Multi-Modal Model Predictive Control through Batch Non-Holonomic Trajectory Optimization: Application to Highway Driving

Vivek K. Adajania, Aditya Sharma, Anish Gupta, Houman Masnavi, K Madhava Krishna and Arun K.Singh

***Abstract*— Standard Model Predictive Control (MPC) or trajectory optimization approaches perform only a local search to solve a complex non-convex optimization problem. As a result, they cannot capture the multi-modal characteristic of human driving. A global optimizer can be a potential solution but is computationally intractable in a real-time setting. In this paper, we present a real-time MPC capable of searching over different driving modalities. Our basic idea is simple: we run several goal-directed parallel trajectory optimizations and score the resulting trajectories based on user-defined meta cost functions. This allows us to perform a search over several locally optimal motion plans. Although conceptually straightforward, realizing this idea in real-time with existing optimizers is highly challenging from technical and computational standpoints. With this motivation, we present a novel batch non-holonomic trajectory optimization whose underlying matrix algebra is easily parallelizable across problem instances and reduces to computing large batch matrix-vector products. This structure, in turn, is achieved by deriving a linearization-free multi-convex reformulation of the non-holonomic kinematics and collision avoidance constraints. We extensively validate our approach using both synthetic and real data sets (NGSIM) of traffic scenarios. We highlight how our algorithm automatically takes lane-change and overtaking decisions based on the defined meta cost function. Our batch optimizer achieves trajectories with lower meta cost, up to 6x faster than competing baselines.**

I. Introduction

Human driving is a complex mixture of discrete level decisions (merge, overtake, etc.) and lower-level motion commands [1]. If we adopt an optimization perspective, the multiple discrete decisions can be seen as local minima associated with the underlying non-convex trajectory optimization problem [1] [2]. Local optimizers based on Sequential Quadratic Programming (SQP) or Gradient Descent (GD) are not equipped to search over all the local minima. On the other hand, global optimization techniques like mixed-integer programming [2] offer a potential solution but are not particularly useful in a real-time setting, especially in dense traffic scenarios.

Main Idea: Let us define goals as a tuple of position, velocity and acceleration to be achieved by the ego-vehicle at the end of the planning horizon. Then, our approach in this paper is built on a simple insight that many different goal-directed trajectories can accomplish a given high-level driving task. For example, if the task is to drive close to maximum velocity, the autonomous car can choose its next goal to be in any lane. Some particular goal choices may require overtaking a slow-moving vehicle directly in front, while some may require safely merging with oncoming cars in a different lane. More concretely, each goal-directed trajectories may converge to a different local minima resulting in a multi-modal driving behavior (see pp-5 [1]). Thus our proposed work is based on the idea of running several parallel goal-directed trajectory optimization problems and ranking the resulting locally optimal trajectories based on a user-defined meta cost function. Although conceptually simple, we are not aware of any such approach in existing works. There are trajectory sampling approaches such as [3], [4] but they ignore collision avoidance and kinematic constraints while computing possible maneuvers. One possible reason the parallel/batch trajectory optimization approach has not been tried is that solving non-convex problem associated with autonomous driving is challenging. Running several instances of the problem in parallel only further increases the computational burden. The conceptually straightforward approach of running different optimizations in parallel CPU threads is not scaleable for a large batch size in a dense highway driving scenario (see Fig.2 and discussions around it). We present a possible solution whose main novelties can be summarized as follows.

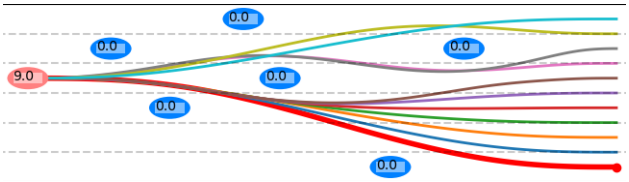

Fig. 1. The different colored samples represent the several locally optimal trajectories for driving as close as possible to the cruise speed (meta cost). The ego-vehicle is shown in red while the neighboring obstacles are shown in blue. The number within the ellipses represent the velocities of the respective entity. Unlike [3], [4], we explicitly consider collision avoidance and kinematic constraints while generating candidate trajectories. The trajectory shown in bold red achieves the best performance (lowest meta-cost). We recommend seeing the accompanying video before reading the paper.

Algorithmic: We present the very first batch non-holonomic trajectory optimizer for parallely generating several goal-directed locally optimal collision-free trajectories in real-time. The core algorithmic challenge lies in achieving linear scalability with respect to the number of batches. As shown in Fig.2 we cannot achieve such scalability by simply running each optimization problem in a separate CPU thread. Instead, we need to efficiently parallelize the per-iteration computation of the batch optimizer. We address the scalability issue by developing an batch optimizer wherein simultaneously iterating over different problem instances boils down to multi-

The first three authors are with the Robotics Research Center, Kohli Center for Intelligent Systems, IIIT Hyderabad, India and Houman Masnavi and Arun Singh are with the Institute of Technology, University of Tartu. The work was supported in part by the European Social Fund through IT Academy program in Estonia, smart specialization project with BOLT and grants COVSG24 and PSG605 from Estonian Research Council.

TABLE I
IMPORTANT SYMBOLS

$x_l(t), y_l(t), \psi_l(t)$	Position and heading of the ego-vehicle at time t.
$\xi_{xj}(t), \xi_{yj}(t)$	Position of the j^{th} obstacle at time t
$\alpha_{j,l}(t), d_{j,l}(t)$	Variables associated with our collision avoidance model. Refer to text for details.

plying a single constant matrix with a set of vectors. We show that the heavily vectorized structure of our optimizer stems naturally from two key algorithmic developments. First, we adopt a linearization-free multi-convex reformulation of the kinematic and collision avoidance constraints. Second, we apply the Alternating Minimization (AM) technique to solve the reformulated problem. We rank the output of batch optimization with some simple meta cost functions that model the higher-level driving objectives (e.g., driving with high-speed). We show that the ranking mechanism coupled with an intelligent goal-sampling approach automatically leads to discovering lane-change, vehicle following, overtaking maneuvers based on the traffic scenario. Refer Section II-B for a summary of algorithmic contribution over the author's prior work.

Applied: We provide an open source implementation [5] for review and to promote further research in this direction.

State-of-the-art Performance: We compare our batch optimizer based MPC with three strong baselines and show that we achieve better solutions (in terms of meta-cost value) while being up to 6x faster. Our first baseline is a standard MPC that computes just a single locally optimal trajectory. Our second baseline is batch multi-threaded implementation of optimal control solver ACADO [6]. Our final baseline is based on trajectory sampling in the Frenet frame [3].

II. PRELIMINARIES AND RELATED WORK

A. Symbols and Notations

Small-case normal and bold font letters will be used to denote scalars and vectors respectively. Bold-font upper-case letters will represent matrix. The superscript T will denote transpose of a matrix and vector. Some of the main symbols are summarized in Table I while some are also defined in their first place of use.

B. Batch Non-Holonomic Trajectory Optimization

We are interested in solving l non-holonomic trajectory optimizations in parallel each of which can be formulated in the following manner. The resulting trajectories from the parallel problems will be ranked based on a meta cost function discussed later.

$$\min \sum_t \ddot{x}_l(t)^2 + \ddot{y}_l(t)^2 + \ddot{\psi}_l(t)^2 \tag{1a}$$

$$\dot{x}_l(t) = v_l(t)\cos\psi_l(t), \dot{y}_l(t) = v_l(t)\sin\psi_l(t), \forall t \tag{1b}$$

$$(x_l(t), y_l(t), \psi_l(t)) \in \mathcal{C}_{b,l} \tag{1c}$$

$$v_{min} \leq v_l(t) \leq v_{max}, \sqrt{\ddot{x}_l(t)^2 + \ddot{y}_l(t)^2} \leq a_{max} \tag{1d}$$

$$-\frac{(x_l(t) - \xi_{xj}(t))^2}{a^2} - \frac{(y_l(t) - \xi_{yj}(t))^2}{b^2} + 1 \leq 0, \tag{1e}$$

The subscript l denotes that the specific variable belongs to the l^{th} instance of the problem in the batch. The variables of the trajectory optimization are $(x_l(t), y_l(t), \psi_l(t), v_l(t))$. The cost function minimizes the squared acceleration value for the linear and angular motions. The equality constraints (1b) stems from the non-holonomic kinematics of the car. Constraints (1c) ensures the boundary conditions on the position, heading angle and their derivatives. The inequalities (1d) represent the bounds on the forward velocities and total acceleration. The set of constraints (1e) enforces the collision avoidance between the ego and the neighboring vehicles with the assumption that both are represented by an axis-aligned ellipse. For the ease of exposition and without loss of generality, we assume that every obstacle ellipse has the same major (a) and minor axis (b) dimension. The (a, b) includes the inflation to account for the size of the ego-vehicle. The ellipse of the ego-vehicle and obstacle will not overlap as long the minimum separation distance is greater than $\sqrt{a^2 + b^2}$. It is worth pointing out that a, b will be larger than the length and width of the car. In other words, there will be some over-approximation of the ego-vehicle footprint. To keep this value limited, we enforce some restriction on the heading of the ego-vehicle. We discuss this more in the beginning of Section III.

The typical control inputs for the ego-vehicle are the acceleration and the steering inputs. The former can be obtained by the derivative or finite difference of $v(t)$. The steering angle is given by $\arctan(\frac{\dot{\psi}(t)*h}{v(t)})$, where h is the inter-axle distance. The control computation is a direct consequence of the differential flatness property of a non-holonomic car-like vehicle which allows the control inputs to be represented as a function of states and their derivatives.

Note: All l instances of the above optimization problem have the same velocity and acceleration bounds and operate around the same set of obstacles.

Existing Works: Trajectory optimizations of the form (1a)-(1e) are typically solved through approaches like SQP [7], GD [8] etc. Authors in [2] proposed a global optimization approach but considered the ego vehicle as a holonomic triple-integrator system. They derived some approximations for the non-holonomic constraints that hold more naturally at high forward velocities. In contrast, [9] adopt a more rigorous approach but the resulting algorithm was tested on environments sparsely filled with obstacles. Authors in [10] used an approach similar to [2] to compute the best driving modality and then refined the solution further through local optimization (e.g SQP) to handle kinematic constraints.

Improvements over our prior efforts: Our approach extends recent work [11] to batch setting and further applies it to highway driving. Specifically, we propose the core batch solution update rule (Eqn.(18), (20)) by leveraging the implicit structures in the matrix algebra of [11]. Furthermore, unlike our current work, [11] did not consider acceleration bounds. Typically, constraints on acceleration are modeled as affine inequalities. However, such representation is not suitable for our formulation that relies on reducing the

trajectory optimization to a sequence of unconstrained QPs to obtain an efficient batch update rule. We thus reformulate the acceleration bounds in the same form as the collision avoidance constraints of [11]. Our current work is also an improvement over [12] that handled collision avoidance constraints through a novel linearization approach. But as discussed in Section III-A, any linearization of the underlying costs and constraints substantially complicates the parallelization of the batch optimization.

III. Main Results

This section derives our main algorithmic results. We begin by summarizing our main assumptions.

1.Road Attached Frame: We assume that our problem set-up (i.e optimization (1a)-(1e)) is defined in the reference frame of the center-line of the road [10]. This allows us to essentially treat curved roads as ones with a straight-line geometry. The non-holonomic constraints are defined in the road attached frame and holds true when the trajectories are reverted back to global frame [10].

2. Restricted Heading Change: We consider trajectories with minimal heading change (+-13 deg) with respect to the center-line. The said heading change is realistic in case of typical highway (and even residential) driving scenarios [13]. The assumption on the heading is made so that we can model the ego and neighboring vehicles as axis-aligned ellipses and use the collision avoidance model defined in (1e). Note that the formulation (1a)-(1e) and specifically the non-holonomic constraints are general and can generate trajectories with arbitrary heading changes. We just enforce our design choice by discarding trajectories with larger heading changes..

3. Trajectory Prediction: The batch optimization generates trajectories that are feasible with respect to the constant-velocity prediction of the trajectories of the dynamic obstacles (neighboring cars). That is, $\xi_{x,j}(t), \xi_{y,j}(t)$ are obtained by linearly interpolating the positions with the current velocity. We adopt such a minimalist representation to test the full potential of our batch optimization, specifically how its fast re-planning ensures safety in the absence of any complex trajectory forecasting algorithms.

A. Intuition from Parallel Least Squares

Consider the following l linear least squares problem

$$\min_{\mathbf{s}_l} \|\mathbf{F}\mathbf{s}_l - \mathbf{g}_l\|_2, \forall l = 1, 2....l \tag{2}$$

All problem instances share the same matrix $\mathbf{F}$ but have different vector $\mathbf{g}_l$. A simple way to solve the problem would be first to compute the inverse of $\mathbf{F}^T\mathbf{F}$, then calculate $\mathbf{F}^T\mathbf{g}_l$ and finally multiply both the entities together. Notably, the last two operations are matrix-vector products that can be trivially parallelized. The expensive inverse needs to be done only **once**.

Now, contrast the above with the following non-linear least squares set-up

$$\min_{\mathbf{s}_l} \|\mathbf{f}(\mathbf{s}_l)\|_2^2 \approx \min_{\mathbf{s}_l} \|\mathbf{F}_l\mathbf{s}_l - \mathbf{g}_l\|_2 \tag{3}$$

The solution process begins by computing the Jacobian of $\mathbf{f}$ around a given guess solutions $\mathbf{s}_l$ to obtain a linear least squares approximation. Importantly, since each $\mathbf{s}_l$ would be different, the matrix $\mathbf{F}_l$ will vary across problem instances. Thus, computing the solution would first require forming $\mathbf{F}_l^T\mathbf{F}_l$ followed by computing inverses (or just factorization) of each of these. Furthermore, these computations need to be refined at each iteration of the non-linear least squares.

The above example illustrates the relative difficulty of parallelizing the per-iteration computations of a non-linear least squares problem compared to the special linear setting presented in (2). Many trajectory optimizations are indeed formulated as non-linear least-squares [14], and thus, they inherit the same bottlenecks discussed above towards parallelization. We note that in the non-linear setting, instead of parallelizing per-iteration operations, it will be more reasonable to solve each non-linear least squares in full in parallel CPU threads. However, such parallelization approach has scalability issues (limited by number of CPU cores) and requires careful synchronization of different CPU threads.

The core feature of our batch optimizer is that its most computationally heavy part has the same structure as the linear least-squares set-up of (2). Thus our parallelization effort essentially reduces to computing batch matrix-vector products.

B. Building Blocks

Reformulating the Collision Avoidance Constraints: We rephrase the quadratic collision avoidance constraints (1e) into the form $\mathbf{f}_{c,l} = \mathbf{0}$ based on our prior works [11].

$$\mathbf{f}_{c,l} = \left\{ \begin{array}{l} x_l(t) - \xi_{xj}(t) - ad_{j,l}(t)\cos\alpha_{j,l}(t), \forall j, t \\ y_l(t) - \xi_{yj}(t) - bd_{j,l}(t)\sin\alpha_{j,l}(t), \forall j, t \end{array} \right\} \tag{4}$$

As evident, $\mathbf{f}_{c,l}$ is a polar representation of the euclidean distance between the robot and the obstacle with the variables $\alpha_{j,l}(t)$ being the angle of the line of sight vector connecting the ego-vehicle and the j^{th} obstacle. The variable $d_{j,l}(t)$ is the ratio of the length of the line-of-sight vector to minimum separation distance required for collision avoidance. Note that these variables will be different for each problem in the batch and the additional subscript l has been introduced to represent that fact. Collision avoidance can be enforced by ensuring $d_{j,l}(t) \geq 1$. It should be noted that in (4), $\alpha_{j,l}(t), d_{j,l}(t)$ are unknown variables that are obtained by our optimizer along with other trajectory variables.

Reformulating Acceleration Bounds: Typically, the quadratic acceleration bounds in (1d) are split into separate affine constraints along each motion direction. However, we reformulate it in the same manner as collision avoidance constraints. That is we have constraints of the form $\mathbf{f}_{a,l} = \mathbf{0}$, where

$$\mathbf{f}_{a,l} = \left\{ \begin{array}{l} \ddot{x}_l(t) - d_{a,l}(t)\cos\alpha_{a,l}(t) \\ \ddot{y}_l(t) - d_{a,l}(t)\sin\alpha_{a,l}(t) \end{array} \right\}, , d_{a,l}(t) \leq a_{max} \tag{5}$$

The variables $\alpha_{a,l}(t), d_{a,l}$ are unknown and will be computed by our optimizer. On the surface, our representation of acceleration bounds seem more complicated but as shown

later, is essential achieve appropriate computational structure in our batch optimizer.

Trajectory Parameterization The trajectory variables in each of the l instantiations of the problem can be represented in the following manner.

$$\big[x(t_1), x(t_2), \ldots, x(t_n)\big] = \mathbf{P}\mathbf{c}_x, \big[\psi(t_1), \psi(t_2), \ldots, \psi(t_n)\big] = \mathbf{P}\mathbf{c}_\psi \tag{6}$$

where, $\mathbf{P}$ is a matrix formed with time-dependent basis functions (e.g polynomials) and $\mathbf{c}_{x,l}, \mathbf{c}_{\psi,l}$ are the coefficients associated with the basis functions. Similar expressions can be written for $y_l(t)$ as well. We can also express the derivatives in terms of $\dot{\mathbf{P}}, \ddot{\mathbf{P}}$.

Matrix Representation: Using the trajectory parametrization presented in (6), we can put constraints (4) and (5) and the non-holonomic constraints (1b) in the matrix form of (7a), (7b), (7c) respectively.

$$\mathbf{F}_o\mathbf{c}_{x,l} = \boldsymbol{\xi}_x + a\mathbf{d}_l \cos\boldsymbol{\alpha}_l, \mathbf{F}_o\mathbf{c}_{y,l} = \boldsymbol{\xi}_y + b\mathbf{d}_l \sin\boldsymbol{\alpha}_l \tag{7a}$$

$$\ddot{\mathbf{P}}\mathbf{c}_{x,l} = \mathbf{d}_{a,l}\cos\boldsymbol{\alpha}_a, \ddot{\mathbf{P}}\mathbf{c}_{y,l} = \mathbf{d}_{a,l}\sin\boldsymbol{\alpha}_a, \tag{7b}$$

$$\dot{\mathbf{P}}\mathbf{c}_{x,l} = \mathbf{v}\cos\mathbf{P}\mathbf{c}_{\boldsymbol{\psi},\boldsymbol{l}}, \dot{\mathbf{P}}\mathbf{c}_{y,l} = \mathbf{v}\sin\mathbf{P}\mathbf{c}_{\boldsymbol{\psi},\boldsymbol{l}} \tag{7c}$$

The matrix $\mathbf{F}_o$ is obtained by stacking the matrix $\mathbf{P}$ from (6) m times (the number of obstacles in the environment). The vector $\boldsymbol{\xi}_x, \boldsymbol{\xi}_y$ is formed by appropriately stacking $\xi_{xj}(t), \xi_{yj}(t)$ at different time instants and for all the obstacles. Similar construction is followed to obtain $\boldsymbol{\alpha}_l, \mathbf{d}_l, \boldsymbol{d}_{a,l}, \boldsymbol{\psi}_l, \mathbf{v}_l$.

C. Multi-Convex Reformulation

Using previous derivations, we substitute (1a)-(1e) by the following:

$$\min_{v_{min}\leq\mathbf{v}_l\leq\mathbf{v}_{max}} \frac{1}{2}\mathbf{c}_{x,l}^T\mathbf{Q}\mathbf{c}_{x,l} + \frac{1}{2}\mathbf{c}_{y,l}^T\mathbf{Q}\mathbf{c}_{y,l} + \frac{1}{2}\mathbf{c}_{\psi,l}^T\mathbf{Q}\mathbf{c}_{\psi,l} \tag{8a}$$

$$\mathbf{A}\begin{bmatrix}\mathbf{c}_{x,l}\\ \mathbf{c}_{y,l}\end{bmatrix} = \mathbf{b}_l, \mathbf{A}\mathbf{c}_{\psi,l} = \mathbf{b}_{\psi,l} \tag{8b}$$

$$\mathbf{F}\begin{bmatrix}\mathbf{c}_{x,l}\\ \mathbf{c}_{y,l}\end{bmatrix} = \mathbf{g}_l(\mathbf{c}_{\psi,l}\boldsymbol{\alpha}_l, \boldsymbol{\alpha}_{a,l}, \mathbf{d}_l, \mathbf{d}_{a,l}) \tag{8c}$$

$$\mathbf{F} = \begin{bmatrix}\begin{bmatrix}\mathbf{F}_o\\ \ddot{\mathbf{P}}\\ \dot{\mathbf{P}}\end{bmatrix} & \mathbf{0}\\ \mathbf{0} & \begin{bmatrix}\mathbf{F}_o\\ \ddot{\mathbf{P}}\\ \dot{\mathbf{P}}\end{bmatrix}\end{bmatrix}, \mathbf{g}_l = \begin{bmatrix}\boldsymbol{\xi}_x + a\mathbf{d}_l\cos\boldsymbol{\alpha}_l\\ \mathbf{d}_{a,l}\cos\boldsymbol{\alpha}_{a,l}\\ \mathbf{v}_l\cos\mathbf{P}\mathbf{c}_{\psi,l}\\ \boldsymbol{\xi}_y + b\mathbf{d}_l\sin\boldsymbol{\alpha}_l\\ \mathbf{d}_{a,l}\sin\boldsymbol{\alpha}_{a,l}\\ \mathbf{v}_l\sin\mathbf{P}\mathbf{c}_{\psi,l}\end{bmatrix}, \tag{9}$$

The cost function is a matrix representation of the sum of squared acceleration term in (1a). The equality constraints (8b) are the matrix representation of the boundary constraints (1c). We stack all the non-convex equality constraints in (8c).

Remark 1. *The matrices $(\boldsymbol{Q}, \boldsymbol{F}, \boldsymbol{F}_o, \boldsymbol{P}, \dot{\boldsymbol{P}}, \ddot{\boldsymbol{P}})$ in the reformulated problem (8a)-(8c) do not depend on the batch index l. In other words, they are the same for all the problem instantiations.*

Remark 1 highlights the motivation behind choosing the specific representation of the collision avoidance (4) and acceleration bounds (5).

D. Solution by Alternating Minimization

We solve (8a)-(8c) by relaxing the non-convex equality constraints (8c) as l_2 penalties and augmenting them into the cost function.

$$f_{xy}(\mathbf{c}_{x,l}, \mathbf{c}_{y,l}, \boldsymbol{\lambda}_{x,l}, \boldsymbol{\lambda}_{y,l}) + f_\psi(\mathbf{c}_{\psi,l}, \boldsymbol{\lambda}_{\psi,l}) + \frac{\rho_{xy}}{2}\left\|\mathbf{F}\begin{bmatrix}\mathbf{c}_{x,l}\\ \mathbf{c}_{y,l}\end{bmatrix} - \mathbf{g}_l\right\|_2^2 \tag{10}$$

$$f_{xy} = \frac{1}{2}\mathbf{c}_{x,l}^T\mathbf{Q}\mathbf{c}_{x,l} + \frac{1}{2}\mathbf{c}_{y,l}^T\mathbf{Q}\mathbf{c}_{y,l} - \langle\boldsymbol{\lambda}_{x,l}, \mathbf{c}_{x,l}\rangle - \langle\boldsymbol{\lambda}_{y,l}, \mathbf{c}_{y,l}\rangle \tag{11a}$$

$$f_\psi = \frac{1}{2}\mathbf{c}_{\psi,l}^T\mathbf{Q}\mathbf{c}_{\psi,l} - \langle\boldsymbol{\lambda}_{\psi,l}, \mathbf{c}_{\psi,l}\rangle \tag{11b}$$

Note the introduction of so-called Lagrange multipliers $\boldsymbol{\lambda}_{x,l}$, $\boldsymbol{\lambda}_{y,l}$, and $\boldsymbol{\lambda}_{\psi,l}$ that play a crucial role in driving the residuals of the equality constraints to zero [15].

Remark 2. *The augmented cost function (10) is convex in $(\boldsymbol{c}_{x,l}, \boldsymbol{c}_{y,l})$ for a given $\boldsymbol{c}_{\psi,l}, \boldsymbol{\alpha}_l, \boldsymbol{\alpha}_{a,l}, \boldsymbol{d}_l, \boldsymbol{d}_{a,l}$. Similarly, it is convex in $\boldsymbol{d}_l, \boldsymbol{d}_{a,l}$ for a given $\boldsymbol{c}_{\psi,l}, \boldsymbol{\alpha}_l, \boldsymbol{\alpha}_{a,l}, \boldsymbol{c}_{x,l}, \boldsymbol{c}_{y,l}$*

Remark 3. *For a given $(\boldsymbol{c}_{x,l}, \boldsymbol{c}_{y,l})$, the cost function (10) is non-convex in $\boldsymbol{c}_{\psi,l}$ but can be replaced with a simple convex surrogate from [12].*

Remark 4. *For a given $(\boldsymbol{c}_{x,l}, \boldsymbol{c}_{y,l})$, the optimizations over variables $(\boldsymbol{\alpha}_l, \boldsymbol{\alpha}_{a,l})$ have a simple closed form solution.*

Remark 5. *The augmented Lagrangian based reformulation of the non-convex constraints ensures that our batch optimization is always feasible. As a result, it can handle infeasible (e.g with respect to collision avoidance) trajectory initialization.*

Remarks 2 and 3 are precisely the multi-convex structure foreshadowed in the earlier sections. Moreover, remarks 2-4 highlight why an AM approach would be suitable: by decomposing the optimization process over separate blocks of variable, we can leverage the implicit convex structures present in the problem. The use of Augmented Lagrangian cost (10) in combination with AM procedure is known as the split-Bregman technique [15].

The different steps of AM are presented in (12)-(16), wherein the left superscript k is used to track the value of the variable over different iterations. For example, ${}^k\mathbf{c}_{x,l}$ represents the value at iteration k of this specific variable. At each optimization block, only few specific variables are optimized while the rest are kept fixed at the values obtained in the previous iteration or the previous step of the same iteration.

E. Analysis

Step (12) This optimization is a convex equality constrained QP that reduces to solving the following set of linear equations, wherein $\boldsymbol{\mu}_{x,l}, \boldsymbol{\mu}_{y,l}$ are the dual variable associated with the equality constraints.

$$\overbrace{\begin{bmatrix}\mathbf{Q}+\mathbf{F}^T\mathbf{F} & \mathbf{A}^T\\ \mathbf{A} & \mathbf{0}\end{bmatrix}}^{\mathbf{Q}_{xy}}\begin{bmatrix}\mathbf{c}_{x,l}\\ \mathbf{c}_{y,l}\\ \boldsymbol{\mu}_{x,l}\\ \boldsymbol{\mu}_{y,l}\end{bmatrix} = \overbrace{\begin{bmatrix}\mathbf{F}^{T\,k}\mathbf{g}_l + \begin{bmatrix}{}^k\boldsymbol{\lambda}_{x,l}\\ {}^k\boldsymbol{\lambda}_{y,l}\end{bmatrix}\\ \mathbf{b}_l\end{bmatrix}}^{\mathbf{q}_l} \tag{17}$$

Algorithm 1 Alternating Minimization based Batch Non-Holonomic Trajectory Optimization

$$^{k+1}(\mathbf{c}_{x,l},\mathbf{c}_{y,l}) = \arg\min_{\mathbf{c}_{x,l},\mathbf{c}_{y,l}} f(\mathbf{c}_x,\mathbf{c}_y,{}^k\boldsymbol{\lambda}_{x,l},{}^k\boldsymbol{\lambda}_{y,l}) + \frac{\rho_{xy}}{2}\left\|\mathbf{F}\begin{bmatrix}\mathbf{c}_{x,l}\\ \mathbf{c}_{y,l}\end{bmatrix} - \mathbf{g}({}^k\boldsymbol{\alpha}_l,{}^k\boldsymbol{\alpha}_{a,l},{}^k\mathbf{c}_{\psi,l},{}^k\mathbf{d}_l,{}^k\mathbf{d}_{a,l})\right\|_2^2, \mathbf{A}\begin{bmatrix}\mathbf{c}_{x,l}\\ \mathbf{c}_{y,l}\end{bmatrix} = \mathbf{b}_l \quad (12)$$

$$\begin{aligned}^{k+1}\mathbf{c}_\psi = \arg\min_{\mathbf{A}\mathbf{c}_{\psi,l}=\mathbf{b}_{\psi,l}} f(\mathbf{c}_\psi) + \frac{\rho_{xy}}{2}\|\mathbf{F}\begin{bmatrix}^{k+1}\mathbf{c}_{x,l}\\ ^{k+1}\mathbf{c}_{y,l}\end{bmatrix} - \mathbf{g}(\mathbf{c}_{\psi,l})\|_2^2 &= \arg\min_{\mathbf{A}\mathbf{c}_{\psi,l}=\mathbf{b}_{\psi,l}} f(\mathbf{c}_\psi) + \frac{\rho_{xy}}{2}\left\|\begin{matrix}^{k+1}\dot{\mathbf{x}}_l - \mathbf{v}_l\cos\mathbf{P}\mathbf{c}_{\psi,l}\\ ^{k+1}\dot{\mathbf{y}}_l - \mathbf{v}_l\sin\mathbf{P}\mathbf{c}_{\psi,l}\end{matrix}\right\|_2^2\\ &= \arg\min_{\mathbf{A}\mathbf{c}_{\psi,l}=\mathbf{b}_{\psi,l}} f(\mathbf{c}_\psi) + \frac{\rho_{xy}}{2}\|\arctan 2(^{k+1}\dot{\mathbf{y}}_l,{}^{k+1}\dot{\mathbf{x}}_l) - \mathbf{P}\mathbf{c}_{\psi,l}\|_2^2 \quad (13)\end{aligned}$$

$$^{k+1}\mathbf{v}_l = \arg\min_{v_{min}\le\mathbf{v}_l\le\mathbf{v}_{max}}\left\|\mathbf{F}\begin{bmatrix}^{k+1}\mathbf{c}_{x,l}\\ ^{k+1}\mathbf{c}_{y,l}\end{bmatrix} - \mathbf{g}(^{k+1}\mathbf{c}_{\psi,l},\mathbf{v}_l)\right\|_2^2 = \arg\min_{v_{min}\le\mathbf{v}\le\mathbf{v}_{max}}\left\|\begin{matrix}^{k+1}\dot{\mathbf{x}}_l - \mathbf{v}_l\cos\boldsymbol{\psi}_l\\ ^{k+1}\dot{\mathbf{y}}_l - \mathbf{v}_l\sin\boldsymbol{\psi}_l\end{matrix}\right\|_2^2 \quad (14)$$

$$^{k+1}\boldsymbol{\alpha}_l = \arg\min_{\boldsymbol{\alpha}_l}\left\|\mathbf{F}\begin{bmatrix}^{k+1}\mathbf{c}_{x,l}\\ ^{k+1}\mathbf{c}_{y,l}\end{bmatrix} - \mathbf{g}(^k\mathbf{d}_l)\right\|_2^2 = \left\|\begin{matrix}^{k+1}\mathbf{x} - \boldsymbol{\xi}_x - a\ {}^k\mathbf{d}_l\cos\boldsymbol{\alpha}_l\\ ^{k+1}\mathbf{y} - \boldsymbol{\xi}_y - b\ {}^k\mathbf{d}_l\sin\boldsymbol{\alpha}_l\end{matrix}\right\|_2^2, {}^{k+1}\mathbf{d}_l = \arg\min_{\mathbf{d}_l\ge 1}\left\|\mathbf{F}\begin{bmatrix}^{k+1}\mathbf{c}_{x,l}\\ ^{k+1}\mathbf{c}_{y,l}\end{bmatrix} - \mathbf{g}(^{k+1}\boldsymbol{\alpha}_l)\right\|_2^2 \quad (15)$$

$$^{k+1}\boldsymbol{\alpha}_{a,l} = \arg\min_{\boldsymbol{\alpha}_{a,l}}\left\|\mathbf{F}\begin{bmatrix}^{k+1}\mathbf{c}_{x,l}\\ ^{k+1}\mathbf{c}_{y,l}\end{bmatrix} - \mathbf{g}(^k\mathbf{d}_{a,l},\boldsymbol{\alpha}_{a,l})\right\|_2^2, {}^{k+1}\mathbf{d}_{a,l} = \arg\min_{\mathbf{d}_{a,l}\ge a_{max}}\left\|\mathbf{F}^{k+1}\begin{bmatrix}\mathbf{c}_{x,l}\\ \mathbf{c}_{y,l}\end{bmatrix} - \mathbf{g}(^{k+1}\boldsymbol{\alpha}_{a,l},\mathbf{d}_{a,l})\right\|_2^2 \quad (16)$$

The set of equations (17) computes the solution for the l^{th} instance of the problem. However, since the left hand side of (17) does not depend on the batch index l, we can compute the solution of the entire batch in one-shot through (18).

$$\begin{bmatrix}\mathbf{c}_{x,1},\mathbf{c}_{y,1},\boldsymbol{\mu}_{x,1},\boldsymbol{\mu}_{y,1}\\ \vdots\\ \mathbf{c}_{x,l},\mathbf{c}_{y,l},\boldsymbol{\mu}_{x,l},\boldsymbol{\mu}_{y,l}\end{bmatrix} = \left(\mathbf{Q}_{xy}^{-1}\left[\mathbf{q}_1|\mathbf{q}_2|\mathbf{q}_1\cdots\mathbf{q}_l\right]\right)^T \quad (18)$$

The major computation cost of (18) stems from obtaining different $\mathbf{F}^k\mathbf{g}_l$. But it is straightforward to formulate this operation as one large matrix-vector product and subsequently parallelize its computation.

Step (13) As mentioned earlier, optimization over $(\mathbf{c}_{\psi,l})$ is non-convex due to the presence of the non-holonomic penalty (second term). However, as shown in the last line of (13), for a given $(^{k+1}\dot{\mathbf{x}}_l{}^{k+1}\dot{\mathbf{y}}_l)$, the non-convex term can be replaced with a convex surrogate over $\mathbf{c}_{\psi,l}$, thereby reducing our problem to an equality-constrained QP. The solution process boils down to solving following set of linear equations

$$\overbrace{\begin{bmatrix}\mathbf{Q}+\mathbf{P}^T\mathbf{P} & \mathbf{A}^T\\ \mathbf{A} & \mathbf{0}\end{bmatrix}}^{\mathbf{Q}_\psi}\begin{bmatrix}\mathbf{c}_{\psi,l}\\ \boldsymbol{\mu}_{\psi,l}\end{bmatrix} = \overbrace{\begin{bmatrix}\mathbf{P}^T\arctan 2(^{k+1}\dot{\mathbf{y}}_l,{}^{k+1}\dot{\mathbf{x}}_l) + {}^k\boldsymbol{\lambda}_{\psi,l}\\ \mathbf{b}_{\psi,l}\end{bmatrix}}^{\mathbf{q}_{\psi,l}} \quad (19)$$

Similar to the previous step, the left hand side of (19) do not depend on the batch index l and thus we can compute the solution for the entire batch in one-shot

$$\begin{bmatrix}\mathbf{c}_{\psi,1},\boldsymbol{\mu}_{\psi,1}\\ \vdots\\ \mathbf{c}_{\psi,l},\boldsymbol{\mu}_{\psi,l}\end{bmatrix} = \left(\mathbf{Q}_\psi^{-1}\left[\mathbf{q}_{\psi,1}|\mathbf{q}_{\psi,2}|\mathbf{q}_{\psi,3}\cdots\mathbf{q}_{\psi,l}\right]\right)^T \quad (20)$$

Step (14): For a given $^{k+1}\mathbf{c}_{x,l}, {}^{k+1}\mathbf{c}_{y,l}, {}^{k+1}\mathbf{c}_{\psi,l}$ or alternately $(^{k+1}\dot{\mathbf{x}}_l, {}^{k+1}\dot{\mathbf{y}}_l, {}^{k+1}\mathbf{c}_{\psi,l})$, the velocity $v_l(t)$ at different time instants can be treated as independent of each other. In other words, each element of $\mathbf{v}_l$ is decoupled and thus the optimization (14) reduces to n parallel single-variable quadratic programming problems with a closed form solution. For $v_{min} > 0$, the solution is given by

$$^{k+1}\mathbf{v}_l = clip(\sqrt{^{k+1}\dot{\mathbf{x}}_l^2 + {}^{k+1}\dot{\mathbf{y}}_l^2}, v_{min}, v_{max}) \quad (21)$$

First part of Step (15): For a given $(^{k+1}\mathbf{x}, {}^{k+1}\mathbf{y})$, each element of $\boldsymbol{\alpha}_l$ can be considered to be decoupled from each other. Thus, optimization (15) separates into n decoupled problems with the following closed form solution

$$^{k+1}\boldsymbol{\alpha}_l = \arctan 2(a(^{k+1}\mathbf{y}_l - \boldsymbol{\xi}_y), b(^{k+1}\mathbf{x}_l - \boldsymbol{\xi}_x)) \quad (22)$$

Second part of Step (15):: Similar to previous step, each element of $\mathbf{d}_l$ can be considered independent and thus optimization over $\mathbf{d}_l$ reduces to n parallel single-variable QP with simple bound constraints. We can obtain a closed form solution by first solving the unconstrained problem and then simply clipping the value to lie between $[0\ 1]$.

Step (16): The two optimizations in this step have the same structure as (15) and thus have a closed form solution.

Remark 6. *The solution of optimization (14)-(16) involves only element-wise operations without any need of computing matrix factorization or inverse. Thus, computing a batch solution is trivial.*

Multiplier Update: The Lagrange multipliers are updated in the following manner [15] which can be trivially done over the entire batch in one-shot.

$$(^{k+1}\boldsymbol{\lambda}_{x,l}, {}^{k+1}\boldsymbol{\lambda}_{y,l}) = (^k\boldsymbol{\lambda}_{x,l}, {}^k\boldsymbol{\lambda}_{y,l}) + \rho_{xy}\mathbf{F}^T(\mathbf{F}\begin{bmatrix}^{k+1}\mathbf{c}_{x,l}\\ ^{k+1}\mathbf{c}_{y,l}\end{bmatrix} - {}^{k+1}\mathbf{g}) \quad (23)$$

$$^{k+1}\boldsymbol{\lambda}_{\psi,l} = {}^k\boldsymbol{\lambda}_{\psi,l} + \rho_{xy}\mathbf{P}^T(\arctan 2(^{k+1}\dot{\mathbf{y}}, {}^{k+1}\dot{\mathbf{x}}) - \mathbf{P}^{k+1}\mathbf{c}_{\psi,l}) \quad (24)$$

F. Goal Sampling and Meta-Cost

This section provides a goal-sampling procedure for our batch optimizer and and a meta-cost function to rank the resulting trajectories. We consider two typical scenarios encountered in highway driving.

Cruise Driving: Our first scenario considers driving forward with velocity as close as possible to a given v_{cruise}. Thus, our meta-cost is defined as simply.

$$\sum_t (v(t) - v_{cruise})^2. \tag{25}$$

The goal position are spread evenly on different lanes, each at a distance of $v_{cruise} * t_f$, where t_f is the planning horizon.

Driving with Maximum Speed close to the Right Lane: In this scenario, the ego-vehicle is required to drive as close as possible to maximum speed v_{max} while being as close to the right-lane. The meta-cost is defined as the following wherein y_{rl} is the lateral coordinate of the right-lane and w_1 and w_2 are user-defined constants.

$$\sum_t w_1 (v(t) - v_{max})^2 + w_2 (y(t) - y_{rl})^2. \tag{26}$$

The goals are sampled in the following manner. Around 60% of the goals are placed on the right-lane at different distances. The remaining goals are spread across different lanes at a distance of $v_{max} * t_f$ from the current position.

IV. Validation and Benchmarks

Implementation Details: We implemented our batch optimizer and MPC in C++ using Eigen [16]. The batch size l was taken as 11. For each driving scenario discussed in the previous section, we created two variants depending on whether the neighboring vehicles follow the synthetic Intelligent Driver Model (IDM) or the pre-recorded trajectories from NGSIM data-set [17]. In the IDM data-set, each neighboring vehicle moves parallel to center-line and just adapt their cruise forward velocity based on the distance to the vehicles in front. In the NGSIM data set, the the neighboring vehicles executes the pre-recorded trajectories. It is worth reiterating that our batch MPC and all the baselines have access to only the instantaneous position and velocity of the neighboring vehicles and not their true trajectories over the planning horizon.

Optimizer Convergence: Fig.2(a) shows the typical residual curve obtained with our batch optimizer in one of the MPC cycles. We show the trend for the best performing trajectory in the batch. We observed that on an average 100 iterations are enough to obtain residuals in the range of 10^{-3} for all the constraints.

Baselines: We benchmark against the following baselines

- **Standard MPC**: We formulate a single batch MPC wherein the meta cost function is directly embedded into the trajectory optimizer to compute a single locally optimal trajectory. We use state-of-the-art optimal control framework ACADO [6] as the solver for the standard MPC.
- **MPC with Batch ACADO:** We construct a batch version of ACADO, which solves several goal directed MPC over parallel CPU threads. This parallelization does not require any changes to be made in the matrix-algebra on the underlying SQP solver in ACADO. Thus, we use this set-up to highlight the computation gain resulting from our batch solver wherein the per-iteration computation itself vectorizes across problem instances.
- **Frenet Frame Planner:** We also compare our batch optimizer with trajectory sampling approach presented in [3] which has been extensively used in the autonomous driving community and inspired similar related approaches like [4].

A. Benchmarking

Cruise Scenario: Table II quantify the performance in the cruise-driving scenario obtained with different methods. The first two columns show the statistics of the velocity residuals $(v(t) - v_{cruise})^2$ observed over the full run of the MPC. The standard MPC performs the worst with a mean residual of 5.41 and 3.26 on the synthetic (IDM) and NGSIM data-set respectively. The worst-case performance is around 50 on both data-sets. The performance is due to the fact in dense traffic scenarios, the standard MPC trajectories are unable to find a trajectory around the neighboring slow moving vehicles (see accompanying video).

Our batch optimizer performs the best with a mean residual of 0.01 and 0.05 on the synthetic (IDM) and NGSIM data-set. In comparison, ACADO with a batch size of 11 and 6 shows comparable performance in terms of mean values. However, our batch optimizer achieves 12 times improvement over the worst-case numbers on the synthetic data set. The Frenet-frame planner with mean residuals of 0.14 and 0.28 performed worse than our's and batch ACADO. Its worst-case residual, though, is ten times ours.

High Speed Driving with Right-Lane Preference: This scenario has two competing terms in the meta cost: maximizing forward velocity and minimizing lateral distance to the right-lane. Thus, we adopt a slightly different analysis then before. We first compare the meta cost value across different methods and subsequently show how those translate to the physical metrics. Table II (last two columns) summarizes the former results. As before, the standard MPC performs worst while our batch optimizer achieves the lowest meta cost value on both synthetic (IDM) and NGSIM data-set. We map these cost values to the achieved forward velocity and lateral distance residual in Table IV. For clarity, we present the combined results over the two data sets. We observe interesting trends here as each approach attempts to minimize the meta-cost value by trading off velocities and lateral-distance residual in their own way.

Our batch optimizer achieves highest mean forward velocity of $19.26m/s$. The performance of ACADO with a batch size of 11 is comparable to ours at $18.77m/s$ while Frenet-frame planner's value stood substantially lower at $16.65m/s$.

Our batch optimizer also maintains a smaller lateral distance to the right-lane than parallelized ACADO. Our mean distance is $4.43m$ and in comparison, ACADO with batch size 11 managed a distance residual of $5.31m$ on average. The respective values achieved with Frenet-frame planner is

TABLE II

META-COST VALUES IN DIFFERENT DRIVING SCENARIOS (MEAN/MIN/MAX). LOWER IS BETTER

Method	Cruise driving (IDM)	Cruise driving (NGSIM)	High-speed driving (IDM)	High-speed driving (NGSIM)
Standard MPC	5.41 / 0.0 / 50.97	3.260 / 0.0 / 54.3	2141.4 / 1360.8 / 2668.75	997.30 / 582.59 / 1256.1
Ours	**0.01 / 0.0 / 0.05**	**0.057 / 0.0 / 0.44**	**238.0 / 135.62 / 425.14**	**236.62 / 142.14 / 574.85**
ACADO batch size 11	0.08 / 0.0 / 0.66	0.114 / 0.0 / 0.62	381.96 / 183.01 / 1304.08	376.29 / 149.93 / 880.42
ACADO batch size 6	0.12 / 0.0 / 1.06	0.103 / 0.0 / 0.85	643.97 / 129.5 / 1379.8	403.46 / 169.69 / 646.85
frenet-frame planner	0.14 / 0.0 / 1.00	0.280 / 0.02 / 0.95	563.71 / 194.64 / 1276.27	640.48 / 323.77 / 1144.0

TABLE III

ACCELERATION MAGNITUDES ACROSS DIFFERENT SCENARIOS(MEAN/MIN/MAX). LOWER IS BETTER FOR CRUISE DRIVING. FOR HIGH-SPEED DRIVING, HIGHER LINEAR ACCELERATION IS BETTER.

Method	Lin. Acc. Cruise driving	Lin. Acc. High-speed driving	Ang. Acc. Cruise driving	Ang. Acc. High-speed driving
Standard MPC	0.93 / 0.00 / 2.63	0.72 / 0.00 / 1.56	0.02 / 0.00 / 0.07	0.01 / 0.00 / 0.03
Ours	**0.11 / 0.00 / 0.28**	**0.99 / 0.00 / 1.77**	**0.02 / 0.00 / 0.07**	**0.43 / 0.00 / 0.16**
ACADO batch size 11	0.39 / 0.00 . 0.99	0.60 / 0.00 / 1.69	0.03 / 0.00 / 0.08	0.04 / 0.00 / 0.12
ACADO batch size 6	0.43 / 0.00 / 1.10	0.57 / 0.00 / 1.69	0.03 / 0.00 / 0.07	0.03 / 0.00 / 0.12
frenet-frame planner	1.21 / 0.00 / 0.45	0.19 / 0.00 / 0.48	0.15 / 0.00 / 0.44	0.13 / 0.00 / 0.40

smaller than batch ACADO and even ours since it choose a smaller velocity to quickly converge to the right-lane.

Acceleration Effort: Table III presents the acceleration statistics observed across different driving scenarios. For ease of exposition, we combined the data obtained on the synthetic (IDM) and NGSIM data-set and present the overall mean, minimum and maximum. In the cruise driving scenario, the ideal acceleration is zero as the ego-vehicle is required to maintain a constant velocity. As can be seen from Table III (first column), our batch optimizer comes very close to the ideal performance with mean and maximum linear acceleration of $0.11m/s^2$ and $0.28m/s^2$ respectively. Both the mean and maximum value of the angular acceleration values are very close to zero. All the other approaches perform worse. Intuitively, a low acceleration value suggests that our batch optimizer could continuously navigate to free space less obstructed by neighboring vehicles. We highlight this explicitly in the accompanying video. All other approaches perform substantially worse. For example, the standard MPC's mean linear acceleration is over 9 times higher than ours. ACADO with batch size of 11 and 6 uses around 4 times higher acceleration magnitudes. Unsurprisingly, Frenet-frame planner performs worst since it ignores collision-avoidance constraints during trajectory generation process. Thus, it routinely enters a situation from where it needs to apply either emergency braking or execute a sharp turn to avoid collisions.

For high-speed driving, we want the ego vehicle to drive at max speed. Thus, in contrast to cruise-driving, here, large accelerations magnitude is indeed necessary for task fulfillment. As shown in Table IV, since our batch optimizer results in trajectories that gives more preference to maximizing speed, our mean accelerations are also highest. On the other hand, Frenet-frame planner achieved the lowest velocity and thus its linear acceleration values are also the lowest.

Computation Time: We now present the most important result that is crucial in understanding the previous results in the appropriate context. Table V compares the computation-time of ours with all the baselines obtained on a $i7-8750$ processor with 16 GB RAM. Our batch optimizer with mean time of $0.07s$ is around $6\times$ faster than ACADO with a batch size of 11. In other words, ACADO needs substantially more computation budget to be even loosely competitive with our optimizer. Frenet-frame planner's timing is comparable to ours but as shown earlier, it performs the worst among all the multi-modal baselines in terms of meta-cost and acceleration effort. Our Frenet frame planner implementation used on an average 500 samples. Increasing this number, could improve the performance but at the expense of higher computation time.

Fig.2 shows how computation time scales with batch size in our optimizer and batched multi-threaded ACADO. Our optimizer shows a linear increase which can be understood in the following manner. In eqn. (18), the matrix $\mathbf{Q}_{xy}$ is independent of the batch size and only the matrix-vector on the r.h.s of (18) increases quadratically with it. The rate of increase can be made linear with simple parallelization of matrix multiplication. In contrast, batch ACADO solves full SQP in parallel CPU cores. Such parallelization efforts have thread synchronization overhead. Since the total cores in a laptop is typically 4 to 6, at batch size of 11 (or even 6), all SQP instantiations compete with each other for computing resources. Unfortunately, that is the best that we can achieve with off-the-shelf solvers. Since SQP relies on linearization, it isn't easy to parallelize its computation at each iteration (recall Section III-A).

TABLE IV

METRICS FOR HIGH-SPEED DRIVING (MEAN/MIN/MAX)

Method	Lat. dist. from right-lane	Velocity
Standard MPC	2.72 / 0.0 / 6.36	11.16 / 7.66 / 16.45
Ours	**4.43 / 0.0 / 18.0**	**19.28 / 17.14 / 20.68**
ACADO batch size 11	5.31 / 0.0 / 18.0	18.77 / 15.48 / 20.17
ACADO batch size 6	3.5 / 0.0 / 15.43	17.64 / 14.10 / 20.56
Frenet-frame planner	2.42 / 0.0 / 9.34	16.65 / 14.0 / 19.74

TABLE V

MPC COMPUTATION TIME[S] (MEAN/MIN/MAX)

Method	Cruise driving	High-speed driving
Standard MPC	0.36 / 0.05 / 0.62	0.28 / 0.05 / 0.61
Ours	**0.07 / 0.06 / 0.08**	**0.07 / 0.06 / 0.07**
ACADO batch size 11	0.36 / 0.10 / 0.61	0.44 / 0.20 / 0.67
ACADO batch size 6 (infeasible)	0.25 / 0.11 / 0.39	0.32 / 0.15 / 0.47
Frenet-frame planner	0.082 / 0.068 / 0.097	0.08 / 0.07 / 0.096

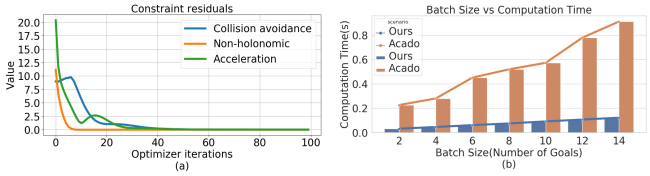

Fig. 2. (a): Residual trend observed for best performing trajectory in a batch in one of the MPC cycles. We show the different components of constraint (8c). (b): Computation time vs Batch Size comparison between our batch optimizer and batched ACADO [6] parallelized on multiple CPU threads.

V. Discussions and Future Work

This paper showed how a standard MPC based on local optimization techniques cannot generate sophisticated driving behaviors. Past works such as [3] have attempted to capture the multi-modality of autonomous driving by generating many candidate trajectories to different goals and ranking them based on a cost-function. Importantly, collision-avoidance and kinematic constraints were ignored during the trajectory generation process, leading to poor fulfillment of the given driving tasks. As a potential solution, we presented a trajectory optimizer that can generate a batch of solutions in parallel while incorporating all the necessary constraints. We showed that potential competing baselines based on state-of-the-art optimal control solver ACADO need up to 6x more computation time budget to produce comparable solution as ours.

Our meta-costs can capture some of the rule-set described in [18]. For example, the velocity residual (25) can be easily modified to induce behaviors where ego-vehicle overtakes only when it leads to some minimum gain in the forward velocity. Similarly, "keep-right" rule from [18] is already incorporated in the meta-cost (26). We present a more detailed analysis in our supplementary material [5].

Limitations and Possible Workarounds: Our batch optimization and MPC is capable of handling curved roads and residential driving scenarios (see accompanying video). However, it might struggle in highly cluttered and unstructured environments like parking lot that might require a large heading change. A possible workaround is to model the ego-vehicle geometry as a combination of circles, thus explicitly bringing the heading angle in the collision avoidance model [12]. Our preliminary results have shown that multi-circle approximation preserves the core batch structure at least for the holonomic robots [19].

Our batch optimizer structure, specifically the update rules (18),(20) is expected to be preserved for the more complex bi-cycle model of the ego-vehicle and we present a sketch of that derivation in the supplementary material [5]. However, extension to include dynamics and tire forces will require major overhaul of the matrix algebra.

Future Work: Our batch optimizer can be an attractive policy class for reinforcement learning algorithms. It can ensure safe exploration and thus would be beneficial for deep-Q learning based approaches. Our batch MPC can also be used to generate supervision data for imitation learning algorithms like [20]. Our future efforts are geared towards these directions.

References

[1] P. Bender, Ö. Ş. Taş, J. Ziegler, and C. Stiller, "The combinatorial aspect of motion planning: Maneuver variants in structured environments," in *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2015, pp. 1386–1392.

[2] X. Qian, F. Altché, P. Bender, C. Stiller, and A. de La Fortelle, "Optimal trajectory planning for autonomous driving integrating logical constraints: An miqp perspective," in *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*. IEEE, 2016, pp. 205–210.

[3] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 987–993.

[4] W. Lim, S. Lee, M. Sunwoo, and K. Jo, "Hybrid trajectory planning for autonomous driving in on-road dynamic scenarios," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[5] V. K. Adajania, A. Sharma, A. Gupta, H. Masnavi, K. M. Krishna, and A. K. Singh, "Batch-opt-highway-driving," ://github.com/dv367/Batch-Opt-Highway-Driving, 2021.

[6] B. Houska, H. J. Ferreau, and M. Diehl, "Acado toolkit—an open-source framework for automatic control and dynamic optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[7] C. Chen, Y. He, C. Bu, J. Han, and X. Zhang, "Quartic bézier curve based trajectory generation for autonomous vehicles with curvature and velocity constraints," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6108–6113.

[8] J. David, R. Valencia, R. Philippsen, P. Bosshard, and K. Iagnemma, "Gradient based path optimization method for autonomous driving," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4501–4508.

[9] K. Esterle, T. Kessler, and A. Knoll, "Optimal behavior planning for autonomous driving: A generic mixed-integer formulation," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1914–1921.

[10] F. Eiras, M. Hawasly, S. V. Albrecht, and S. Ramamoorthy, "A two-stage optimization-based motion planner for safe urban driving," *IEEE Transactions on Robotics*, 2021.

[11] V. K. Adajania, H. Masnavi, F. Rastgar, and A. K. Sing, "Embedded hardware appropriate fast 3d trajectory optimization for fixed wing aerial vehicles by leveraging hidden convex structures," in *2021 IEEE International conference on intelligent robots and systems (IROS)*. IEEE.

[12] A. K. Singh, R. R. Theerthala, M. Babu, U. K. R. Nair, and K. M. Krishna, "Bi-convex approximation of non-holonomic trajectory optimization," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 476–482.

[13] J. Nilsson, P. Falcone, M. Ali, and J. Sjöberg, "Receding horizon maneuver generation for automated highway driving," *Control Engineering Practice*, vol. 41, pp. 124–133, 2015.

[14] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.

[15] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein, "Training neural networks without gradients: A scalable admm approach," in *International conference on machine learning*. PMLR, 2016, pp. 2722–2731.

[16] G. Guennebaud, B. Jacob *et al.*, "Eigen," *URl: http://eigen. tuxfamily. org*, vol. 3, 2010.

[17] V. G. Kovvali, V. Alexiadis, and L. Zhang PE, "Video-based vehicle trajectory data collection," Tech. Rep., 2007.

[18] K. Esterle, L. Gressenbuch, and A. Knoll, "Formalizing traffic rules for machine interpretability," in *2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS)*. IEEE, 2020, pp. 1–7.

[19] F. Rastgar, H. Masnavi, K. Kruusamäe, A. Aabloo, and A. K. Singh, "Gpu accelerated batch multi-convex trajectory optimization for a rectangular holonomic mobile robot," *arXiv preprint arXiv:2109.13030*, 2021.

[20] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018.

VI. Supplementary Results: Extension to Bi-Cycle Model

We now present some preliminary derivation to show that our batch non-holonomic trajectory optimization could be extended to incorporate the bi-cycle model of the ego-vehicle while still preserving the core structure of vectorized/parallel batch solution update rule. The developments in this section can be divided into two main components

- First, we show that even with bi-cycle model, we obtain a representation for our multi-convex optimization wherein all the matrices are independent of the batch index l.
- Second, we derive the various steps of the AM procedure

A. Extension to Bi-Cycle Kinematic Model

The kinematic bi-cycle model can be represented by the following set of equations, where the symbols have the same meaning as in the main text of the paper.

$$\dot{x}(t) = v(t)\cos(\psi(t) + \beta(t)) \tag{27}$$
$$\dot{y}(t) = v(t)\sin(\psi + \beta(t) \tag{28}$$
$$\dot{\psi}(t) = \frac{v(t)}{h_r}\sin\beta(t) \tag{29}$$
$$\beta(t) = \tan^{-1}(\frac{h_r}{h_f + h_r}\tan\delta) \tag{30}$$
$$\dot{v}(t) = u_1 \tag{31}$$
$$\delta(t) = u_2 \tag{32}$$

The constants h_f, h_r respectively represent the distance from the center of mass to front and rear axle respectively. In the bi-cycle model, we incorporate an additional state variable $\beta(t)$ that represents the side-slip angle. The control inputs are the forward acceleration $\dot{v}(t)$ and the front steering input $\delta(t)$.

We note that the steering angle can be eliminated from the motion model by treating the side slip angle $\beta(t)$ as the input instead of the steering angle. The latter can always be recovered from $\beta(t)$ by using (30). Thus, we consider the following modified bi-cycle model in the following form

$$\dot{x}(t) = v(t)\cos(\psi(t) + \beta(t)) \tag{33}$$
$$\dot{y}(t) = v(t)\sin(\psi + \beta(t) \tag{34}$$
$$\dot{\psi}(t) = \frac{v(t)}{h_r}\sin\beta(t) \tag{35}$$
$$\dot{v}(t) = u_1 \tag{36}$$
$$\beta(t) = u_2 \tag{37}$$

B. Problem Formulation

Let us introduce three new variables $\widetilde{\psi}_l(t), c_{\beta,l}(t), s_{\beta,l}(t)$ and use them to formulate the following batch non-holonomic trajectory optimization with bi-cycle model.

$$\min \sum_t \ddot{x}_l(t)^2 + \ddot{y}_l(t)^2 + \ddot{\widetilde{\psi}}_l(t)^2 \tag{38a}$$
$$\dot{x}_l(t) = v_l(t)\cos\widetilde{\psi}_l(t) \tag{38b}$$
$$\dot{y}_l(t) = v_l(t)\sin\widetilde{\psi}_l(t) \tag{38c}$$
$$\dot{\psi}_l(t) = \frac{v_l(t)}{h_r}s_{\beta,l}(t) \tag{38d}$$
$$\dot{v}_l(t) = u_1 \tag{38e}$$
$$\beta_l(t) = u_2 \tag{38f}$$
$$\widetilde{\psi}_l(t) = \psi_l(t) + \beta_l(t) \tag{38g}$$
$$\sin\beta_l(t) = c_{\beta,l}(t), \cos\beta_l(t) = s_{\beta,l}(t) \tag{38h}$$
$$(x_l(t), y_l(t), \psi_l(t)) \in \mathcal{C}_{b,l} \tag{38i}$$
$$v_{min} \leq v_l(t) \leq v_{max}, \sqrt{\ddot{x}_l(t)^2 + \ddot{y}_l(t)^2} \leq a_{max} \tag{38j}$$
$$-\frac{(x_l(t) - \xi_{xj}(t))^2}{a^2} - \frac{(y_l(t) - \xi_{yj}(t))^2}{b^2} + 1 \leq 0, \tag{38k}$$

An important point to note is how we reformulate some of the motion model equations in terms of the newly introduced variables. Specifically, $\widetilde{\psi}(t)$ appears in (38b)-(38c). Similarly, the angular velocity equation is re-written in terms of $s_\beta(t)$. The intuition behind (38d) can be understood in the following manner. We treat $c_\beta(t)$ and $s_\beta(t)$ as independent variables and use it to reformulate (38d) as an affine equality constraint in terms $\dot{\psi}(t)$ and $s_\beta(t)$. At convergence, $s_\beta(t), c_\beta(t)$ would need to behave like $\sin\beta(t), \cos\beta(t)$ respectively. This consistency is enforced by introducing a new equality constraint (38h).

C. Multi-Convex Reformulation with Batch Independent Matrices

Let $\widetilde{\boldsymbol{\psi}}_l, \boldsymbol{\beta}_l$, $\mathbf{c}_{\beta,l}, \mathbf{s}_{\beta,l}$ represent the vectors formed by respectively stacking $\widetilde{\psi}_l(t), \beta(t)$, $c_{\beta,l}(t), s_{\beta,l}(t)$ at different time instants. Using these transformations and the reformulations for the collision avoidance and acceleration constraints presented in the main paper, we present the following reformulation of (38a)-(38k) in matrix form.

$$\min_{v_{min} \leq \mathbf{v}_l \leq \mathbf{v}_{max}} \frac{1}{2}\mathbf{c}_{x,l}^T\mathbf{Q}\mathbf{c}_{x,l} + \frac{1}{2}\mathbf{c}_{y,l}^T\mathbf{Q}\mathbf{c}_{y,l} + \frac{1}{2}\mathbf{c}_{\psi,l}^T\mathbf{Q}\mathbf{c}_{\psi,l} \tag{39a}$$
$$\mathbf{A}\begin{bmatrix}\mathbf{c}_{x,l}\\ \mathbf{c}_{y,l}\end{bmatrix} = \mathbf{b}_l, \mathbf{A}\mathbf{c}_{\psi,l} = \mathbf{b}_{\psi,l} \tag{39b}$$
$$\mathbf{F}\begin{bmatrix}\mathbf{c}_{x,l}\\ \mathbf{c}_{y,l}\end{bmatrix} = \mathbf{g}_l(\widetilde{\boldsymbol{\psi}}_l, \boldsymbol{\alpha}_l, \boldsymbol{\alpha}_{a,l}, \mathbf{d}_l, \mathbf{d}_{a,l}) \tag{39c}$$
$$\mathbf{F}_\psi\mathbf{c}_{\psi,l} = \mathbf{g}_\psi(\widetilde{\boldsymbol{\psi}}_l, \mathbf{v}_l, \mathbf{s}_{\beta,l}) \tag{39d}$$
$$\cos\boldsymbol{\beta}_l = \mathbf{c}_{\beta,l}, \sin\boldsymbol{\beta}_l = \mathbf{s}_{\beta,l} \tag{39e}$$

$$\mathbf{F} = \begin{bmatrix}\begin{bmatrix}\mathbf{F}_o\\ \ddot{\mathbf{P}}\\ \dot{\mathbf{P}}\end{bmatrix} & \mathbf{0}\\ \mathbf{0} & \begin{bmatrix}\mathbf{F}_o\\ \ddot{\mathbf{P}}\\ \dot{\mathbf{P}}\end{bmatrix}\end{bmatrix}, \mathbf{g}_l = \begin{bmatrix}\boldsymbol{\xi}_x + a\mathbf{d}_l\cos\boldsymbol{\alpha}_l\\ \mathbf{d}_{a,l}\cos\boldsymbol{\alpha}_{a,l}\\ \mathbf{v}_l\cos\widetilde{\boldsymbol{\psi}}_l\\ \boldsymbol{\xi}_y + b\mathbf{d}_l\sin\boldsymbol{\alpha}_l\\ \mathbf{d}_{a,l}\sin\boldsymbol{\alpha}_{a,l}\\ \mathbf{v}_l\sin\widetilde{\boldsymbol{\psi}}_l\end{bmatrix}, \tag{40}$$

$$\mathbf{F}_\psi = \begin{bmatrix}\mathbf{P}\\ \dot{\mathbf{P}}\end{bmatrix}, \mathbf{g}_{\psi,l} = \begin{bmatrix}\boldsymbol{\beta}_l - \widetilde{\boldsymbol{\psi}}_l\\ \frac{\mathbf{v}_l}{h_r}\mathbf{s}_{\beta,l}\end{bmatrix} \tag{41}$$

Remark 7. *All the matrices $(\boldsymbol{F}_0, \boldsymbol{P}, \boldsymbol{P}, \dot{\boldsymbol{F}}_\psi)$ are constant and independent of the batch index l.*

It is worth pointing out one subtle difference between (39c) and its counterpart (8c) introduced in the main paper. The argument of $\mathbf{g}_l$ in the former has $\widetilde{\psi}_l$ instead of $\mathbf{c}_{\psi,l}$ present in the latter.

Following the same process described earlier in the main paper, we formulate an augmented Lagrangian wherein we relax the non-convex constraints (39c)-(39e) as squared penalties.

$$\begin{aligned} & f_{xy}(\mathbf{c}_{x,l}, \mathbf{c}_{y,l}, \boldsymbol{\lambda}_{x,l}, \boldsymbol{\lambda}_{y,l}) + f_{\psi}(\mathbf{c}_{\psi,l}, \boldsymbol{\lambda}_{\psi,l}) \\ & + \frac{\rho_{xy}}{2} \left\| \mathbf{F} \begin{bmatrix} \mathbf{c}_{x,l} \\ \mathbf{c}_{y,l} \end{bmatrix} - \mathbf{g}_l \right\|_2^2 \\ & + \frac{\rho_{\psi}}{2} \left\| \mathbf{F}_{\psi} \mathbf{c}_{\psi,l} - \mathbf{g}_{\psi,l} \right\|_2^2 + \frac{\rho_{\beta}}{2} \left\| \begin{matrix} \cos \boldsymbol{\beta}_l - \mathbf{c}_{\beta,l} \\ \sin \boldsymbol{\beta}_l - \mathbf{s}_{\beta,l} \end{matrix} \right\|_2^2 \end{aligned} \tag{42}$$

$$f_{xy} = \frac{1}{2} \mathbf{c}_{x,l}^T \mathbf{Q} \mathbf{c}_{x,l} + \frac{1}{2} \mathbf{c}_{y,l}^T \mathbf{Q} \mathbf{c}_{y,l} - \langle \boldsymbol{\lambda}_{x,l}, \mathbf{c}_{x,l} \rangle - \langle \boldsymbol{\lambda}_{y,l}, \mathbf{c}_{y,l} \rangle \tag{43a}$$

$$f_{\psi} = \frac{1}{2} \mathbf{c}_{\psi,l}^T \mathbf{Q} \mathbf{c}_{\psi,l} - \langle \boldsymbol{\lambda}_{\psi,l}, \mathbf{c}_{\psi,l} \rangle \tag{43b}$$

D. AM Iterates

The AM approach when applied to (42) leads to very similar kind of iterates as obtained for the uni-cycle model in the main paper. We summarize this assertion in Algorithm 2. The first step (44) has the exact same structure as obtained for the uni-cycle model (see (12) in the main paper.)

Algorithm 2 Alternating Minimization based Batch Non-Holonomic Trajectory Optimization (Bi-Cycle Model)

$$ {}^{k+1}(\mathbf{c}_{x,l}, \mathbf{c}_{y,l}) = \arg\min_{\mathbf{c}_{x,l}, \mathbf{c}_{y,l}} f(\mathbf{c}_x, \mathbf{c}_y, {}^{k}\boldsymbol{\lambda}_{x,l}, {}^{k}\boldsymbol{\lambda}_{y,l}) + \frac{\rho_{xy}}{2} \left\| \mathbf{F} \begin{bmatrix} \mathbf{c}_{x,l} \\ \mathbf{c}_{y,l} \end{bmatrix} - \mathbf{g}({}^{k}\boldsymbol{\alpha}_l, {}^{k}\boldsymbol{\alpha}_{a,l}, {}^{k}\widetilde{\boldsymbol{\psi}}_l, {}^{k}\mathbf{d}_l, {}^{k}\mathbf{d}_{a,l}) \right\|_2^2, \mathbf{A} \begin{bmatrix} \mathbf{c}_{x,l} \\ \mathbf{c}_{y,l} \end{bmatrix} = \mathbf{b}_l \tag{44} $$

$$ \begin{aligned} {}^{k+1}\widetilde{\boldsymbol{\psi}}_l &= \arg\min_{\widetilde{\boldsymbol{\psi}}_l} \frac{\rho_{xy}}{2} \| \mathbf{F}^{k+1} \begin{bmatrix} {}^{k+1}\mathbf{c}_{x,l} \\ {}^{k+1}\mathbf{c}_{y,l} \end{bmatrix} - \mathbf{g}(\widetilde{\boldsymbol{\psi}}_l) \|_2^2 + \frac{\rho_{\psi}}{2} \left\| \mathbf{F}_{\psi} {}^{k}\mathbf{c}_{\psi,l} - \mathbf{g}_{\psi,l}(\widetilde{\boldsymbol{\psi}}_l) \right\|_2^2 \\ &= \arg\min_{\widetilde{\boldsymbol{\psi}}_l} \frac{\rho_{xy}}{2} \left\| \begin{matrix} {}^{k+1}\dot{\mathbf{x}}_l - \mathbf{v}\cos\widetilde{\boldsymbol{\psi}_l} \\ {}^{k+1}\dot{\mathbf{y}}_l - \mathbf{v}\sin\widetilde{\boldsymbol{\psi}_l} \end{matrix} \right\|_2^2 + \frac{\rho_{\psi}}{2} \left\| -\widetilde{\boldsymbol{\psi}_l} + \mathbf{P}^{k}\mathbf{c}_{\psi,l} + \boldsymbol{\beta}_l \right\|_2^2 \\ &= \arg\min_{\widetilde{\boldsymbol{\psi}}_l} \frac{\rho_{xy}}{2} \| \arctan 2({}^{k+1}\dot{\mathbf{y}}_l, {}^{k+1}\dot{\mathbf{x}}_l) - \widetilde{\boldsymbol{\psi}_l} \|_2^2 + \frac{\rho_{\psi}}{2} \left\| -\widetilde{\boldsymbol{\psi}_l} + \mathbf{P}^{k}\mathbf{c}_{\psi,l} + \boldsymbol{\beta}_l \right\|_2^2 \end{aligned} \tag{45} $$

$$ {}^{k+1}\mathbf{c}_{\psi} = \arg\min_{\mathbf{A}\mathbf{c}_{\psi,l} = \mathbf{b}_{\psi,l}} f(\mathbf{c}_{\psi}) + \frac{\rho_{\psi}}{2} \left\| \mathbf{F}_{\psi}\mathbf{c}_{\psi,l} - \mathbf{g}_{\psi,l}({}^{k+1}\widetilde{\boldsymbol{\psi}}_l, {}^{k}\mathbf{v}_l, {}^{k}\mathbf{s}_{\beta,l}) \right\|_2^2 \tag{46} $$

$$ \begin{aligned} {}^{k+1}\mathbf{s}_{\beta,l}, {}^{k+1}\mathbf{c}_{\beta,l} &= \arg\min_{\mathbf{s}_{\beta,l}} \frac{\rho_{\psi}}{2} \left\| \mathbf{F}_{\psi} {}^{k+1}\mathbf{c}_{\psi,l} - \mathbf{g}_{\psi,l}({}^{k+1}\widetilde{\boldsymbol{\psi}}_l, {}^{k}\mathbf{v}_l, \mathbf{s}_{\beta,l}) \right\|_2^2 \\ {}^{k+1}\mathbf{s}_{\beta,l} &= \arg\min_{\mathbf{s}_{\beta,l}} \frac{\rho_{\psi}}{2} \left\| \mathbf{P}^{k+1}\mathbf{c}_{\psi,l} - {}^{k}\mathbf{v}_l\mathbf{s}_{\beta,l} \right\|_2^2 + \frac{\rho_{\beta}}{2} \left\| \begin{matrix} \cos {}^{k}\boldsymbol{\beta}_l - \mathbf{c}_{\beta,l} \\ \sin {}^{k}\boldsymbol{\beta}_l - \mathbf{s}_{\beta,l} \end{matrix} \right\|_2^2 \end{aligned} \tag{47} $$

$$ \begin{aligned} {}^{k+1}\boldsymbol{\beta}_l &= \arg\min_{\boldsymbol{\beta}_l} \frac{\rho_{\beta}}{2} \left\| \begin{matrix} \cos\boldsymbol{\beta}_l - {}^{k+1}\mathbf{c}_{\beta,l} \\ \sin\boldsymbol{\beta}_l - {}^{k+1}\mathbf{s}_{\beta,l} \end{matrix} \right\|_2^2 \\ {}^{k+1}\boldsymbol{\beta}_l &= \arg\min_{\boldsymbol{\beta}_l} \frac{\rho_{\beta}}{2} \left\| \boldsymbol{\beta}_l - \arctan 2({}^{k+1}\mathbf{s}_{\beta,l}, {}^{k+1}\mathbf{c}_{\beta,l}) \right\|_2^2 \end{aligned} \tag{48} $$

$$ \begin{aligned} {}^{k+1}\mathbf{v}_l &= \arg\min_{v_{min} \le \mathbf{v}_l \le \mathbf{v}_{max}} \frac{\rho_{xy}}{2} \left\| \mathbf{F} \begin{bmatrix} {}^{k+1}\mathbf{c}_{x,l} \\ {}^{k+1}\mathbf{c}_{y,l} \end{bmatrix} - \mathbf{g}({}^{k+1}\mathbf{c}_{\psi,l}, \mathbf{v}_l) \right\|_2^2 + \frac{\rho_{\psi}}{2} \left\| \mathbf{F}_{\psi} {}^{k+1}\mathbf{c}_{\psi,l} - \mathbf{g}_{\psi,l}({}^{k+1}\widetilde{\boldsymbol{\psi}}_l, \mathbf{v}_l, {}^{k+1}\mathbf{s}_{\beta,l}) \right\|_2^2 \\ &= \arg\min_{v_{min} \le \mathbf{v} \le \mathbf{v}_{max}} \frac{\rho_{xy}}{2} \left\| \begin{matrix} {}^{k+1}\dot{\mathbf{x}}_l - \mathbf{v}\cos\boldsymbol{\psi}_l \\ {}^{k+1}\dot{\mathbf{y}}_l - \mathbf{v}\sin\boldsymbol{\psi}_l \end{matrix} \right\|_2^2 + \frac{\rho_{\psi}}{2} \left\| \dot{\mathbf{P}}^{k+1}\mathbf{c}_{\psi,l} - \frac{\mathbf{v}_l}{h_r} {}^{k+1}\mathbf{s}_{\beta,l} \right\|_2^2 \end{aligned} \tag{49} $$

$$ {}^{k+1}\boldsymbol{\alpha}_l = \arg\min_{\boldsymbol{\alpha}_l} \left\| \mathbf{F} \begin{bmatrix} {}^{k+1}\mathbf{c}_{x,l} \\ {}^{k+1}\mathbf{c}_{y,l} \end{bmatrix} - \mathbf{g}({}^{k}\mathbf{d}_l) \right\|_2^2 = \left\| \begin{matrix} {}^{k+1}\mathbf{x} - \boldsymbol{\xi}_x - a\ {}^{k}\mathbf{d}_l \cos\boldsymbol{\alpha}_l \\ {}^{k+1}\mathbf{y} - \boldsymbol{\xi}_y - b\ {}^{k}\mathbf{d}_l \sin\boldsymbol{\alpha}_l \end{matrix} \right\|_2^2, {}^{k+1}\mathbf{d}_l = \arg\min_{\mathbf{d}_l \ge 1} \left\| \mathbf{F}^{k+1} \begin{bmatrix} {}^{k+1}\mathbf{c}_{x,l} \\ {}^{k+1}\mathbf{c}_{y,l} \end{bmatrix} - \mathbf{g}({}^{k+1}\boldsymbol{\alpha}_l) \right\|_2^2 \tag{50} $$

$$ {}^{k+1}\boldsymbol{\alpha}_{a,l} = \arg\min_{\boldsymbol{\alpha}_{a,l}} \left\| \mathbf{F}^{k+1} \begin{bmatrix} {}^{k+1}\mathbf{c}_{x,l} \\ {}^{k+1}\mathbf{c}_{y,l} \end{bmatrix} - \mathbf{g}({}^{k}\mathbf{d}_{a,l}, \boldsymbol{\alpha}_{a,l}) \right\|_2^2, {}^{k+1}\mathbf{d}_{a,l} = \arg\min_{\mathbf{d}_{a,l} \ge a_{max}} \left\| \mathbf{F}^{k+1} \begin{bmatrix} \mathbf{c}_{x,l} \\ \mathbf{c}_{y,l} \end{bmatrix} - \mathbf{g}({}^{k+1}\boldsymbol{\alpha}_{a,l}, \mathbf{d}_{a,l}) \right\|_2^2 \tag{51} $$