Learning-based Warm-Starting for Fast Sequential Convex Programming and Trajectory Optimization

Somrita Banerjee
Stanford University
450 Serra Mall
Stanford, CA 94305
somrita@stanford.edu

Thomas Lew
Stanford University
450 Serra Mall
Stanford, CA 94305
thomas.lew@stanford.edu

Riccardo Bonalli
Stanford University
450 Serra Mall
Stanford, CA 94305
rbonalli@stanford.edu

Abdulaziz Alfaadhel
Center of Excellence for Aeronautics and Astronautics
Riyadh 13512
Saudi Arabia
aalfaadhel@kacst.edu.sa

Ibrahim Abdulaziz Alomar
Center of Excellence for Aeronautics and Astronautics
Riyadh 13512
Saudi Arabia
ialomar@kacst.edu.sa

Hesham M Shageer
Center of Excellence for Aeronautics and Astronautics
Riyadh 13512
Saudi Arabia
hshageer@kacst.edu.sa

Marco Pavone
Stanford University
450 Serra Mall
Stanford, CA 94305
pavone@stanford.edu

***Abstract*—Sequential convex programming (SCP) has recently emerged as an effective tool to quickly compute locally optimal trajectories for robotic and aerospace systems alike, even when initialized with an unfeasible trajectory. In this paper, by focusing on the Guaranteed Sequential Trajectory Optimization (GuSTO) algorithm, we propose a methodology to accelerate SCP-based algorithms through warm-starting. Specifically, leveraging a dataset of expert trajectories from GuSTO, we devise a neural-network-based approach to predict a locally optimal state and control trajectory, which is used to warm-start the SCP algorithm. This approach allows one to retain all the theoretical guarantees of GuSTO while simultaneously taking advantage of the fast execution of the neural network and reducing the time and number of iterations required for GuSTO to converge. The result is a faster and theoretically guaranteed trajectory optimization algorithm.**

TABLE OF CONTENTS

1. INTRODUCTION 1
2. PROBLEM FORMULATION 2
3. TECHNICAL APPROACH 3
4. NUMERICAL RESULTS 4
5. CONCLUSIONS 6
ACKNOWLEDGMENTS 6
REFERENCES 6
BIOGRAPHY 7

1. INTRODUCTION

In the field of robotics, an ubiquitous problem is the problem of trajectory optimization, which consists of finding a high-quality strategy described as a sequence-in-time of states and controls that is dynamically feasible, obeys all state and control constraints, and minimizes a chosen cost function. Among other applications, this is a problem relevant to the control of the new generation of autonomous space robots, such as Astrobee [1]. The goal of these assistive free-flyers on-board the International Space Station (ISS) is to assist humans in space and reduce the cost of operations, e.g. by performing tasks such as positioning of sensors, maintenance, and manipulation of payloads. However, to enable the safe operation of these robots, it is necessary to devise computationally fast algorithms to optimize their trajectories.

In the current literature, there are two broad approaches to tackle trajectory optimization. The first class of methods consists of deterministic planning algorithms, while the second one leverages methods from machine learning. A short review of those techniques is provided below.

Techniques in trajectory optimization can be divided into global search methods and local methods. Global search methods include motion planning techniques, such as sampling-based motion planning algorithms (e.g., RRT*, PRM*, and FMT*) [2], [3], [4]. Though these techniques require no initialization, they scale poorly to high-dimensional systems with kinodynamic constraints. For such reasons, they are thus instead used in practice to initialize other trajectory optimization algorithms. On the other hand, local methods are much faster and can find locally optimal solutions. However, they are often heuristic and can be very sensitive to the initial starting point [5]. Among the most promising local methods is Sequential Convex Programming (SCP) [6], [7], [8]. This technique consists of successively convexifying the underlying non-convex problem, allowing for the use of convex solvers, which usually benefit from fast convergence properties. One of the most recent examples of an SCP algorithm is Guaranteed Sequential Trajectory Optimization (GuSTO) [9], [10]. This method enjoys theoretical guarantees on convergence, does not need a feasible initial guess, and is faster than many other optimization techniques. Despite such upsides, the convergence of GuSTO and computation time strongly depend on the quality of its initialization.

A second class of methods is represented by algorithms that leverage machine learning techniques. For instance, a neural network architecture presented in [11] is capable of imitating the value iteration algorithm to solve path planning problems. Alternatively, a conditional variational autoen-

978-1-7281-2734-7/20/$31.00

coder is developed in [12] to learn a sampling distribution to inform sampling-based motion planning algorithms. By leveraging a dataset of collision-free near-optimal paths using RRT*, [13] proposed a neural network architecture to predict optimal control actions from any state, as well as a method to accelerate sampling-based planners. These methods tend to be computationally very effective; however, they usually lack guarantees of local optimality or even feasibility.

Instead of opting for an approach based on either trajectory optimization or machine learning, we propose to combine the two approaches to exploit the advantages of each. Specifically, with the objective of improving the numerical convergence of theoretically guaranteed trajectory optimization algorithms, we propose to train a neural network to predict a strategy for initializing GuSTO. More precisely, this trajectory is used as a warm-start for the GuSTO SCP algorithm. The end result is a framework that features both fast convergence and theoretical guarantees of local optimality [9]. Recently, [14] presented a similar method to predict initializing trajectories, which are then used as an initial guess for a local planner. However, instead of using a neural network as in this work, their approach uses a nearest neighbors predictor with hand-crafted descriptors of the problem. Also, their local trajectory optimization algorithms such as iLQG do not enjoy theoretical guarantees of optimality as in GuSTO. Also, in [15], an initial trajectory predicted by a neural network is refined by solving a one step quadratic program. However, their approach does not include nonconvex inequality constraints (e.g., obstacle avoidance constraints), and does not guarantee dynamical feasibility or local optimality. Our approach is different, as our goal is to predict a warm-starting trajectory for an SCP algorithm, thus retaining all the theoretical guarantees of local optimality.

In summary, this paper presents a method which uses a neural network to initialize an SCP algorithm such that it converges faster, while ensuring that the final solution retains all the theoretical guarantees provided by the original optimization algorithm. By leveraging GuSTO to provide expert trajectories and refine the predictions of the neural network, the proposed framework can handle general trajectory optimization problems with nonlinear dynamics and non-convex constraints. Specifically, we provide two main contributions:

- We combine the expressive power of machine learning tools and the robustness of trajectory optimization algorithms to design a faster algorithm whose output is a feasible solution of the original problem. Specifically, by training a neural network via supervised learning to warm-start an SCP algorithm (in particular, GuSTO), our framework not only provides a feasible strategy but also returns a locally optimal solution to the original non-convex optimal control problem.
- We demonstrate the proposed framework on a high dimensional nonlinear system with non-convex constraints. We verify the accuracy of the neural network and observe a significant speed improvement when using the neural network prediction to initialize the SCP algorithm compared to a naive initialization. Moreover, we show on specific examples that the framework performs well even on scenarios outside of the distribution of problems on which the neural network is trained, demonstrating the robustness of our hybrid learning and trajectory optimization approach.

The paper is organized as follows. In Section 2, we formulate the trajectory optimization problem. Then in Section 3, we present the framework combining a neural-network-based initialization scheme with SCP. Section 4 presents numerical results demonstrating the accuracy of the method, a reduction in computation time, and robustness properties. Finally, Section 5 provides conclusions and future research directions.

2. PROBLEM FORMULATION

We address the problem of trajectory optimization. Given an initial state $\boldsymbol{x}_{\text{init}}$, a final goal point $\boldsymbol{x}_{\text{final}}$, as well as a fixed final time t_f, the objective consists of finding a strategy, i.e. a sequence of states $\boldsymbol{x}(t)$ and controls $\boldsymbol{u}(t)$ for all times $t \in [0, t_f]$, such that a cost function (e.g. time or fuel consumption) $J(\boldsymbol{x}, \boldsymbol{u})$ is minimized, while simultaneously satisfying the dynamics of the system as well as all state and control constraints (including collision avoidance constraints). The safe subsets of the state and control space are denoted by $\mathcal{X}_{\text{safe}} \subseteq \mathbb{R}^{\text{n}}$ and $\mathcal{U} \subseteq \mathbb{R}^{\text{m}}$. This trajectory optimization problem can be expressed as the following Optimal Control Problem (**OCP**):

$$
\begin{aligned}
\min_{\boldsymbol{x},\boldsymbol{u}} \quad & J(\boldsymbol{x}, \boldsymbol{u}) \\
\text{subject to} \quad & \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)) && \forall\, t \in [0, t_f] \\
& \boldsymbol{x}(t) \in \mathcal{X}_{\text{safe}} && \forall\, t \in [0, t_f] \\
& \boldsymbol{u}(t) \in \mathcal{U} && \forall\, t \in [0, t_f] \\
& \boldsymbol{x}(0) = \boldsymbol{x}_{\text{init}} \\
& \boldsymbol{x}(t_f) = \boldsymbol{x}_{\text{final}}.
\end{aligned}
$$

The method proposed in this work applies to general instances of **OCP**. However, in this paper, for the sake of clarity, we describe the method by focusing on the dynamical model of the 13D free-flying spacecraft robot Astrobee [1]. For this system, the dynamics are as follows [16]. The state $\boldsymbol{x} = (\boldsymbol{r}, \boldsymbol{v}, \boldsymbol{q}, \boldsymbol{\omega}) \in \mathbb{R}^{13}$ consists of position $\boldsymbol{r} \in \mathbb{R}^3$, velocity $\boldsymbol{v} \in \mathbb{R}^3$, the quaternion attitude representation for rotations $\boldsymbol{q} \in \mathcal{S}^3 \subset \mathbb{R}^4$, and angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$. The control input $\boldsymbol{u} = (\boldsymbol{f}, \boldsymbol{\tau}) \in \mathbb{R}^6$ consists of the input force $\boldsymbol{f} \in \mathbb{R}^3$ and the input torque $\boldsymbol{\tau} \in \mathbb{R}^3$.

Assuming a microgravity space environment, the translational dynamics are expressed as a double integrator as

$$
\begin{bmatrix} \dot{\boldsymbol{r}}(t) \\ \ddot{\boldsymbol{r}}(t) \end{bmatrix} = \begin{bmatrix} 0_{3\times3} & I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} \end{bmatrix} \begin{bmatrix} \boldsymbol{r}(t) \\ \dot{\boldsymbol{r}}(t) \end{bmatrix} + \begin{bmatrix} 0_{3\times3} \\ \frac{I_{3\times3}}{M} \end{bmatrix} \boldsymbol{f}(t),
$$

where $M \in \mathbb{R}$ is the mass of the Astrobee robot. The attitude dynamics are given as

$$
\dot{\boldsymbol{q}} = \frac{1}{2} \boldsymbol{\Omega}(\boldsymbol{\omega}) \boldsymbol{q}, \qquad \boldsymbol{J}\dot{\boldsymbol{\omega}} = \boldsymbol{\tau} - \boldsymbol{S}(\boldsymbol{\omega}) \boldsymbol{J} \boldsymbol{\omega},
$$

where $\boldsymbol{J} \in \mathbb{R}^{3\times3}$ is the constant spacecraft inertia matrix and

$$
\boldsymbol{S}(\boldsymbol{\omega}) := \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix},
$$

$$
\boldsymbol{\Omega}(\boldsymbol{\omega}) := \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}.
$$

The state and control vectors are restricted to lie within specified bounds

$$
\begin{aligned}
\boldsymbol{x}_{\min} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{\max} \\
\boldsymbol{u}_{\min} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{\max},
\end{aligned}
$$

due to limits on the thrust of the Astrobee robot, safety limits on the speed, and position limits.

Non-convex obstacle avoidance constraints are imposed using signed distance fields. Specifically, given $n_{\mathcal{O}}$ obstacles $\mathcal{O}_i$, we define the signed distance function $d_s(\cdot)$ as

$$d_s(\boldsymbol{r}(t), \mathcal{O}_i) = \inf_{\boldsymbol{y} \in \mathcal{O}_i} \|\boldsymbol{r}(t) - \boldsymbol{y}\|_2 - \inf_{\boldsymbol{z} \notin \mathcal{O}_i} \|\boldsymbol{r}(t) - \boldsymbol{z}\|_2,$$

and impose non-convex obstacle avoidance constraints for each obstacle $\mathcal{O}_i$ as

$$d_s(\boldsymbol{r}(t), \mathcal{O}_i) \geq d_{\text{min}},$$

where $d_{\text{min}} > 0$ is a safety margin such that a ball of radius d_{min} centered at $\boldsymbol{r}(t)$ contains Astrobee. For the purpose of SCP, this constraint can be convexified using the procedure proposed in [17].

As Astrobee operates in a fixed environment, namely the ISS, we assume all obstacles to be static. If this is not the case, fast replanning with this framework is capable of handling slowly moving objects and avoid any unforeseen collision. For our numerical experiments, we consider a room with two polygonal obstacles. This scenario is similar to turning a corner on the ISS with an obstacle blocking the center of the corridor, as shown in Figure 1.

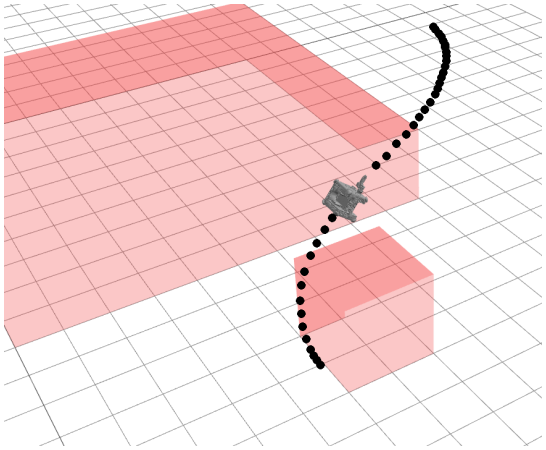

Figure 1: *3D representation of the environment used for numerical experiments, with Astrobee navigating along a collision-free trajectory.*

Finally, the cost metric is the total control effort for Astrobee:

$$J = \int_0^{t_f} \left\|\boldsymbol{u}(t)\right\|^2 \, dt = \int_0^{t_f} \left(\left\|\boldsymbol{f}(t)\right\|^2 + \left\|\boldsymbol{\tau}(t)\right\|^2 \right) \, dt$$

More details of this system can be found in [18].

An SCP method, such as Guaranteed Sequential Trajectory Optimization (GuSTO) [9], [10], solves **OCP** by successively convexifying the dynamics and state constraints, yielding a sequence of convex programs which can be efficiently solved using any generic convex program solver. In this family of convex problems, the problem at iteration $k+1$ is obtained by linearizing dynamics and state constraints around the solution found at iteration k. Initially, all non-convex terms are linearized around an initial state and control strategy, denoted as $\boldsymbol{x}_0$ and $\boldsymbol{u}_0$, respectively.

GuSTO is faster than many other SCP-based methods for complex scenarios [9], [10]. It also does not need to be initialized with a feasible state and control strategy for the method to successfully converge. Importantly, when convergence is achieved, GuSTO is proven to provide a locally optimal solution of **OCP**.

However, as mentioned in the introduction, GuSTO is not as fast as machine learning approaches, which can compute solutions rapidly, although without any guarantee of feasibility nor optimality. Additionally, GuSTO's performance, like that of any other SCP-based method, is tied to the quality of the initial state and control strategies, $\boldsymbol{x}_0$ and $\boldsymbol{u}_0$. Generally, the closer the initial guess is to the locally optimal solution, the faster and the more likely is that the SCP procedure converges.

To accelerate GuSTO while retaining all its theoretical guarantees, this paper proposes a method that leverages machine learning to compute a state trajectory $\boldsymbol{x}_0 = \{\boldsymbol{x}(t), t \in [0, t_f]\}$ and a control trajectory $\boldsymbol{u}_0 = \{\boldsymbol{u}(t), t \in [0, t_f]\}$ that efficiently initialize the algorithm.

3. TECHNICAL APPROACH

The proposed approach consists of training a neural network which, for an environment with a predefined set of obstacles, and given an initial state $\boldsymbol{x}_{\text{init}}$ and a final state $\boldsymbol{x}_{\text{final}}$, provides a state and control trajectory $(\boldsymbol{x}_0, \boldsymbol{u}_0)$ which is a good initial guess for GuSTO. To do so, a dataset of expert trajectories from the SCP algorithm is generated and used for supervised learning. As mentioned previously, only static obstacles are accounted for during the training of the neural network (e.g., walls and corridors of the ISS). Additional obstacles that are not in the training dataset (e.g., smaller floating objects entering the field of operation of Astrobee) can be accounted for via high frequency recomputations, critically enabled by the speed-ups achieved via warm-starting.

In our approach, the neural network predicts a continuous time trajectory parameterized as a p^{th} order polynomial, which implies that the neural network only needs to predict $p+1$ polynomial coefficients for each dimension. For the dynamics of Astrobee and the obstacle field considered in this work, a polynomial of degree $p = 4$ provides accurate trajectories. Other parameterizations such as Bezier curves [19], B-splines [20], [21] or higher degree polynomials could also be used for different scenarios and the approach proposed in this work can be easily adapted to such parameterizations. As shown in Table 1, the inputs to the neural network are the desired initial and final states of the problem, that is $\boldsymbol{x}_{\text{init}}$ and $\boldsymbol{x}_{\text{final}}$. The output is a prediction of both the locally optimal state and control trajectories, with each dimension represented as a p^{th} degree polynomial in time. Denoting the i^{th} components of the state $\boldsymbol{x}$ as x_i, the state trajectory at time t is expressed as

$$x_i(t) = \sum_{k=0}^{p} \alpha_{i,k} t^k;$$

the control trajectory $\boldsymbol{u}(t)$ has an analogous parameterization.

As shown in Figure 2, the neural network has a feedforward architecture with 3 fully connected hidden layers with 256, 512 and 256 units, respectively. All weights are initialized uniformly and ReLU activation functions are chosen for

Table 1: *Neural network inputs and outputs. The inputs are the desired initial and final states of the problem. The output is a prediction of the locally optimal state and control trajectories, represented as fourth order polynomials in time.*

Input	Description	Size
$\boldsymbol{x}_{\text{init}}$	Initial state $(\boldsymbol{r}, \boldsymbol{v}, \boldsymbol{q}, \boldsymbol{\omega})$	13
$\boldsymbol{x}_{\text{final}}$	Final state $(\boldsymbol{r}, \boldsymbol{v}, \boldsymbol{q}, \boldsymbol{\omega})$	13
	Total	**26**

Output	Description	Size
$\alpha_{i,k}$	Polynomial coefficients k = 0:4 for the i^{th} dimension of the state or control, i = 1:19	5x19
	Total	**95**

each layer. The architecture was chosen by comparing the test error across different models, and choosing the smallest architecture yielding good generalization over the different scenarios in the test set.

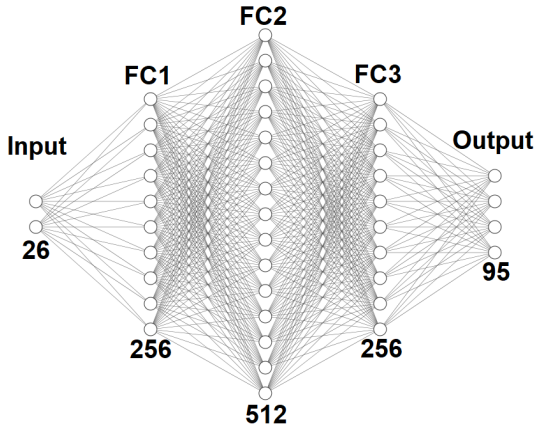

Figure 2: *Architecture of the neural network predicting an initializing strategy for GuSTO. ReLU activation functions are used for each neuron.*

The training dataset consists of the solutions to 11,297 instances of **OCP** computed by GuSTO[2] in an environment with two polygonal obstacles represented in Figure 1. To generate different scenarios, the initial and final states are uniformly sampled throughout the 13-dimensional space, while respecting the state constraints of the problem. IPOPT [22] is used to solve the convex subproblems defined by GuSTO and generate the optimal trajectories for **OCP**. Then, the resulting sequence of locally optimal states and control inputs are fitted via polynomial regression to obtain the polynomial coefficients $\alpha_{i,k}$ describing the continuous time trajectories $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ to be predicted by the neural network. As discussed previously, a polynomial representation of degree 4 is accurate enough for the class of trajectories considered in this work.

Using the Keras API for Tensorflow [23], the neural network is trained for 30 epochs using an ℓ_2 loss on the predicted and fitted (true) polynomial coefficients $\alpha_{i,k}$. Of the 11,297 problems in the dataset, 10% (1130 problems) are set aside and reserved for validation, to never be used in training. The remaining 90% are further split into 75% training (to be used to fit the neural network model) and 25% test (to tune hyperparameters during training).

[2]The code is available at `https://github.com/StanfordASL/nnGuSTO`.

4. Numerical Results

In this section, the accuracy of the predictions of the neural network are evaluated on the validation set, i.e. on the 1130 problems which were never used for training. Then, the predicted trajectories are used to warm-start GuSTO. We compare the results generated with: 1) GuSTO with a naive straight-line initialization, 2) the neural network prediction, and 3) GuSTO with a warm-start from the neural network.

Prediction Accuracy

To assess the accuracy of the predictions of the neural network, the relative trajectory prediction error is computed for each of the 1130 problems in the validation dataset for each of the dimensions of the state and control trajectories as

$$\text{Traj. pred. error for } i^{\text{th}} \text{ dim} := \frac{\int_{t=0}^{t_f} \left\| x_i^{\text{pred}}(t) - x_i^{\text{true}}(t) \right\| \mathrm{d}t}{\int_{t=0}^{t_f} \left\| x_i^{\text{true}}(t) \right\| \mathrm{d}t},$$

where $\boldsymbol{x}^{\text{pred}}$ is the trajectory predicted by the neural network, and $\boldsymbol{x}^{\text{true}}$ is the locally optimal trajectory obtained from GuSTO and fitted to a polynomial of degree 4, as described in Section 3. The same error metric is used to evaluate the prediction accuracy of the control input trajectory. All results are reported in Table 2, showing an average error under 15%. Although the accuracy of the neural network is not perfect, these results are sufficient to provide a speed improvement for GuSTO, as shown in the following section.

Table 2: *Trajectory prediction errors evaluated on the 1130 scenarios from the validation dataset.*

	Trajectory prediction error	
	Mean	Std Dev
Position ($\boldsymbol{r}$)	3.66%	4.55%
Velocity ($\boldsymbol{v}$)	13.13%	16.96%
Quaternion attitude ($\boldsymbol{q}$)	1.19%	6.63%
Angular velocity ($\boldsymbol{\omega}$)	2.34%	9.21%
Input force ($\boldsymbol{f}$)	13.96%	17.62%
Input torque ($\boldsymbol{\tau}$)	3.24%	9.40%

Further, we evaluate the prediction accuracy of the neural network on a challenging problem shown in Figure 3. For this specific scenario, the trajectory predicted by the neural network is close to the locally optimal trajectory computed by GuSTO. Moreover, as expected, this result also shows that the solution of GuSTO warm-started using the prediction from the neural network is identical to the solution of GuSTO initialized with a straight-line trajectory.

Acceleration through Warm-Starting

In this section, we show that using the predictions of the neural network as a warm-start accelerates the convergence

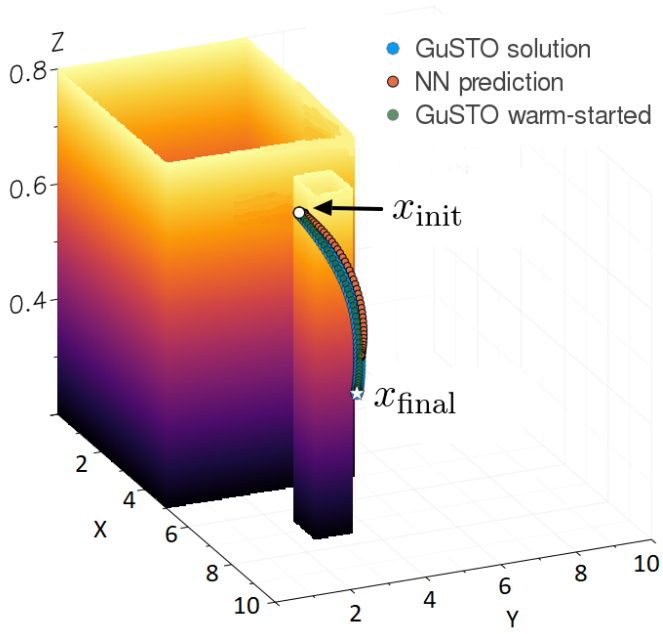

Figure 3: *Locally optimal trajectory computed by (1) GuSTO initialized with a straight line, (2) direct inference from the neural network, and (3) GuSTO warm-started with the neural network prediction. The two solutions obtained after convergence of GuSTO are nearly identical, and the neural network is able to predict a trajectory very close to the locally optimal trajectory.*

of GuSTO, sometimes significantly. In Table 3, the number of SCP iterations until convergence are reported for all 1130 scenarios of the validation dataset. On average, we observe a reduction of 17% in the number of iterations. Also, the cost of the final trajectory computed using warm-starting remains similar to that of a traditional straight-line initialization, as expected. Moreover, all constraints (including non-convex dynamics and obstacle avoidance constraints) are satisfied in all problems, demonstrating the robustness of the proposed framework.

Table 3: *Number of SCP iterations and optimal cost of solution for all 1130 problems in the validation set, comparing the results using a straight-line initialization and the prediction of the neural network as a warm-start.*

		Cold-start	Warm-start
Number of iterations	Average	4.03	3.33
	Std dev	4.69	3.25
	Worst case	52	34
Total cost of solution	Average	0.010	0.008
	Std dev	0.016	0.014
	Worst case	0.050	0.050

Next, we focus our study on challenging scenarios where GuSTO requires at least 10 iterations to converge using a naive straight-line initialization. In such cases, we observe an improvement of approximately 57% fewer iterations when leveraging the predictions of the neural network as a warm-start, as reported in Table 4. This demonstrates that using a neural network to warm-start GuSTO can greatly reduce the number of SCP iterations required to converge to a locally optimal solution.

Table 4: *Number of SCP iterations and optimal cost of solution for complex problems from the validation set, where GuSTO requires at least 10 iterations to converge using a straight-line initialization.*

		Cold-start	Warm-start
Number of iterations	Average	16.59	7.20
	Std dev	8.36	5.71
	Worst case	52	34
Total cost of solution	Average	0.020	0.019
	Std dev	0.017	0.016
	Worst case	0.050	0.050

For the purpose of visualization, four of the problems from the validation dataset are presented in Figure 4. For each scenario, we compare the computed trajectories obtained from the neural network, from GuSTO using a straight-line initialization, and using the warm-start provided by the neural network. First, we observe that the trajectories after convergence of GuSTO are nearly identical, as reflected in Table 4. Moreover, we observe that although the prediction of the neural network is not a feasible solution to **OCP**, it still leads to a reduction in the number of required SCP iterations for GuSTO, as reported in Table 4.

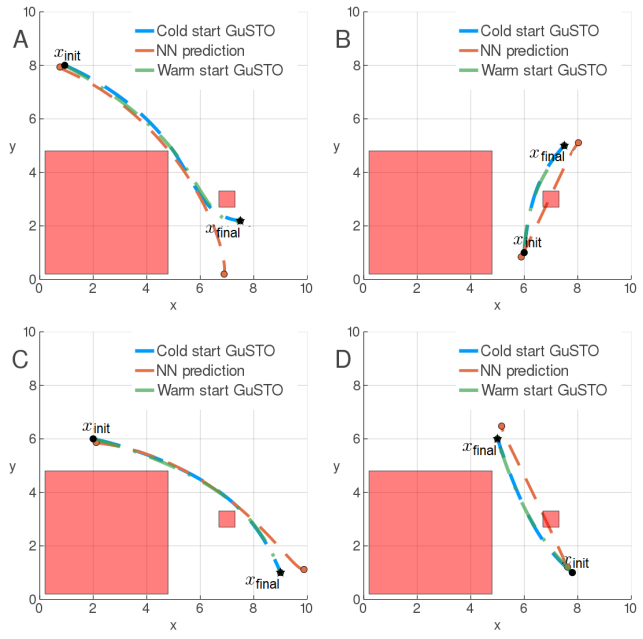

Figure 4: *Four scenarios comparing the trajectories obtained from (1) GuSTO initialized with a straight line, (2) direct inference from the neural network, and (3) GuSTO warm-started with the neural network prediction.*

Robustness in the Presence of Unmodeled Obstacles

Finally, we demonstrate that the prediction of the neural network can also be used to warm-start GuSTO in the presence of obstacles which are not present in the training set, e.g. a

moving obstacle entering the workspace. Four scenarios are extracted from the validation dataset and obstacles of arbitrary size are inserted into the environment. Then, GuSTO is used with and without warm-starting to compute a solution to **OCP**, and the resulting trajectories are shown in Figure 5. Note that in all four of these cases, a speed improvement due to warm-starting is observed.

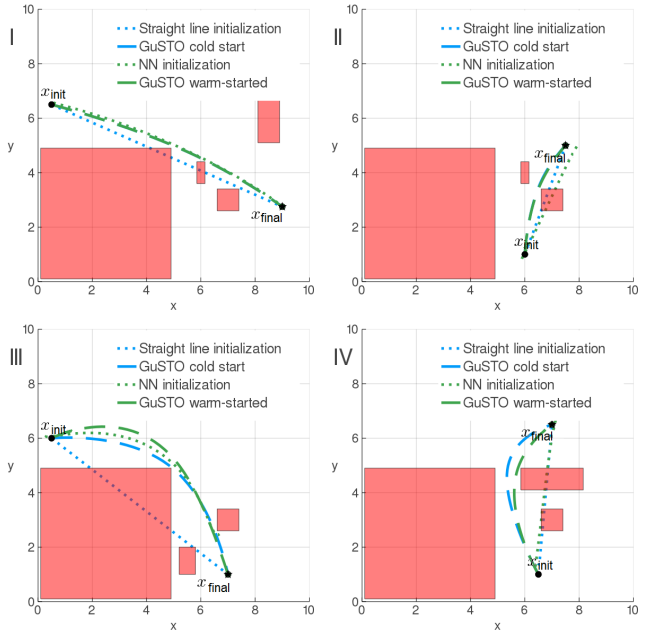

Figure 5*: Trajectory optimization results in presence of obstacles which are not in the training set of the neural network. Although the prediction is often infeasible, the neural network can still be used to warm-start GuSTO, reducing computation time and leading to a feasible solution for* ***OCP****.*

These preliminary results show promise that using a neural network with partial knowledge of the environment still enables robust warm-starting of trajectory optimization algorithms. The infeasibility of the solution predicted by the neural network does not appear to be an issue, as GuSTO is capable of refining this initial guess to compute a feasible and locally optimal trajectory.

5. Conclusions

This work presented a framework which combines GuSTO, an efficient sequential convex programming technique for trajectory optimization, with a neural network, enabling fast and robust trajectory optimization with theoretical guarantees. Using the output of the trained model to warm-start the nonlinear solver, the proposed approach reduces computation time, while obtaining feasible and locally optimal solutions to the trajectory optimization problem. Furthermore, we showed that even in the presence of obstacles which are not accounted for in the training of the neural network, its output can still be used as a warm-start to improve the speed of GuSTO, while retaining local optimality and feasibility.

An interesting future research direction consists of training a neural network to predict the Lagrange multipliers corresponding to the initial state constraint. Indeed, these multipliers were shown to converge to the Pontryagin extremal associated with the optimal control problem evaluated at time zero [9], which can be used to initialize a shooting method and efficiently compute an optimal solution to the trajectory optimization problem. Second, different initialization strategies could be used to generate the training dataset of the neural network. For instance, smoothing solutions from sampling-based planners by using SCP would ensure that the training dataset consists of globally, near-optimal solutions, similarly as in [24]. Finally, providing the network with a representation of the environment, as in [12], [14], [25], would help generalize to different obstacle fields. This could improve the robustness of the solution in changing environments.

Acknowledgments

This work is supported by the King Abdulaziz City for Science and Technology (KACST), and by the Stanford Graduate Fellowship (SGF). The authors would like to thank Andrew Bylard and Boris Ivanovic for useful discussion during the development of this work.

References

[1] T. Smith, J. Barlow, M. Bualat, T. Fong, C. Provencher, H. Sanchez, and E. Smith, "Astrobee: A new platform for free-flying robotics on the International Space Station," in *Int. Symp. on Artificial Intelligence, Robotics and Automation in Space*, 2016.

[2] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Workshop on Algorithmic Foundations of Robotics*, 2000.

[3] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[4] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast Marching Tree: a fast marching sampling-based method for optimal motion planning in many dimensions," *Int. Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, 2015.

[5] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge Univ. Press, 2004.

[6] X. Liu and P. Lu, "Solving nonconvex optimal control problems by convex optimization," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 37, no. 3, pp. 750 – 765, 2014.

[7] Y. Mao, M. Szmuk, and B. Açikmeşe, "Successive convexification of non-convex optimal control problems and its convergence properties," in *Proc. IEEE Conf. on Decision and Control*, 2016.

[8] Y. Mao, D. Dueri, M. Szmuk, and B. Açikmeşe, "Successive Convexification of Non-Convex Optimal Control Problems with State Constraints," *IFAC-Papers Online*, vol. 50, no. 1, pp. 4063–4069, 2017.

[9] R. Bonalli, A. Cauligi, A. Bylard, and M. Pavone, "GuSTO: guaranteed sequential trajectory optimization via sequential convex programming," in *Proc. IEEE Conf. on Robotics and Automation*, 2019.

[10] R. Bonalli, A. Bylard, A. Cauligi, T. Lew, and M. Pavone, "Trajectory optimization on manifolds: A theoretically-guaranteed embedded sequential convex

programming approach," in *Robotics: Science and Systems*, 2019.

[11] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," in *Conf. on Neural Information Processing Systems*, 2016.

[12] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *Proc. IEEE Conf. on Robotics and Automation*, 2018.

[13] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *Proc. IEEE Conf. on Robotics and Automation*, 2019.

[14] N. Jetchev and M. Toussaint, "Fast motion planning from experience: trajectory prediction for speeding up movement generation," *Autonomous Robots*, vol. 34, no. 1-2, pp. 111–127, 2013.

[15] G. Tang, W. Sun, and K. Hauser, "Learning trajectories for real-time optimal control of quadrotors," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2018.

[16] G. S. Aoudé, "Two-stage path planning approach for designing multiple spacecraft reconfiguration maneuvers and applications to SPHERES onboard ISS," Master's thesis, Massachusetts Inst. of Technology, 2007.

[17] J. Virgili-llop, C. Zagaris, R. Zappulla II, A. Bradstreet, and M. Romano, "Convex optimization for proximity maneuvering of a spacecraft with a robotic manipulator," in *AIAA/AAS Space Flight Mechanics Meeting*, 2017.

[18] Astrobee robot software. NASA. Available at https://github.com/nasa/astrobee.

[19] K. R. Simba, N. Uchiyama, and S. Sano, "Real-time smooth trajectory generation for nonholonomic mobile robots using bézier curves," *Robotics and Computer-Integrated Manufacturing*, vol. 41, no. 1, pp. 31–42, 2016.

[20] K. Wang, "B-splines joint trajectory planning," *Computers in Industry*, vol. 10, no. 2, pp. 113 – 122, 1988.

[21] Y.-C. Chen, "Solving robot trajectory planning problems with uniform cubic B-splines," *Optimal Control Applications and Methods*, vol. 12, no. 4, pp. 247–262, 1991.

[22] A. Wachter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[23] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[24] F. Baldini, S. Bandyopadhyay, R. Foust, S.-J. Chung, A. Rahmani, J.-P. de la Croix, A. Bacula, C. M. Chilan, and F. Hadaegh, "Fast motion planning for agile space systems with multiple obstacles," in *AIAA/AAS Astrodynamics Specialist Conference*, 2016.

[25] T. Salzmann, B. Ivanovic, and M. Pavone, "Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control," 2020, available at https://stanfordasl.github.io/wp-content/papercite-data/pdf/Salzmann.Ivanovic.ea.2020.pdf.

BIOGRAPHY

Somrita Banerjee *received her B.S. degree in Mechanical and Aerospace Engineering from Cornell University in 2017. She is currently a graduate student in the Aeronautics and Astronautics department at Stanford University. Her research focuses on problems involving robotics in space applications.*

Thomas Lew *received his BSc. degree in Microengineering from Ecole Polytechnique Federale de Lausanne in 2017, his MSc. degree in Robotics from ETH Zurich in 2019, and is currently pursuing his Ph.D. degree in Aeronautics and Astronautics at Stanford University. His research focuses on the intersection between optimization, control theory and machine learning techniques for aerospace applications.*

Dr. Riccardo Bonalli *obtained his MSc in Mathematical Engineering from Politecnico di Milano, Italy, in 2014, and his Ph.D. in applied mathematics from Sorbonne Universite, France, in 2018, in collaboration with ONERA - The French Aerospace Lab, France. He is now postdoctoral researcher at the Department of Aeronautics and Astronautics, at Stanford University. His main research interests concern the theoretical and numerical optimal control with applications in aerospace engineering and robotics.*

Abdulaziz Alfaadhel *is a research specialist with the Center of Excellence for Aeronautics and Astronautics (CEAA) at KACST and Stanford University. He received his Bachelor's Degree in Aerospace Engineering from the King Fahd University of Petroleum and Minerals (KFUPM), Dhahran in May 2018. His research interests include optimal control, trajectory optimization and orbital mechanics.*

Ibrahim Abdulaziz Alomar *has a bachelor of science in Mechanical engineering from King Saud University and he currently works with the Center of Excellence for Aeronautics and Astronautics (CEAA) at KACST. His main research interest are modeling and control of mechanical and aerospace systems.*

Dr. Hesham M Shageer *is an Assistant Research Professor with the King Abdulaziz City for Science and Technology (KACST). He is one of the Principal Investigators with the collaborative research Center of Excellence for Aeronautics and Astronautics (CEAA) a joint effort between KACST and Stanford University. His research experience includes Adaptive Control Theory and Design Methodologies, System Modeling and Numerical Simulation, as well as Optimal Experimental Design. Hesham received his B.S. (2004), M.S. (2006) and Ph.D. (2013) degrees from the University of Virginia (UVA) with a major in Electrical Engineering and a specialization in Control Systems. While at UVA, he participated on a novel aircraft control design project funded by NASA, in addition to the Solar Decathlon Design Competition. His research interests span multiple domains including Machine Learning Based Adaptive Control Synthesis, Autonomous Aircraft System Design, Green Engineering and Bio-mimicking Design Concepts.*

Dr. Marco Pavone *is an Associate Professor of Aeronautics and Astronautics at Stanford University, where he is the Director of the Autonomous Systems Laboratory. Before joining Stanford, he was a Research Technologist within the Robotics Section at the NASA Jet Propulsion Laboratory. He received a Ph.D. degree in Aeronautics and Astronautics from the Massachusetts Instituteof Technology in 2010. His main research interests are in the development of methodologies for the analysis, design, and control of autonomous systems, with an emphasis on self-driving cars, autonomous aerospace vehicles, and future mobility systems. He is a recipient of a number of awards, including a Presidential Early Career Award for Scientists and Engineers, an ONR YIP Award, an NSF CAREER Award, and a NASA Early Career Faculty Award. He was identified by the American Society for Engineering Education (ASEE) as one of America's 20 most highly promising investigators under the age of 40. He is currently serving as an Associate Editor for the IEEE Control Systems Magazine.*