and micro-controllers on the ESE 421 cars, before producing the 5 cars that were ready to platoon. He then configured the cars on the network, calibrated the cameras for each vehicle, and developed the vision-based system for a car to determine its distance from the car in front of it. Finally, he worked with Alexandre on the steering control based on the vision-based track detection.

IX Budget and Justification

The team used vehicles from the Penn autonomous vehicle course, ESE 421. Using these same vehicles saved the team significant costs as purchasing a fleet of vehicles for testing is very expensive. Alternative vehicles that the team looked at cost upwards of thousands of dollars. In order to deploy our controller on hardware, we needed to purchase a router ($51.99). This allowed us to run our fleet of vehicles over an isolated network so that they can communicate with one another free of interference from other traffic. In addition, we initially planned to create a track for the vehicles using pool noodles ($62.69). This did not work as the vehicle ultrasonic sensors had a blind spot due to the location of their placement on the vehicle which was not rectifiable. These pool noodles were returned to Detkin Lab.

As an alternative, the team created a vision-based line following autonomy stack and created a new track using tape. The first tape ($7.57) did not contrast enough with the floor to be used and so was also returned to Detkin. The final yellow tape chosen by the team to create the track cost $25.95. The final budget the team spent (ignoring the items returned to the lab) was $77.94.

The team had no software licensing costs as all software used is available open source.

X Standards and Compliance

Platooning is primarily enabled by two technologies: autonomous vehicles and wireless communication. Both of these technologies are regulated and standardized by a variety of governing bodies. In this section we note a few examples of standards from both the autonomous vehicle and the wireless communication industries which inform the set of specifications to which we choose to design as outlined in section 3.2.

10.1 Standards Concerning Autonomous Vehicles

Autonomous vehicles are considered a strategically important emerging technology by the U.S. government and thus their development is closely monitored and road mapped by the United States Department of Transportation (USDOT). As outlined in [17], USDOT is concerned with ensuring the production of high performing and safe autonomous vehicles.

Though [17] comes short of giving quantitative specifications for these two criterion, it has nonetheless led to leaders in the AV industry to self-specify that the former means vehicles which achieve super-human response times to their environments while the latter requires these vehicles to satisfy the requirements of safety-critical systems.

In an influential study of the speed of the human visual system, Thorpe et. al. establish that the fastest human response time to visual stimuli is near 100ms [42]. Such a characterization of the limits of human response time has led the AV industry to specify an upper limit of 10Hz frequency of control decisions, or 100ms from input measurement to output control.

Furthermore, due to the safety-critical nature of autonomous driving, AV companies need to meet a variety of standards outlined by various bodies of the IEEE. The first is that safety-control systems qualify as a real-time system. This is regulated by the IEEE's Technical Committee on Real Time Systems (TCRTS). In [16], a real-time system in defined as "a system whose response time and delay time are deterministic without uncertainty and non-reproducibility and has an internal configuration that makes the worst value predictable or makes it easy to produce an educated guess value."

In addition to being real-time, as safety-critical control system must meet further specifications defined by the IEEE's Control System Society. A good general reference concerning the IEEE's definition of a

safety-critical control system can be found in [15]. Therein, they defined a safety-critical system as having "sufficient margins to allow redundant safety functions in case of failure" and go on to give quantitative examples of such.

10.2 Standards Concerning Communication

Rapid wireless communication is of paramount importance to enable "AVs to receive and contribute data beyond their on-board sensors' physical range" [17]. The improvement of wireless communications in the United States is the prerogative of the Federal Communications Committee (FCC) and a plan for the adoption of next generation, 5G communication can be found outlined in [18].

As such communication datarates are for now generally unavailable and could remain unavailable in large portions of the world, platooning technology should for now constrain itself to at least the 4G specifications for maximal communication datarates. These specifications can be found outlined in [13].

In order to provide redundant safety margins in the domain of autonomous platoons, it is advisable to assume that the communication network is a full generation behind. In the case of 3G technologies, this assumes a maximal communication datarate of 348kbits/s communicated over an IEEE 802.11p network. A comprehensive review of vehicle to vehicle communication can be found in [14].

XI Business Model

11.1 Value proposition

Car-Tooning eliminates traffic for everyone by driving the world's autonomous vehicles together. We deliver a technically sophisticated control scheme enabling a variety of multi-scale, coordinated maneuvers between individual vehicles, eradicating gridlock at a lower cost than any other proposed solution.

11.2 Stakeholders

Car-Tooning provides a privacy-preserving, traffic-reduction product which will save individuals on the road time and money. Our product promises cities and governments the opportunity to reduce noise and pollution while boosting their economic prospects through reduced infrastructure costs and improved transportation throughputs. Car-Tooning's pure software solution provides the 173.15 billion dollar (Market Watch, 2019) autonomous vehicle industry immediate benefits over human-based driving without the need for costly hardware adjustments. Our partner-based incentive systems encourage healthy competition between companies, while also reaping the maximum reward of indiscriminate collaboration.

11.3 Market Opportunity and Customer Segments

Due to the emergent nature of the autonomous vehicle industry, the relative influence of each stakeholder is difficult to predict and will likely vary across the globe. Car-Tooning's flexible business model aligns our incentive structure to the dominant stakeholder in various markets, ensuring the project's viability no matter the evolution of the industry. We demonstrate three likely scenarios in various international markets.

11.3.1 Model 1 Government Mandated Adoption: Beijing, China

The rapid growth of China's vehicle industry has made it a primary target market for autonomous vehicle companies. Already plagued by traffic problems, China's centralized government provides Car-Tooning a

prime opportunity for rapid adoption by simply mandating all autonomous vehicles to be equipped with platooning software. Due to the early development of our platooning software and its easy-to-implement design, Car-Tooning is uniquely positioned to be rapidly adopted as the standard for mandated platooning.

A revenue stream based usage fees will be difficult to sustain in a market defined by state-mandated adoption as it would require constantly maintaining significant technological superiority in order to be the standard of choice. Over time, Car-Tooning would move to a data based revenue stream, collecting information about the efficiency of our controller, the quality of vehicles in our platoons, and the movement statistics of populations at different scales which could be resold to various entities.

11.3.2 Model 2 Autonomous Vehicle Fleet: London, UK

In Europe, state subsidies and heavy regulation such as congestion pricing and carbon taxes have made communal transport a norm. Because of this, Uber and Lyft's communal transportation business models are more likely to succeed in London than Los Angeles where historically private car ownership has been preferred.

By promising to reduce their traffic and carbon footprint, Car-Tooning offers companies with large fleets of vehicles a way to reduce their own costs while promising added benefits to their customers. Early adoption would be facilitated by the need of such companies to cut costs on their new fleets. By strategically pricing below the government fee-burden, Car-Tooning will remain a competitive option for large vehicle companies looking to cut costs through efficient driving.

11.3.3 Model 3 Individual Vehicle Ownership: Los Angeles, CA, USA

The dramatic increase in the population in California has made traffic in cities like Los Angeles infamous. Despite government efforts, individual car ownership has persisted in these cities, leading Los Angeles to have the most extensive network of HOV lanes in the country. Individual drivers can be incentivized to adopt platooning software through the reduction of tolls to use such surfaces. To encourage adoption, drivers will be incentivized with a free trial period. Car-Tooning software will be offered standard on vehicles through partnerships with manufacturers.

The United State's favorable data collection laws currently would enable a similar revenue stream as proposed in China by reselling vital data collected during platooning operations. Additional revenue can be generated through a "freemium" subscription model. Users will have limited monthly access for free to Car-Tooning platoons, but will pay monthly fees for the rights to drive unlimited distances in our platoons, have access to the latest and most efficient software, and/or the ability to platoon on more roads.

11.4 Estimation of the size and growth of the market segment

The global market for self driving cars is expected to reach a global revenue of 173.15 billion dollars by 2023 (Self-driving Car Market Global Industry Trends). Furthermore, it is estimated that 50.7 billion dollars will be spent in 2023 alone on traffic management (Traffic Management Market). Car-Tooning promises to deliver significant value to this young, rapidly growing industry by reducing cities' infrastructure expenditures and providing essential differentiating features to autonomous vehicles.

11.5 Competition

Current competition comes primarily from the truck platooning industry which is focused on centralized coordination of heavy weight vehicles for the delivery sector. Though these platoons are hyper-efficient, such

solutions are hardware specific and are not scalable beyond a handful of trucks.

By contrast, Car-Tooning is fully decentralized and scalable to arbitrarily large platoons. As such, the project seeks to target the most common vehicle in cities: the everyday consumer vehicle. As large companies race to produce the first fully autonomous vehicle, Car-Tooning is positioning itself to be a leader in leveraging these new capabilities to solve traffic in a new way.

11.6 Cost

Car-Tooning strives to be a low-cost solution to traffic. This is achieved through a hardware-agnostic, pure software solution. While software has significant labor costs associated with its development, post-production maintenance and material costs will be very low.

Car-Tooning's primary costs will come in the form of testing, which will need to be done on as many autonomous vehicle models as possible.While our business model will be to have a free model available to drivers for the first year or two, Car-Tooning will reach net positive revenue within two to three years.

11.7 Revenue model

Car-Tooning has a flexible revenue model intended to align incentives with the most dominant stakeholder in each market category. To encourage early adoption, Car-Tooning will start as a free software. Three potential revenue streams present themselves based on market conditions 1) subscription pricing (sold either to the driver in monthly or annual subscriptions), 2) data collection revenue (in which we collect data on drivers habits and vehicle performance), 3) large-scale alleviating of government fees (in which companies pay Car-Tooning to help them reduce their carbon and traffic footprints to avoid carbon and congestion taxes).

XII Work Done Since Last Semester

During the fall semester, the majority of the team's effort was focused on selecting an appropriate project topic and scope, as well as deriving the theoretical aspects necessary to make a meaningful contribution to the field of platooning. As the majority of these theoretical developments were concluded at the end of last semester, efforts this semester were focused on creating a realistic test bed in which we could test our controller and on researching the business and societal aspects essential to transitioning our project from a research endeavor to a true product.

These efforts came largely in the development of our rudimentary autonomous vehicles. Significant effort was made in assembling the necessary hardware. Despite efforts, logistical challenges inhibited our ability to acquire encoders for the project in a timely manner. Nonetheless, working vehicles were assembled using the components outlined in section 3.5.1. We progressed by shifting our focus to developing the necessary software infrastructure to enable autonomous driving around a taped down track. This is fully detailed in 3.5.2-3.6. We are confident that on better performing hardware, our architecture could have resulted in limited autonomous behavior. However, given the limited nature of the hardware, we did not have sufficient enough fine grained control to enable the desired behavior. Nonetheless, we are proud of the system we built as it was sufficiently abstract to enable a transition to a simulated environment distributed across multiple computers. This final simulation was not completed due to lost time and resources from the coronavirus pandemic.

Finally, efforts were made to place our project in the appropriate business and societal contexts. This is discussed in detail in section XI. In this section, we outline the relevance of our project regardless of how the future of the autonomous vehicle industry involves. Platooning is seen as a natural and necessary development in an increasing mobile world. To prevent gridlock, technologies like Car-Tooning must be adopted, but this requires significant efforts to ensure the highest performance due to the safety-critical nature of the technology.

XIII Discussion and conclusion

Car-Tooning seeks to add value to existing platooning research by designing and implementing a robust controller under adverse conditions. This exercise spught to demonstrate the feasibility and challenges of translating theory specifically designed for non-idealities to a real system. Future stakeholders who would benefit include drivers, cities, and autonomous vehicle companies from the advancement of platooning knowledge. Effective platooning will enable drivers to reach their destination faster, saving them time and fuel costs by reducing traffic in a way no other technology is capable. Cities will benefit from less pollution as a result of less traffic and autonomous vehicle companies implementing Car-Tooning will need to make less significant upgrades in order to a satisfactory level of performance. This decreased cost will transfer to the consumer who will more easily be able to adopt an autonomous vehicle to platoon and reap the benefits of the technology.

This year, the team designed a provably stable and robust controller for platooning by propagating a stability certificate. These efforts are the first to our knowledge to attempt to improve platooning in this manner. The team attempted to deploy the controller on low-quality hardware however given constraints of the vehicles and the coronavirus outbreak, the team was unable to test the controller over a real 802.11p network.

These challenges will be met in the future by performing more edge cases of metrics for the hardware chosen to better identify these shortfalls early on. The team learned about control theory, specifically the theory of platooning in great detail, as well as applications of their product in society all over the world.

The team learned a lot about designing a project that take into account appropriate societal, environmental, or economic factors. Through numerous iterations and pivots, the team came up with a concrete problem and solution to that problem. Since our project is extremely technical, another challenge that the team faced was articulating how our project works and why its relevant. Based on the feedback we received during class presentations, the team worked hard to frame our project to audiences without a controls background.

Lastly, the team also learned a lot about time management. While the main goals of this project is to design the controller, software simulation, and hardware implementation there were also many other deliverables that needed to be met. The team worked together in order to prioritize based on deadlines and distribute work efficiently among team members.

References

[1] E Jaffe, *The U.S. Transportation System Has $100 Billion Worth of Inefficiencies.*, 2013.

[2] "San francisco curb study - no links," Oct 2018.

[3] Peter T Martin, J Perrin, Peiling Wu, and R Lambert, "Evaluation of the effectiveness of high occupancy vehicle lanes," *Report to Utah Department of Transportation*, 2004.

[4] Chris Giarratana, "Green lighting hov lanes: Do they help traffic problems? " traffic safety resource center," Mar 2019.

[5] Michael Zabat, Nick Stabile, Stefano Farascaroli, and Frederick Browand, "The aerodynamic performance of platoons: A final report," 1995.

[6] Gerrit JL Naus, Rene PA Vugts, Jeroen Ploeg, Marinus JG van De Molengraft, and Maarten Steinbuch, "String-stable cacc design and experimental validation: A frequency-domain approach," *IEEE Transactions on vehicular technology*, vol. 59, no. 9, pp. 4268–4279, 2010.

[7] Jan Lunze, "Design of the communication structure of cooperative adaptive cruise controllers for vehicular platoons," 2019.

[8] Shahab Sheikholeslam and Charles A Desoer, "Longitudinal control of a platoon of vehicles with no communication of lead vehicle information: A system level study," *IEEE Transactions on vehicular technology*, vol. 42, no. 4, pp. 546–554, 1993.

[9] Xiangheng Liu, Andrea Goldsmith, Sunider Sonia Mahal, and J Karl Hedrick, "Effects of communication delay on string stability in vehicle platoons," in *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 01TH8585)*. IEEE, 2001, pp. 625–630.

[10] D. Swaroop and J. K. Hedrick, "String stability of interconnected systems," *IEEE Transactions on Automatic Control*, vol. 41, no. 3, pp. 349–357, March 1996.

[11] W. Saad M. Bennis T. Zeng, O. Semiari, "Joint communication and control for wireless autonomous vehicular platoon systems," 2019.

[12] Ferry Grijpink, Alexandre Menard, Halldor Sigurdsson, and Nemanja Vucevic, "The road to 5g: The inevitable growth of infrastructure cost," 2018.

[13] "Ieee standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (lr-wpans) amendment 3: Physical layer (phy) specifications for low-data-rate, wireless, smart metering utility networks," *IEEE Std 802.15.4g-2012 (Amendment to IEEE Std 802.15.4-2011)*, pp. 1–252, 2012.

[14] José Manuel Lozano Domínguez and Tomás Jesús Mateo Sanguino, "Review on v2x, i2x, and p2x communications and their applications: A comprehensive analysis over time," *Sensors*, vol. 19, no. 12, pp. 2756, 2019.

[15] author. Smith, David J., *The safety critical systems handbook : a straightforward guide to functional safety: IEC 61508 (2010 edition), IEC 61511 (2016 edition) & related guidance, including machinery and other industrial sectors /*, Butterworth-Heinemann,, Amsterdam, [Netherlands] :, fourth edition. edition.

[16] "Ieee standard for a real-time operating system (rtos) for small-scale embedded systems," *IEEE Std 2050-2018*, pp. 1–333, 2018.

[17] Elaine L Chao and Michael Kratsios, "Ensuring american leadership in automated vehicle technologies: Automated vehicles 4.0," Jan 2020.

[18] Ajit Pai, "Facilitate america's superiority in 5g technology plan," Sep 2016.

[19] Charles L Phillips and Royce D Habor, *Feedback control systems*, Simon & Schuster, Inc., 1995.

[20] David Gordon Wilson, "Automated guideway transportation between and within cities," Tech. Rep., 1971.

[21] Stephen Boyd and Lieven Vandenberghe, *Convex Optimization*, Cambridge University Press, New York, NY, USA, 2004.

[22] Ekkehard W Sachs, "Semi-infinite programming in control," in *Semi-Infinite Programming*, pp. 389–411. Springer, 1998.

[23] Karen Hao, "Training a single ai model can emit as much carbon as five cars in their lifetimes," Apr 2020.

[24] Assad Al Alam, Ather Gattami, and Karl Henrik Johansson, "Suboptimal decentralized controller design for chain structures: Applications to vehicle formations," *IEEE Conference on Decision and Control and European Control Conference*, pp. 6894–6900, 2011.

[25] Rudolf Emil Kalman et al., "Contributions to the theory of optimal control," *Bol. soc. mat. mexicana*, vol. 5, no. 2, pp. 102–119, 1960.

[26] L. Dritsas, G. Nikolakopoulos, and A. Tzes, "Constrained optimal control over networks with uncertain delays," in *Proceedings of the 45th IEEE Conference on Decision and Control*, Dec 2006, pp. 4993–4998.

[27] B. Wittenmark K. J. Astrom, *Computer Controlled Systems*, Englewood Cliffs: Prentice-Hall, 1997.

[28] A. Bemporad, F. Borrelli, and M. Morari, "Min-max control of constrained uncertain discrete-time linear systems," *IEEE Transactions on Automatic Control*, vol. 48, no. 9, pp. 1600–1606, Sep. 2003.

[29] E. Feron S.P. Boyd, L. El Ghaoui and V. Balakrishnan, "Linear matrix inequalities in systems and control theory," vol. 15, Society for Industrial and Applied Mathematics(SIAM).

[30] Francesco Borrelli, Alberto Bemporad, and Manfred Morari, *Predictive control for linear and hybrid systems*, Cambridge University Press, 2017.

[31] W. B. Dunbar and D. S. Caveney, "Distributed receding horizon control of vehicle platoons: Stability and string stability," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 620–633, March 2012.

[32] Michele Segata, Stefan Joerer, Bastian Bloessl, Christoph Sommer, Falko Dressler, and Renato Lo Cigno, "PLEXE: A Platooning Extension for Veins," in *6th IEEE Vehicular Networking Conference (VNC 2014)*, Falko Dressler, Onur Altintas, Suman Banerjee, Björn Scheuermann, and David Eckhoff, Eds., Paderborn, Germany, 12 2014, pp. 53–60, Institute of Electrical and Electronics Engineers.

[33] Nathan Koenig and Andrew Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, Sep 2004, pp. 2149–2154.

[34] Shamik Sural, Gang Qian, and Sakti Pramanik, "Segmentation and histogram generation using the hsv color space for image retrieval," in *Proceedings. International Conference on Image Processing*. IEEE, 2002, vol. 2, pp. II–II.

[35] XM Pang, ZJ Min, and JM Kan, "Color image segmentation based on hsi and lab color space," *Guangxi Daxue Xuebao(Ziran Kexue Ban)*, vol. 36, no. 6, pp. 976–980, 2011.

[36] Elisabeth Ågren, "Lateral position detection using a vehicle-mounted camera," 2003.

[37] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. 2018, IEEE.

[38] Christoph Sommer, Reinhard German, and Falko Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3–15, 1 2011.

[39] Earl Ritchie, "Self-driving automobiles: How soon and how much?," May 2019.

[40] Mark Fahey, "Driverless cars will destroy the most jobs in these us states," Sep 2016.

[41] Quoctrung Bui, "Map: The most common* job in every state," Feb 2015.

[42] Simon Thorpe, Denis Fize, and Catherine Marlot, "Speed of processing in the human visual system," *nature*, vol. 381, no. 6582, pp. 520–522, 1996.

A Appendices

1.1 Generalized Decentralized Suboptimal Chain Graph Controller

Algorithm 1 Decentralized suboptimal control strategy

1: Set $\boldsymbol{Q}_i$, $\boldsymbol{R}^{(i)}$ for $i = 0, \ldots, n$ according to the desired performance criteria

2: Solve the lead vehicle control according to:

$$\min_{\boldsymbol{u}^{(1)}} \int_{t_0}^{t_f} \left(\boldsymbol{x}^{(0)}(t)\right)^T \boldsymbol{Q}^{(0)} \boldsymbol{x}^{(0)}(t) + \left(\boldsymbol{u}^{(0)}(t)\right)^T \boldsymbol{R}^{(0)} \boldsymbol{u}^{(0)}(t)$$
$$\text{subject to } \dot{\boldsymbol{x}}^{(0)}(t) = \boldsymbol{A}^{00} \boldsymbol{x}^{(0)}(t) + \boldsymbol{B}^{(0)} \boldsymbol{u}^{(0)}(t)$$

using the classical LQR solution [30] yielding:

$$\boldsymbol{u}^0(t) = -K_{00} \boldsymbol{x}^{(0)}$$

Communicate $\boldsymbol{x}^{(0)}(t)$ and $\boldsymbol{K}_{00}$ to the adjacent vehicles in the network topology.

3: For each vehicle $i = 1, \ldots, N$ solve

$$\min_{u_i} \int_{t_0}^{t_f} \left(\begin{bmatrix} \boldsymbol{x}^{(i-1)}(t) \\ \boldsymbol{x}^{(i)}(t) \end{bmatrix} \right)^T \boldsymbol{Q}^{(i)} \begin{bmatrix} \boldsymbol{x}^{(i-1)}(t) \\ \boldsymbol{x}^{(i)}(t) \end{bmatrix} + \left(\boldsymbol{u}^{(i)}(t)\right)^T \boldsymbol{R}^{(i)} \boldsymbol{u}^{(i)}(t)$$
$$\text{subject to } \begin{bmatrix} \dot{\boldsymbol{x}}^{(i-1)}(t) \\ \dot{\boldsymbol{x}}^{(i)}(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}^{(i-1)(i-1)} - \boldsymbol{B}^{(i-1)} \boldsymbol{K}^{(i-1)} & 0 \\ \boldsymbol{A}^{i(i-1)} & \boldsymbol{A}^{(ii)} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^{(i-1)}(t) \\ \boldsymbol{x}^{(i)}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \boldsymbol{B}^{(i)} \end{bmatrix} \boldsymbol{u}^{(i)}(t)$$

by solving:

$$\dot{\boldsymbol{S}}^{(i)} = \boldsymbol{S}^{(i)} \begin{bmatrix} 0 \\ \boldsymbol{B}^{(i)} \end{bmatrix} \left(\boldsymbol{R}^{(i)}\right)^{-1} \begin{bmatrix} 0 \\ \boldsymbol{B}^{(i)} \end{bmatrix}^T \boldsymbol{S}^{(i)} - \begin{bmatrix} \boldsymbol{A}^{(i-1)(i-1)} - \boldsymbol{B}^{(i-1)} \boldsymbol{K}^{(i-1)(i-1)} & 0 \\ \boldsymbol{A}^{i(i-1)} & \boldsymbol{A}^{(ii)} \end{bmatrix}^T \boldsymbol{S}^{(i)} -$$
$$\boldsymbol{S}^{(i)} \begin{bmatrix} \boldsymbol{A}^{(i-1)(i-1)} - \boldsymbol{B}^{(i-1)} \boldsymbol{K}^{(i-1)(i-1)} & 0 \\ \boldsymbol{A}^{i(i-1)} & \boldsymbol{A}^{(ii)} \end{bmatrix} - \boldsymbol{Q}^{(i)}$$

$$\begin{bmatrix} \boldsymbol{K}^{i(i-1)} & \boldsymbol{K}^{ii} \end{bmatrix} = \left(\boldsymbol{R}^{(i)}\right)^{-1} \begin{bmatrix} 0 \\ \boldsymbol{B}^{(i)} \end{bmatrix} \boldsymbol{S}^{(i)}$$
$$\boldsymbol{u}^{(i)}(t) = \begin{bmatrix} \boldsymbol{K}^{i(i-1)} & \boldsymbol{K}^{ii} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^{(i-1)}(t) \\ \boldsymbol{x}^{(i)}(t) \end{bmatrix}$$

1.2 Control of Discrete, Finite Time Horizon Linear Systems

For completeness, we include a brief overview of the optimal control of discrete, finite horizon linear systems. For a more thorough treatment of the matter, the reader is referred to [30]

1.2.1 The Unconstrained Case

We consider the problem of controlling a linear time invariant system

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \tag{36}$$

over a time horizon of N steps subject to the quadratic cost

$$J_i(\boldsymbol{x}_i, \mathcal{U}) = \boldsymbol{x}_{i+N}^T \boldsymbol{P} \boldsymbol{x}_{i+1} + \sum_{k=i}^{N+i-1} \boldsymbol{x}_k^T \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k \tag{37}$$

where $\mathcal{U} = \{\boldsymbol{u}_i, \boldsymbol{u}_{i+1}, \ldots, \boldsymbol{u}_{i+N-1}\}$ is the input sequence, $\boldsymbol{P} = \boldsymbol{P}^T \succeq 0$ is the terminal cost, $\boldsymbol{Q} = \boldsymbol{Q}^T \succeq 0$ is the state cost, and $\boldsymbol{R} = \boldsymbol{R}^T \succ 0$ is the input cost.

The optimal control of this system to the origin is written as the optimization problem:

$$J_i^*(\boldsymbol{x}_i) = \min_{\mathcal{U}} \boldsymbol{x}_{i+N}^T \boldsymbol{P} \boldsymbol{x}_{i+1} + \sum_{k=i}^{N+i-1} \boldsymbol{x}_k^T \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k \tag{38}$$

$$\text{subject to } \boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \tag{39}$$

This problem is known as the linear quadratic regulator. It has a well known solution purely in terms of the initial state $\boldsymbol{x}_i$ which can be achieved either in batch or recursively through dynamics programming [30]. The optimal input at time k is given by:

$$\boldsymbol{u}_k^* = -\left(\boldsymbol{B}^T \boldsymbol{P}_{k+1} \boldsymbol{B} + \boldsymbol{R}\right)^{-1} \boldsymbol{P}_{k+1} \boldsymbol{A} \boldsymbol{x}_k = -\boldsymbol{K}_k \boldsymbol{x}_k \tag{40}$$

$$\boldsymbol{P}_k = \boldsymbol{A}^T \boldsymbol{P}_{k+1} \boldsymbol{A} + \boldsymbol{Q} - \boldsymbol{A}^T \boldsymbol{P}_{k+1} \boldsymbol{B} (\boldsymbol{B}^T \boldsymbol{P}_{k+1} \boldsymbol{B} + \boldsymbol{R})^{-1} \boldsymbol{B}^T \boldsymbol{P}_{k+1} \boldsymbol{A} \tag{41}$$

1.2.2 The Constrained Case

We consider the same problem as in (38) except we add polytopic constraints to our variables:

$$\begin{aligned} J_i^*(\boldsymbol{x}_i) = \min_{\mathcal{U}} \; & \boldsymbol{x}_{i+N}^T \boldsymbol{P} \boldsymbol{x}_{i+1} + \sum_{k=i}^{N+i-1} \boldsymbol{x}_k^T \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k \\ \text{subject to } & \boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \\ & \boldsymbol{C}_x^{(k)} \boldsymbol{x}_k \leq \boldsymbol{b}_x^{(k)} \\ & \boldsymbol{C}_u^{(k)} \boldsymbol{u}_k \; \leq \boldsymbol{b}_u^{(k)} \end{aligned} \tag{42}$$

By constructing the matrices:

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{C}_u^{(i)} & 0 & \ldots & 0 \\ 0 & \boldsymbol{C}_u^{(i+1)} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \boldsymbol{C}_u^{(i+1)} \\ 0 & 0 & \ldots & 0 \\ \boldsymbol{C}_x^{(i)} \boldsymbol{B} & 0 & \ldots & 0 \\ \boldsymbol{C}_x^{(i+1)} \boldsymbol{A}\boldsymbol{B} & \boldsymbol{C}_x^{(i)} \boldsymbol{B} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{C}_x^{(N+i)} \boldsymbol{A}^{N+i-1} \boldsymbol{B} & \boldsymbol{C}_x^{(N+i)} \boldsymbol{A}^{N+i-2} \boldsymbol{B} & \ldots & \boldsymbol{C}_x^{(N+i)} \boldsymbol{B} \end{bmatrix}, \; \boldsymbol{E}_0 = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -\boldsymbol{C}_x^{(i)} \\ -\boldsymbol{C}_x^{(i+1)} \boldsymbol{A} \\ -\boldsymbol{C}_x^{(i+2)} \boldsymbol{A}^2 \\ \vdots \\ -\boldsymbol{C}_x^{(i+N-1)} \boldsymbol{A}^n \end{bmatrix}, \; \boldsymbol{w}_0 = \begin{bmatrix} \boldsymbol{b}_u^{(i)} \\ \boldsymbol{b}_u^{(i+1)} \\ \vdots \\ \boldsymbol{b}_u^{(i+N-1)}) \\ \boldsymbol{b}_x^{(i)} \\ \boldsymbol{b}_x^{(i+1)} \\ \vdots \\ \boldsymbol{b}_x^{(i+N)}) \end{bmatrix}$$

$$
\boldsymbol{H} = \begin{bmatrix} \boldsymbol{B} & & & \\ & \boldsymbol{B} & & \\ & & \ddots & \\ & & & \boldsymbol{B} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{Q} & & & \\ & \boldsymbol{Q} & & \\ & & \ddots & \\ & & & \boldsymbol{P} \end{bmatrix} \begin{bmatrix} \boldsymbol{B} & & & \\ & \boldsymbol{B} & & \\ & & \ddots & \\ & & & \boldsymbol{B} \end{bmatrix} + \begin{bmatrix} \boldsymbol{R} & & & \\ & \boldsymbol{R} & & \\ & & \ddots & \\ & & & \boldsymbol{R} \end{bmatrix}
$$

$$
\boldsymbol{F} = \begin{bmatrix} \boldsymbol{A} & & & \\ & \boldsymbol{A} & & \\ & & \ddots & \\ & & & \boldsymbol{A} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{Q} & & & \\ & \boldsymbol{Q} & & \\ & & \ddots & \\ & & & \boldsymbol{P} \end{bmatrix} \begin{bmatrix} \boldsymbol{B} & & & \\ & \boldsymbol{B} & & \\ & & \ddots & \\ & & & \boldsymbol{B} \end{bmatrix}
$$

$$
\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{A} & & & \\ & \boldsymbol{A} & & \\ & & \ddots & \\ & & & \boldsymbol{A} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{Q} & & & \\ & \boldsymbol{Q} & & \\ & & \ddots & \\ & & & \boldsymbol{P} \end{bmatrix} \begin{bmatrix} \boldsymbol{A} & & & \\ & \boldsymbol{A} & & \\ & & \ddots & \\ & & & \boldsymbol{A} \end{bmatrix}
$$

and allowing $\boldsymbol{U} = vstack(\{\boldsymbol{u}_k\}_{k=i}^{i+N-1}$ then the problem in (42) can be rewritten as the quadratic program:

$$
\begin{aligned} J_i^*(\boldsymbol{x}_i) = \min_{\boldsymbol{U}} \; & \begin{bmatrix} \boldsymbol{U}^T & \boldsymbol{x}_i \end{bmatrix} \begin{bmatrix} \boldsymbol{H} & \boldsymbol{F}^T \\ \boldsymbol{F} & \boldsymbol{Y} \end{bmatrix} \begin{bmatrix} \boldsymbol{U}^T & \boldsymbol{x}_i \end{bmatrix}^T \\ & \text{subject to } \boldsymbol{G}\boldsymbol{U} \leq \boldsymbol{w} + \boldsymbol{E}\boldsymbol{x}_i \end{aligned} \tag{43}
$$

For which a number of methods exist for solving such quadratic programs [21].

Alternatively, by performing the substitution:

$$
\boldsymbol{z} \coloneqq \boldsymbol{U} + \boldsymbol{H}^{-1}\boldsymbol{F}^T\boldsymbol{x}_i
$$

we can write (42) as:

$$
\begin{aligned} J_i^*(\boldsymbol{x}_i) = \min_{\boldsymbol{z}} & \boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} \\ & \text{subject to} \boldsymbol{G}\boldsymbol{z} \leq \boldsymbol{w} + \left(\boldsymbol{E} + \boldsymbol{G}\boldsymbol{H}^{-1}\boldsymbol{F}^T\right)\boldsymbol{x}_i \end{aligned} \tag{44}
$$

This is a multiparametric quadratic program whose explicit solution can be computed with respect to the initial value $\boldsymbol{x}_i$. This enables the offline computation of the entire set of possible controllers, converting the problem of finding the optimal control from and $\mathcal{O}(N * (\dim(\boldsymbol{x}_k) + \dim(\boldsymbol{u}_k)\log(N))$ problem to a $\mathcal{O}(\log_2(\dim(\boldsymbol{x}_k))$ [21], [30].