



University of Pennsylvania  
ScholarlyCommons

---

Real-Time and Embedded Systems Lab (mLAB)

School of Engineering and Applied Science

---

9-22-2019

# Tech Report: TUNERCAR: A Superoptimization Toolchain for Autonomous Racing

Matthew O'Kelly

*University of Pennsylvania*, mokelly@seas.upenn.edu

Hongrui Zheng

*University of Pennsylvania*, billyzheng.bz@gmail.com

Achin Jain

*University of Pennsylvania*, achinj@seas.upenn.edu

Joseph Auckley

*University of Pennsylvania*, jauckley@seas.upenn.edu

Kim Luong

*University of Pennsylvania*, kimluong@seas.upenn.edu

*See next page for additional authors*

Follow this and additional works at: [https://repository.upenn.edu/mlab\\_papers](https://repository.upenn.edu/mlab_papers)

 Part of the [Computer Engineering Commons](#), and the [Electrical and Computer Engineering Commons](#)

---

## Recommended Citation

Matthew O'Kelly, Hongrui Zheng, Achin Jain, Joseph Auckley, Kim Luong, and Rahul Mangharam, "Tech Report: TUNERCAR: A Superoptimization Toolchain for Autonomous Racing", . September 2019.

---

# Tech Report: TUNERCAR: A Superoptimization Toolchain for Autonomous Racing

**Disciplines**

Computer Engineering | Electrical and Computer Engineering

**Author(s)**

Matthew O'Kelly, Hongrui Zheng, Achin Jain, Joseph Ackley, Kim Luong, and Rahul Mangharam

# Tech Report: TUNERCAR: A Superoptimization Toolchain for Autonomous Racing

Matthew O’Kelly<sup>1</sup>, Hongrui Zheng<sup>1</sup>, Joseph Ackley<sup>1</sup>, Achin Jain<sup>1</sup>, Kim Luong<sup>1,2</sup>, and Rahul Mangharam<sup>1</sup>

**Abstract**—The objective of this effort is to develop an optimal autonomous racer with safe and reusable core autonomy components that are agnostic to vehicle planning and control software. TUNERCAR is a toolchain that jointly optimizes racing strategy, planning methods, control algorithms and vehicle parameters for an autonomous racecar. In this paper, we detail the target hardware, software, simulators, and systems infrastructure for this toolchain. Our methodology employs a massively parallel implementation of CMA-ES which enables simulations to proceed 6 times faster than real-world rollouts. Besides a massive speed up, we show our approach can reduce the lap times in autonomous racing, given a fixed computational budget. We demonstrate improvements over naive random search of 2.0 seconds per lap, improvements over expert solutions of 0.81 seconds per lap, and beat a human driver by 6.52 seconds. For all tested tracks, our method provides the lowest lap-time, and relative lap-time improvements between 6 and 12 percent. We further compare the performance of our method against hand tuned solutions submitted by over 30 international teams, comprised of graduate students working in the field of autonomous vehicles. Finally, we discuss the effectiveness of utilizing an online planning mechanism to reduce the reality gap between our simulation and actual tests. By driving at the limits of vehicle performance, we are able to efficiently accelerate the development of safe autonomous vehicles.

## I. INTRODUCTION

Since its inception, racing has been a key driver of new technology in the automotive industry. Some domains, such as powertrain engineering, are obvious beneficiaries of racing development. However, the goals of racing – fast and aggressive driving – are seemingly orthogonal to the safety oriented specifications of the autonomous vehicle industry. Nevertheless, scenarios faced by race car drivers (and autonomous racers) force the development of technology which must operate in both nominal conditions and more importantly *at the limits* of vehicle performance.

The objective of developing an optimal autonomous racer is motivated by the desire to create safe and reusable core autonomy components – namely vehicle agnostic planning and control software. Racing, in this context, is a mechanism to create a competitive environment where the quality of the chosen vehicle configuration has a clear and continuous measure: lap time. While nominal conditions may be handled with even poorly integrated components (*e.g.* a pure pursuit controller which works even when accidentally used on the wrong robot [1]), racing conditions severely punish sub-

optimal tuning in vehicle dynamics, racing strategy (path and speed selection) and tracking controller parameters.

This paper introduces the notion that component reuse and adaptation is analogous to creating a new kind of compiler that targets computational, physical, and external environmental details of a robot’s operational domain. In general, the goal of a compiler is to validate and then transform a source program in one language to another (usually lower-level *e.g.* assembly) which is suitable for the target domain [2]. Modern optimizing compilers [3] also seek to improve the performance of the transformed program. To concretize the analogy, we define the source program as a parameterized description of the vehicle dynamics, tracking controllers, and local planning method which we wish to transform to perform safely and efficiently in the operational environment represented by a map, physical laws, and sensing capabilities.

We propose a solution to the cyber-physical compilation problem utilizing the concept of *superoptimization* (*c.f.* [4], [5], [6], [7]), a technique that searches the space of equivalent and correct programs for the most performant instance rather than applying a sequence of optimization passes which attack specific performance bottlenecks. Despite the success of superoptimization approaches in limited domains [8], [9], viewing the autonomous vehicle as the compilation target creates entirely new issues due to the cyber-physical nature of the platform. First, no simulator or model can perfectly emulate the target thus creating a noisy performance measure and potentially falsifying correctness claims. Second, ameliorating this reality gap by executing proposed program transformations on the vehicle is dangerous, slow, and expensive. The vehicle may get damaged due to aggressive testing, tests cannot proceed faster than real time, and evaluations cannot be easily parallelized. In response to these challenges, our solution provides a validated, deterministic, and massively parallel simulation environment as well as a method of adapting derived strategies to reality via an efficient online optimization component.

There are three primary contributions of this work. First, we provide a compilation toolchain (Fig. 1) for superoptimization of autonomous racers, TUNERCAR. This toolchain includes target hardware, modular software, and a calibrated simulator. Second, we describe a methodology for *tuning* a high-dimensional set of hyperparameters spanning control, planning, dynamics, and strategy; it is a general approach for adapting and optimizing heterogeneous components to new robots and domains. Lastly, we validate our solution on an AV software stack and 1/10th-scale open-source vehicle developed for this project [[f1tenths.org](http://f1tenths.org)].

<sup>1</sup>Authors are with the Real-Time and Embedded Systems Laboratory, University of Pennsylvania, Philadelphia, USA

<sup>2</sup>Authors are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, USA

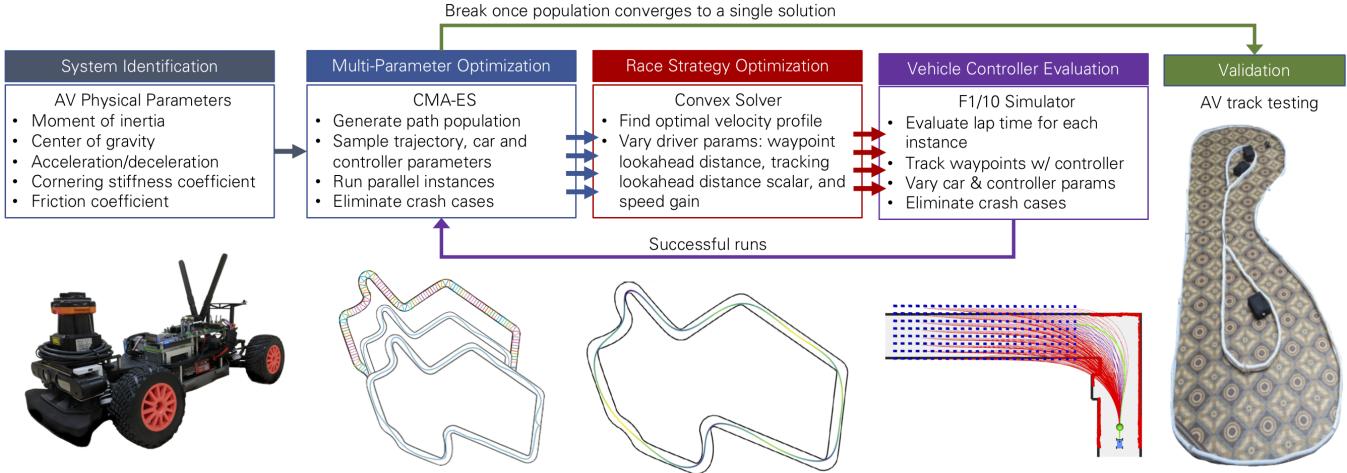


Fig. 1: TunerCar Toolchain

In what follows, we demonstrate TUNERCAR, a superoptimization toolchain, which achieves a 2 second improvement in race lap time relative to other approaches tested with a fixed computation budget and beats crowd-sourced expert solutions by up to 12%. Section II describes the problem setup and optimization pipeline. Section III places our solution in context with previous approaches. Section IV describes the modular autonomy stack, simulator, and hardware utilized for large-scale experiments detailed and analyzed in Section V. Section VI details future work and conclusions.

## II. METHODOLOGY

### A. Problem Statement

Given an environment (here, a racetrack encoded as a map), an objective function (here, lap-time and implicitly both safety and correctness), and a parameterized model of the agent dynamics, strategy, and controller, we wish to find an assignment of parameters which minimize the objective function.

The search space, parameterized by  $\Theta \in \mathbb{R}^n$ , is the concatenation of three components: agent dynamics, strategy and controllers. The objective function,  $f(\Theta) : \mathbb{R}^n \rightarrow \mathbb{R}$ , is computed by simulating the system, producing a trajectory  $\phi_\theta(t)$ . Note that in this work we utilize a simulator where the resulting scalar measure is deterministic given an assignment of parameters  $\theta \in \Theta$ ; however, no other assumptions such as the existence of a gradient are made.

### B. Search Space

As shown in Fig. 1, the compilation framework searches across vehicle parameters, splits the optimization in two stages and the validation in simulation, and eventually with track testing. The physical parameters,  $\Theta_p$  are defined as mass  $m$ , center of gravity longitudinal position  $l_g$ , friction coefficient  $\mu_s$ , center of gravity height  $h_g$ , cornering stiffness front  $C_{\alpha_f}$ , and cornering stiffness rear  $C_{\alpha_r}$ . We limit the range of these parameters such that they are physically achievable without major modifications, nominal values and their ranges

are based on system identification performed on the actual vehicle, Appendix A.

For the PHILADELPHIA map,  $\Theta \in \mathbb{R}^{139}$ . The strategy parameters,  $\Theta_s$ , are vectors, defined as the nominal path  $(x, y)$  and the velocity profile  $(v_x, v_y)$ ; the size varies depending on the track; on PHILADELPHIA,  $\Theta_s \in \mathbb{R}^{260}$ . The path and velocity profile combined, represent the largest portion of the search. In order to improve the convergence of our proposed optimization method, we note that a deterministic assignment of the optimal velocity profile is possible given a path  $x, y$ , reducing the size of the search space by a factor of two. This optimal velocity profile can be determined efficiently in polynomial time by setting up a convex optimization problem for minimizing the lap time. Due to space constraints, full details of the minimum time path traversal formulation can be found in Appendix A, Appendix B, and [10].

Finally, the driver parameters  $\Theta_d$  represent aspects of the control algorithms used, there are three components: the waypoint lookahead distance  $d_{wl}$ , tracking lookahead distance scalar  $d_{tl}$ , and the speed gain  $v_g$ .

### C. Evaluation Criteria

Solutions are determined to be feasible if the trajectory does not intersect with the boundaries of the track (accounting for the car's height and width); infeasible solutions are rejected and replaced at the sampling stage described in the next subsection. Equipped with a set of waypoints with velocities, car parameters and controller parameters for each solution, we then calculate the lap time using our simulator detailed in Section IV. The simulator updates the sampled parameters in the agent dynamics model, strategy, and controllers and attempts to traverse the track. The simulated lap time is used as the objective function which sorts the population of samples into quantiles.

### D. Optimization

The search proposed is both high-dimensional and challenging. Specifically it is non-convex, non-smooth and non-analytic, we attempt to solve the optimization problem with a gradient-free black-box optimization technique. A

potential pitfall of this approach is the existence of many local minima which could limit the exploration of the space. One method, CMA-ES, [11] as described in Algorithm 1 is known to perform well in such regimes, because it explicitly balances exploration with hill-climbing. Specifically, CMA-ES increases the covariance matrix of the parameters when the top performers of the population are far from the rest of the population, and decreases it as the population converges together. This gradient-like process is key to avoiding many local minima in this high-dimensional space. Lastly, this type of optimization also allows us to massively parallelize the evaluation of solutions, which is the most computationally expensive aspect of this pipeline.

---

**Algorithm 1** Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

---

- 1: **Input:** Population size  $\lambda$ , parent number  $\mu$
  - 2: Randomly initialize population:  $\theta_k$  for  $k = 1, \dots, \lambda$
  - 3: **while** termination criterion not met **do**
  - 4:   Calculate population means  $\langle \theta \rangle^{(g)}$  of the parameters
  - 5:   Evaluate current population  $\theta^{(g)}$  using  $f(\theta)$
  - 6:   Sort  $\theta$  from smallest to largest objective,  $f(\theta)$
  - 7:   Isolate top  $\mu$  individuals:  $\theta_k$  for  $k = 1, \dots, \mu$
  - 8:   Estimate the covariance matrix  $C^{(g+1)}$  of the top individuals using  $\langle \theta \rangle^{(g)}$
  - 9:   Calculate the means  $\langle \theta \rangle^{(g+1)}$  of  $\theta_k$  for  $k = 1, \dots, \mu$
  - 10:   Sample a new population of  $\lambda$  individuals using  $\langle \theta \rangle^{(g+1)}$  and  $C^{(g+1)}$
  - 11: **end while**
- 

We first randomly initialize a population of feasible solutions from uniform distributions. Once the population has been evaluated, the top  $\mu$  solutions are isolated. In Section V we detail the results of a grid search to identify high-performing, generalizable settings for  $\mu$  and the population size. The top quantile of solutions are used to fit a multivariate Gaussian distribution from which the next generation of samples are drawn. This process repeats until a convergence condition is reached. Following [11] our convergence condition is met when the norm of the covariance matrix is below  $\epsilon = 0.1$ . This condition is determined empirically in Section V.

### III. RELATED WORK

This paper presents a general approach to heterogenous component integration and optimization suitable for robotic systems that interact with a physical environment. A complete discussion of superoptimization and compiler literature has been omitted; however, specific approaches such as program sketching, population-based training, and program synthesis are of interest to the robotics community. The approach presented in Section II utilizes a black-box optimization (*c.f.* [12]). We note that most literature in superoptimization employs similar search algorithms. In contrast, a narrower view of this work requires a comparison of numerous attempts to optimize specific vehicle components within the race car engineering community. Amongst the domain specific knowledge presented, we utilize a convex formulation of the

minimum-time path traversal problem [10] in order to increase the empirical convergence rate of the CMA-ES method further differentiating our toolchain.

Sketching [6] utilizes fragments of programs which capture macroscopic details about the structure of the solution in order to synthesize the low-level details. In contrast, program synthesis [13] attempts to construct a correct program from scratch using only formal specifications. Neither approach inherently considers optimality, only correctness. Examples of sketching [14] and program synthesis [15] techniques applied to robotics generally fail to address the gap between models of robotic systems and realizable interactions between the program and the world. Even still, a variety of computational complexity and undecidability results [16] remain a barrier to applying these methods to realistic systems. Another closely related approach, population-based training (PBT) [17] has recently been utilized to search for efficient neural network designs; however, earlier work [18] focused only on physical robot design, a subset of the problem addressed in this paper. Likewise, [19] directly addresses the synthesis of a controller and planner interface leaving the vehicle static and but does not address the real-world applicability of the approach, specifically with respect to the (lack-of) realism in the simulator and vehicle dynamics.

Just as in the more general literature, race car optimization has historically been divided into two main categories: vehicle parameter optimization and racing trajectory optimization. The former uses a fixed track and fixed trajectory while varying the car parameters while the latter uses a fixed track and a fixed car while varying the trajectories. In [20] 500 pairs of roll stiffness and weight distribution values are evaluated using pareto minimum analysis to optimize for the fastest lap time. More recently, [21], [22], [23], feature expanded design spaces, parallel computation, and utilize simple genetic algorithms.

In raceline optimization several approaches for convexifying the problem have been proposed: [24] explored an iterative two-step algorithm that separates the longitudinal and lateral control components. Alternatively, [10] proposes a method which, *given a path* solves for both lateral and longitudinal forces and then transforms the solution into a velocity profile. In this paper, we apply this procedure to determine optimal raceline by searching over different trajectories. In another related work [25], the autonomous racecar performance is optimized in two steps. First system identification is done offline and then the lap time is minimized using a model predictive controller. The drawback of the approach lies in the finite horizon of the MPC. Thus the raceline is not truly optimal. Talvala et al. [26] developed a robust lookahead controller for lane keeping that when coupled with longitudinal control provides stability to an autonomous racing controller.

### IV. SIMULATOR AND HARDWARE

TUNERCAR includes a complete set of system components which are utilized to demonstrate the approach. Importantly, the target hardware, an open-source 1/10th scale autonomous

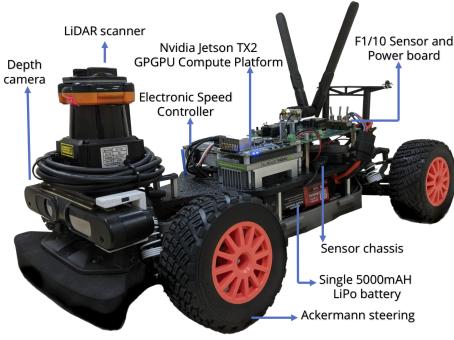


Fig. 2: Target 1/10th scale vehicle

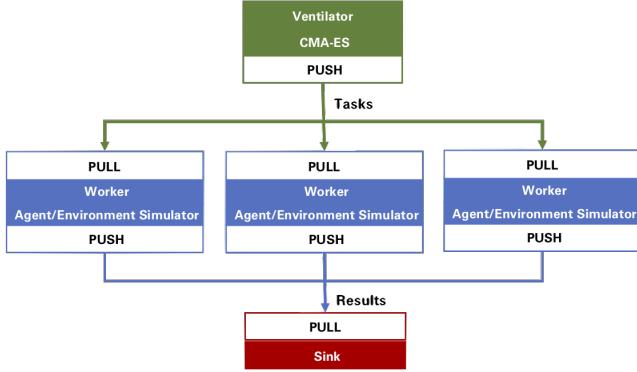


Fig. 3: MapReduce pattern of software stack.

vehicle created by the authors, is mapped to a validated, deterministic simulator capable of modeling both the dynamics and sensors included on the vehicle. The simulator also includes a wrapper which enables a distributed approach to optimization of the source program with low communication overhead. In addition, the toolchain provides performance oriented, abstracted implementations and interfaces (sketches) for the core algorithms of the source program.

#### A. Hardware

The F1/10 vehicle, shown in Fig. 2, is designed around a ready-to-run RC car chassis [27]. A power board (used to manage the onboard compute and sensors) as well as mounting plate design are provided; all aspects (hardware, mechanical design, software) of these additional parts are open source. All computation occurs on the onboard NVIDIA TX2, a modern embedded system on a chip (SoC) which contains a conventional multicore ARM CPU in addition to a power-efficient GPU [28]. The main sensor is a planar laser scanner (or LIDAR) which can capture range measurements. The LIDAR enables the vehicle to implement reactive obstacle avoidance strategies, estimate odometry, create maps, and localize. Due to the operating environment (typically corridors with few features), we supplement the LIDAR measurements with odometry information from the electronic speed controller [29]. An optional RGB-D camera provides additional sensing modalities, but is not used in these experiments.

#### B. Simulator

The simulator uses a lightweight 2D physics engine written in C++ which implements the single track vehicle model described in [30]. Parameter identification was performed to derive vehicle parameters: mass, center of mass moment of inertia, friction coefficient, cornering stiffness, and maximum acceleration/deceleration rates, Appendix A. The moment of inertia was estimated using the bifilar (two-wire) pendulum method (*e.g.* [31]). Tire parameters were found using the PAC2002 Magic-Formula model [32] and the cornering stiffness coefficient was back-calculated using [33]. A force scale was used to measure the kinetic friction coefficient  $\mu$  between the rubber tires and linoleum floor as the vehicle was dragged laterally at a constant velocity. In addition to modeling vehicle dynamics, the simulator detects collisions between the vehicle and obstacles in the environment in linear-time using a pre-computed lookup table of range measurements [34]. This method is also used to simulate the vehicle's LIDAR.

The simulator differs significantly from existing ROS-based tools. In order to create deterministic rollouts the C++ executable is wrapped in the OpenAI Gym [35] environment which explicitly steps time when all inputs have been computed. A further benefit of this approach is the ability to take advantage of faster than real-time execution; on commodity hardware, simulations proceed at approximately 6x real-time. Finally, the simulation environment provides a method for the members of the vehicle dynamics parameter class to be modified by TUNERCAR at the beginning of each rollout. Lastly, the simulation toolchain include a front end for generating random maps and pre-processing occupancy grid maps created using SLAM.

The TUNERCAR toolchain employs a MapReduce [36] messaging pattern implemented using ZeroMQ [37], a high-performance asynchronous messaging library. As shown in Fig. 3, many ‘workers’ are spawned in parallel, each equipped with a simulator, velocity optimizer, and full vehicle stack. Samples from the CMA-ES search node are then broadcast to a pool of worker nodes. On receipt of a new sample, the worker instantiates vehicle and environment parameters and simulates a rollout to compute the lap time. In order to update the searcher’s sampling distribution, the worker simply transmits the lap time and task index to a sink node which collects the worker results and synchronizes the search epochs.

#### C. Vehicle Software

Some algorithms are considered adaptable; others such as the simultaneous localization and mapping package (SLAM) are only used offline to create a model of the environment and, thus, are not tuned. The complete set of non-adaptable algorithms and software is as follows: Google Cartographer [38] for SLAM, a particle filter for pure localization [39], a behavior controller which manages communication between the planning nodes and the motor controller.

TUNERCAR compiles parameterized versions of two processes on the vehicle: a geometric path tracking controller

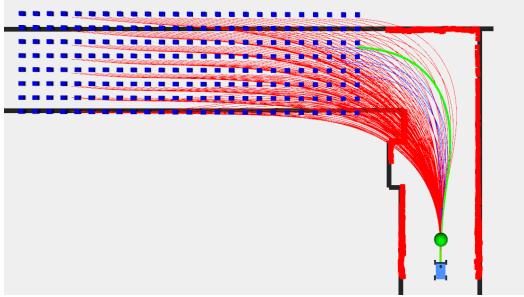


Fig. 4: Adjusting the horizon of the path planner shifts the lattice of samples (blue markers), the green marker represents the pure pursuit trackers selected lookahead distance.

based on pure pursuit [1], [40] and a path planning module which utilizes a sampling-based non-linear model predictive control [41], [42].

*1) Path Tracker:* The goal of path tracker is to compute steering inputs which allow the autonomous vehicle to follow a sequence of waypoints defined in the map frame. Performance of a path tracker is measured by the feasibility of the inputs, heading error, and lateral offset between the path and vehicle’s position [40]. We utilize the pure pursuit path tracker [1], a simple geometric method which has been shown to be effective if properly tuned for the expected operating conditions. In particular, the lookahead distance, which is used to select a point on the path ahead of the vehicle significantly affects the performance. Too small of a lookahead and the vehicle oscillates; too large and corners are cut. Thus, the lookahead distance of the path tracker computed as:  $d_{wl}d_{tl}$  in order to ensure that it does not exceed the path planning horizon, is included in the search space as a tunable parameter.

*2) Path Planner:* The goal of the path planner is to generate kinematically and dynamically feasible trajectories that can take the vehicle from its current pose to a sampled set of goal poses (see Fig. 4). Each trajectory is represented as a set of cubic spirals,  $p = [s, a, b, c, d]$  where  $s$  is the total arc length of the trajectory and  $(a, b, c, d)$  are equispaced knot points encoding the path curvature. For each trajectory, gradient descent is used to find spline parameters which minimize the error between the goal pose and the calculated end point given by a forward simulation of the vehicle dynamics. Real-time performance of the system is improved by creating a dense lookup table of precomputed goal, solution pairs as described in [43]. Thus, online, once a goal point has been chosen, all that remains is to sample  $N$  equidistant points corresponding to the spline parameters stored in the lookup-table and check them against an occupancy grid for feasibility. In order to adapt the method to the operational domain the horizon of the planner is adjusted to account for track geometry.

## V. CASE STUDY

CMA-ES requires only three hyperparameters:  $\epsilon$  the threshold of the L2 norm of covariance matrix for the sampling distribution,  $\lambda$  the population size, and  $\mu$  the quantile of

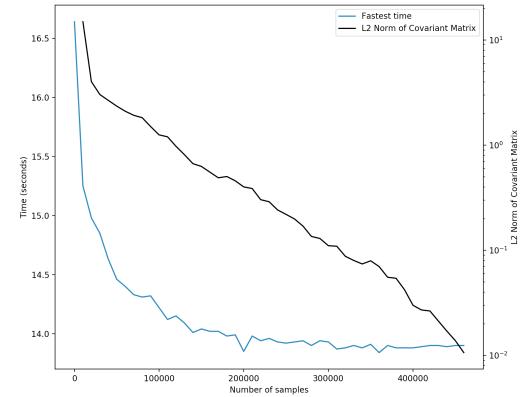


Fig. 5: Convergence on L2 Norm of Covariance Matrix

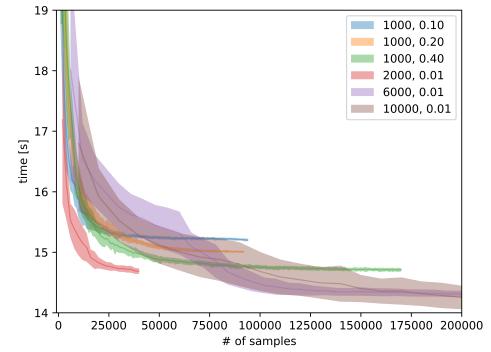


Fig. 6: Effect of hyperparameters on performance

samples to retain between epochs. Fig. 5 shows an experiment in which  $\epsilon$  is determined. Fig. 6 details the performance of the top 5% of the CMA-ES algorithm relative to the number of simulations performed for a variety of population sizes,  $\lambda$ , and quantile-thresholds,  $\mu$ . Based on the results of these experiments we select a  $\lambda$  of 10000, and  $\mu$  of 0.01 for best overall performance in terms of convergence rate and best lap time.

In order to further evaluate the CMA-ES implementation in TUNERCAR, we compare the top 100 laptimes relative to a naive-random-search for fixed computational budgets between 10,000 and 200,000 simulations. The results, shown in Fig. 7 show that our methodology converges to a set of solutions with lap times significantly lower than that of naive random sampling.

We designed three sets of experiment to evaluate the effectiveness of our toolchain. First, we conducted experiments to validate the results our methods using the F1/10th race cars detailed in Section IV. These experiments are conducted on the Levine Track map which is a challenging hallway loop due to the 90 degree turns and narrow driveable surface area. The list of physical car parameters to tune were limited to an easily modifiable subset that avoided major structural changes to the car’s components: total car mass and longitudinal location of the car’s center of mass. After the pipeline has computed an optimal trajectory and set of parameters, we physically modified the car’s parameters and deployed the tuned system,

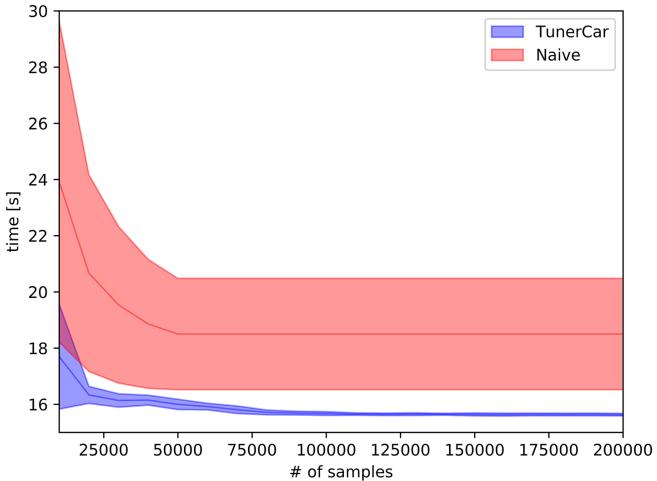


Fig. 7: Performance on Levine Map with fixed computation

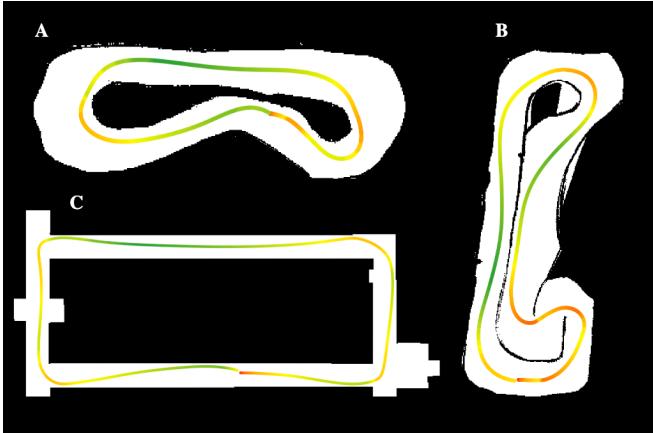


Fig. 8: Agent solutions on the track collection.

using the optimized waypoints and velocities.

Our second set of experiments were conducted primarily in simulation and compared against historical racing times recorded in the environments. We chose to run the optimization process on a set of three tracks (PORTO [44], TORINO [45], and PHILADELPHIA [46]) where we have hosted competitive racing events in order to benchmark the performance of TUNERCAR relative to hand-tuned solutions. Due to the variety of modifications various entrants utilized (*e.g.* springs, suspension, geometry and custom chassis), we do not restrict the set of physical parameters.

Lastly, we evaluate the reality gap in light of the proposed online optimization strategy by comparing vehicle performance in simulation to that which was achievable in reality. Fig. 7 details the performance of TUNERCAR relative to naive-random search given a fixed computational budget. Fig. 8 shows solutions found by TUNERCAR which exceed expert tuning-performance on (A) TORINO, (B) PORTO, and (C) PHILADELPHIA. Fig. 9 shows a comparison of the relative performance between top solutions deployed on the car and in the simulator.

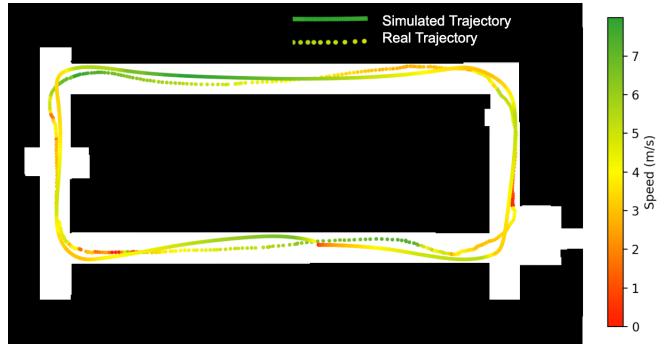


Fig. 9: Comparison of simulated vs. real performance

TABLE I: TUNERCAR vs. Expert-tuned Agents

Torino Track		
Method	Lap Time (s)	
TUNERCAR	17.96	
Follow The Gap	18.29	
Pure Pursuit	19.29	
Porto Track		
Method	Lap Time (s)	
TUNERCAR	8.11	
Follow The Gap	9.10	
Pure Pursuit	11.9	
Philadelphia Track		
Method	Lap Time (s)	Top Speed (m/s)
TUNERCAR	16.19	9.4967
Pure Pursuit	17.0	7.4532
Teleop	22.62	6.0

## VI. CONCLUSIONS

We introduce TUNERCAR, a superoptimization toolchain for jointly optimizing racing strategy, vehicle parameters, planning methods, and control algorithms for an autonomous racecar. We detail the target hardware, software, simulators, and systems infrastructure necessary for implementation. Our methodology deploys a modern evolution strategy onto a massively parallelized software stack that enables simulations to proceed 6P times faster than real-world rollouts, achieving 6% faster lap times compared a naive sampling approach given a fixed computational budget. We validate our solution on an AV software stack and 1/10th-scale open-source vehicle developed for this project [[f1tenth.org](http://f1tenth.org)].

While the method as described in this paper has not been rigorously tested in head-to-head racing, the online path-planning component is capable of navigating the vehicle in the presence of other agents. Future work should investigate the feasibility of deploying this methodology utilizing an expanded search space that encodes the behaviors of other agents. A clear limitation of the presentation of this work is a lack of comparisons to other black-box optimizations, we mitigate this deficiency by measuring the toolchains performance against a variety of hand-tuned solutions entered in F1/10 competitions. Given that the experimental results show that we can find significant performance improvements over the other competition entries it is important to explore whether other search strategies can create even faster agents.

## APPENDIX

### A.

#### Parameter Identification

Parameter identification was performed on the real world vehicle in order to match the simulation vehicle as close as possible. The parameters identified were mass, center of mass moment of inertia, friction coefficient, cornering stiffness, and maximum acceleration/deceleration rates.

*1) Mass and Center of Mass:* The vehicle was placed on scale and found to be  $3518.6 \pm 0.1\text{g}$ . The lithium polymer battery, whose weight is  $363 \pm 0.1\text{g}$ , was included on the vehicle when the mass measurement of the vehicle was taken.

The center of mass was estimated by balancing the vehicle on the edge of the ruler. For a wheelbase of  $0.317\text{m}$ , the distance from the center of mass to the front wheel center  $l_{cg,f}$  was found to be  $0.147 \pm 0.005\text{m}$  and the distance to the rear wheel center  $l_{cg,r}$  was  $0.147 \pm 0.005\text{m}$ .

*2) Moment of Inertia:* The moment of inertia of the vehicle was estimated using the bifilar (two-wire) pendulum method. This method is used to measure the moment of inertia of symmetric objects, such as airplanes [31], unmanned air vehicles [47], and tennis rackets [48]. The bifilar pendulum is a torsional pendulum that consists of the test object suspended by two thin parallel wires that are equidistant from the center of mass. A small moment is applied to the vehicle and the angular frequency  $\omega$  is found by recording the period of oscillation about the vertical axis that goes through the center of mass. The moment of inertia can then be calculated using the following equation:

$$I = \frac{mgd^2}{4L\omega^2} \quad (1)$$

where  $m$  is the mass of the vehicle,  $g$  is acceleration due to gravity,  $d$  is the distance between the wires,  $L$  is the length of the wires, and  $\omega$  is the angular frequency found above. Equation 1 is obtained from the nonlinear mathematical model of a bifilar pendulum and derivations can be found in [49]. 10 shows the setup of the vehicle used in this paper.

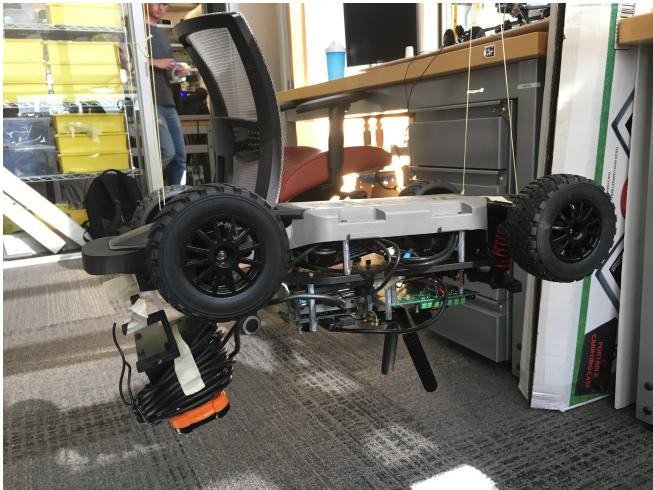


Fig. 10: Test setup of vehicle.

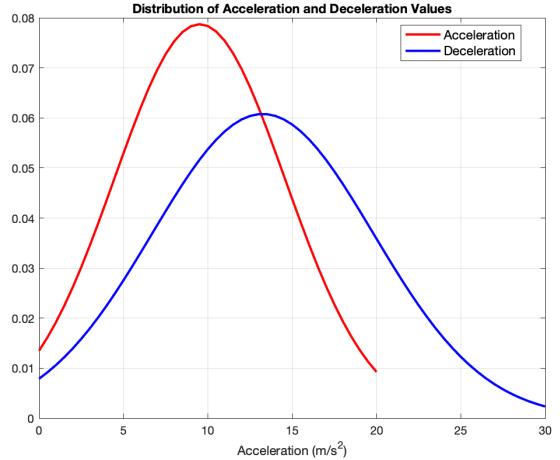


Fig. 11: Acceleration and Deceleration Curves

The moment of inertia of the vehicle was estimated using the bifilar pendulum method was found to be  $0.047\text{kg}\cdot\text{m}^2$  compared with the moment of inertia obtained from a CAD model of the vehicle was  $0.041\text{kg}\cdot\text{m}^2$ . The moment of inertia of the vehicle was estimated to be  $0.047\text{kg}\cdot\text{m}^2$ . The moment of inertia obtained from a CAD model of the vehicle was  $0.041\text{kg}\cdot\text{m}^2$ .

*3) Friction Coefficient and Cornering Stiffness:* The simulator is based on the single-track CommonRoad vehicle model [30]. Instead of using cornering stiffness  $C$ , as is conventionally used for single-track models, the CommonRoad model separates the effect of the friction coefficient  $\mu$ , the cornering stiffness coefficient  $C_s$ , and the vertical force  $F_z$  and relates them with  $C_i = \mu C_{S,i} F_{z,i}$  where  $i = \{f, r\}$  for the front and rear axle. This separation allows for the influence of weather to be modeled in friction coefficient. A force scale was used to measure the kinetic friction coefficient  $\mu$  between the rubber tires and linoleum floor as the vehicle was dragged laterally at a constant velocity.  $\mu$  was found to be  $0.5230 \pm 0.0014$ . The CommonRoad model used the PAC2002 Magic-Formula tire model developed by MSC Software [32] in order to find the tire parameters. For the purposes of this paper, the cornering stiffness coefficient was back-calculated by estimating the cornering stiffness using [33] as 16.5% of the weight on the tires.  $C_{S,f}$  was found to be  $4.191 \pm 0.002$  and  $C_{S,r}$  was found to be  $4.8469 \pm 0.002$ .

*4) Maximum Acceleration and Deceleration Rates:* Nineteen speed tests ranging from  $2.0\text{m/s}$  to  $11.0\text{m/s}$  in increasing increments of  $0.5\text{m/s}$  were performed. The vehicle was commanded to accelerate to a certain speed, maintain the speed for 0.5 second, and then brake. Position data from a particle filter on the vehicle and velocity data from the vehicle odometry were recorded for all of the tests. The acceleration and deceleration sections were fitted with separate exponential curves and the constant speed region was fitted with a linear curve. Finally, the maximum acceleration and deceleration for each test was extracted. It was also found that the commanded velocity was about  $0.5\text{m/s}$  faster than the measured speed obtained from the constant velocity region.

Figure 11 shows the distribution of the results obtained. The mean for the acceleration curve  $accel_{max}$  was  $9.514\text{m/s}^2$  while the mean for the deceleration curve  $decel_{max}$  was  $13.257\text{m/s}^2$ .

### B. Minimum Time Path

[MOK: Achin: details of convex optimization problem go here]

## REFERENCES

- [1] R. C. Coulter, “Implementation of the pure pursuit path tracking algorithm,” Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, Tech. Rep., 1992.
- [2] A. V. Aho, R. Sethi, and J. D. Ullman, “Compilers, principles, techniques,” *Addison wesley*, vol. 7, no. 8, p. 9, 1986.
- [3] C. A. Lattner, “Llvm: An infrastructure for multi-stage optimization,” Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2002.
- [4] E. Schkufza, R. Sharma, and A. Aiken, “Stochastic superoptimization,” in *ACM SIGPLAN Notices*, vol. 48, no. 4. ACM, 2013, pp. 305–316.
- [5] H. Massalin, “Superoptimizer: a look at the smallest program,” in *ACM SIGARCH Computer Architecture News*, vol. 15, no. 5. IEEE Computer Society Press, 1987, pp. 122–126.
- [6] A. Solar-Lezama, “Program sketching,” *International Journal on Software Tools for Technology Transfer*, vol. 15, no. 5-6, pp. 475–495, 2013.
- [7] P. Liang, M. I. Jordan, and D. Klein, “Learning programs: A hierarchical bayesian approach,” in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 639–646.
- [8] N. Vasilache, O. Zinenko, T. Theodoridis, P. Goyal, Z. DeVito, W. S. Moses, S. Verdoolaege, A. Adams, and A. Cohen, “Tensor comprehensions: Framework-agnostic high-performance machine learning abstractions,” *arXiv preprint arXiv:1802.04730*, 2018.
- [9] J. Ragan-Kelley, C. Barnes, A. Adams, S. Paris, F. Durand, and S. Amarasinghe, “Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines,” in *Acm Sigplan Notices*, vol. 48, no. 6. ACM, 2013, pp. 519–530.
- [10] T. Lipp and S. Boyd, “Minimum-time speed optimisation over a fixed path,” *International Journal of Control*, vol. 87, no. 6, pp. 1297–1311, 2014.
- [11] N. Hansen and A. Ostermeier, “Completely derandomized self-adaptation in evolution strategies,” *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.
- [12] N. Hansen, “The cma evolution strategy: A tutorial,” *arXiv preprint arXiv:1604.00772*, 2016.
- [13] N. Piterman, A. Pnueli, and Y. Sa’ar, “Synthesis of reactive (1) designs,” in *International Workshop on Verification, Model Checking, and Abstract Interpretation*. Springer, 2006, pp. 364–380.
- [14] T. Campos, J. P. Inala, A. Solar-Lezama, and H. Kress-Gazit, “Task-based design of ad-hoc modular manipulators,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6058–6064.
- [15] J. Liu, N. Ozay, U. Topcu, and R. M. Murray, “Synthesis of reactive switching protocols from temporal logic specifications,” *IEEE Transactions on Automatic Control*, vol. 58, no. 7, pp. 1771–1785, 2013.
- [16] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, “Discrete abstractions of hybrid systems,” *Proceedings of the IEEE*, vol. 88, no. 7, pp. 971–984, 2000.
- [17] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan *et al.*, “Population based training of neural networks,” *arXiv preprint arXiv:1711.09846*, 2017.
- [18] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, “Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding,” *ACM SIGEVolution*, vol. 7, no. 1, pp. 11–23, 2014.
- [19] D. Ha and J. Schmidhuber, “Recurrent world models facilitate policy evolution,” in *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2018, pp. 2451–2463, <https://worldmodels.github.io>. [Online]. Available: <https://papers.nips.cc/paper/7512-recurrent-world-models-facilitate-policy-evolution.pdf>
- [20] E. M. Kasprzak, K. E. Lewis, and D. L. Milliken, “Steady-state vehicle optimization using pareto-minimum analysis,” *SAE transactions*, pp. 2624–2631, 1998.
- [21] K. Hacker, K. Lewis, and E. M. Kasprzak, “Racecar optimization and tradeoff analysis in a parallel computing environment,” *SAE Technical Paper*, Tech. Rep., 2000.
- [22] F. Castellani and G. Franceschini, “Use of genetic algorithms as an innovative tool for race car design,” *SAE Technical Paper*, Tech. Rep., 2003.
- [23] K. Hayward, *Application of Evolutionary Algorithms to Engineering Design*. University of Western Australia, 2007.
- [24] N. R. Kapania, J. Subotsis, and J. C. Gerdes, “A sequential two-step algorithm for fast generation of vehicle racing trajectories,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 138, no. 9, p. 091005, 2016.
- [25] A. Liniger, A. Domahidi, and M. Morari, “Optimization-based autonomous racing of 1: 43 scale rc cars,” *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.
- [26] K. L. Talvala, K. Kritayakirana, and J. C. Gerdes, “Pushing the limits: From lanekeeping to autonomous racing,” *Annual Reviews in Control*, vol. 35, no. 1, pp. 137–148, 2011.
- [27] Traxxas, “Fiesta® st rally, 1/10 scale ford fiesta awd rally car”.
- [28] D. Frank, K. Kolotka, A. Phillips, M. Schulz, C. Rigney, A. Drown, R. Stricko, K. Harper, and R. Freuler, “Developing and improving a multi-element first-year engineering cornerstone autonomous robotics design project,” in *Proceedings of the 2017 American Society for Engineering Education Annual Conference*, 2017.
- [29] B. Vedder, “Vedder electronic speed controller.” [Online]. Available: <https://vesc-project.com/documentation>
- [30] M. Althoff, M. Koschi, and S. Manzinger, “Commonroad: Composable benchmarks for motion planning on roads,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 719–726.
- [31] H. A. Soule and M. P. Miller, “The experimental determination of the moments of inertia of airplanes,” 1934.
- [32] “Adams/tire help.” MSC Software, 2 MacArthur Place, Santa Ana, CA 92707, April 2011.
- [33] “Discussion: Cornering stiffness,” McHenry Software Manual, M-smac Input, 2002.
- [34] P. F. Felzenszwalb and D. P. Huttenlocher, “Distance transforms of sampled functions,” *Theory of computing*, vol. 8, no. 1, pp. 415–428, 2012.
- [35] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” 2016.
- [36] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
- [37] P. Hintjens, *ZeroMQ: messaging for many applications*. O’Reilly Media, Inc., 2013.
- [38] W. Hess, D. Kohler, H. Rapp, and D. Andor, “Real-time loop closure in 2d lidar slam,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1271–1278.
- [39] C. Walsh and S. Karaman, “Cddt: Fast approximate 2d ray casting for accelerated localization,” vol. abs/1705.01167, 2017. [Online]. Available: <http://arxiv.org/abs/1705.01167>
- [40] J. M. Snider *et al.*, “Automatic steering methods for autonomous automobile path tracking,” *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*, 2009.
- [41] T. M. Howard, *Adaptive model-predictive motion planning for navigation in complex environments*. Carnegie Mellon University, 2009.
- [42] M. McNaughton, “Parallel algorithms for real-time motion planning,” 2011.
- [43] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer *et al.*, “Autonomous driving in urban environments: Boss and the urban challenge,” *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.
- [44] R. Mangharam, M. Behl, H. Abbas, and M. O’Kelly, “2018 Portuguese Grand Prix,” in *F1/10 Workshop and Competition as part of the 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, 2018.
- [45] R. Mangharam, M. Behl, H. Abbas, M. Bertonga, and M. O’Kelly, “2018 Italian Grand Prix,” in *F1/10 Workshop and Competition as part of ESWeek 2018, Torino, Italy*, 2018.
- [46] H. Abbas, R. Mangharam, and M. O’Kelly, “Ese 680: Autonomous racing.” [Online]. Available: <http://web.engr.oregonstate.edu/~abbasho/papers/F110PennSyllabus.pdf>

- [47] M. R. Jardin and E. R. Mueller, "Optimized measurements of unmanned-air-vehicle mass moment of inertia with a bifilar pendulum," *Journal of Aircraft*, vol. 46, no. 3, pp. 763–775, 2009.
- [48] J. Spurr, S. Goodwill, J. Kelley, and S. Haake, "Measuring the inertial properties of a tennis racket," *Procedia Engineering*, vol. 72, pp. 569–574, 2014.
- [49] A. Kotikalpudi, B. Taylor, C. Moreno, H. Pfifer, and G. Balas, "Swing tests for estimation of moments of inertia," *Unpublished notes, University of Minnesota Dept. of Aerospace Engineering and Mechanics*, 2013.