sensors

MDPI

Article

Fast Adaptation of Manipulator Trajectories to Task Perturbation by Differentiating through the Optimal Solution

Shashank Srikanth [1], Mithun Babu [1], Houman Masnavi [2], Arun Kumar Singh [2,*], Karl Kruusamäe [2] and Krishnan Madhava Krishna [1]

[1] Robotics Research Center, KCIS, IIIT Hyderabad, Hyderabad 500032, India; s.shashank2401@gmail.com (S.S.); mithunbabu1141995@gmail.com (M.B.); mkrishna@iiit.ac.in (K.M.K.)
[2] Institute of Technology, University of Tartu, 50090 Tartu, Estonia; houman.masnavi@ut.ee (H.M.); karl.kruusamae@ut.ee (K.K.)
* Correspondence: arun.singh@ut.ee

Abstract: Joint space trajectory optimization under end-effector task constraints leads to a challenging non-convex problem. Thus, a real-time adaptation of prior computed trajectories to perturbation in task constraints often becomes intractable. Existing works use the so-called warm-starting of trajectory optimization to improve computational performance. We present a fundamentally different approach that relies on deriving analytical gradients of the optimal solution with respect to the task constraint parameters. This gradient map characterizes the direction in which the prior computed joint trajectories need to be deformed to comply with the new task constraints. Subsequently, we develop an iterative line-search algorithm for computing the scale of deformation. Our algorithm provides near real-time adaptation of joint trajectories for a diverse class of task perturbations, such as (i) changes in initial and final joint configurations of end-effector orientation-constrained trajectories and (ii) changes in end-effector goal or way-points under end-effector orientation constraints. We relate each of these examples to real-world applications ranging from learning from demonstration to obstacle avoidance. We also show that our algorithm produces trajectories with quality similar to what one would obtain by solving the trajectory optimization from scratch with warm-start initialization. Most importantly, however, our algorithm achieves a worst-case speed-up of 160x over the latter approach.

Keywords: manipulation; task perturbation; optimization; control

Citation: Srikanth, S.; Babu, M.; Masnavi, H.; Kumar Singh, A.; Kruusamäe, K.; Krishna, K.M. Fast Adaptation of Manipulator Trajectories to Task Perturbation by Differentiating through the Optimal Solution. *Sensors* **2022**, *22*, 2995. https://doi.org/10.3390/s22082995

Academic Editor: Gregor Klancar

Received: 1 March 2022
Accepted: 6 April 2022
Published: 13 April 2022

1. Introduction

A change in task-specification is often unavoidable in real-world manipulation problems. For example, consider a scenario where a manipulator is handing over an object to a human. The robot's estimate of the goal position can change as it executes its prior computed trajectories. Consequently, it needs to quickly adapt its joint motions to reach the new goal position. In this paper, we model motion planning as a parametric optimization problem wherein the task specifications are encoded in the parameters. In this context, adaptation to a new task requires re-computing the optimal joint trajectories for the new set of parameters. This is a computationally challenging process as the underlying cost functions in typical manipulation tasks are highly non-linear and non-convex [1]. Existing works leverage the so-called warm-starting technique where prior computed trajectories are used as initialization for the optimization solvers [2]. However, our extensive experimentation with off-the-shelf optimization solvers such as Scipy-SLSQP [3] show it is not sufficient for real-time adaptation of joint trajectories to task perturbations.

1.1. Main Idea

The proposed work explores an alternate approach based on differentiating the optimal solution with respect to the problem parameters, hereafter referred to as the Argmin differenti-

ation [4]. To understand this further, consider the following constrained optimization problem over variable $\boldsymbol{\xi}$ (e.g., joint angles) and parameter vector $\mathbf{p}$ (e.g., end-effector position).

$$\boldsymbol{\xi}^*(\mathbf{p}) = \arg\min f(\boldsymbol{\xi}, \mathbf{p}) \tag{1}$$
$$g_i(\boldsymbol{\xi}, \mathbf{p}) \leq 0, \forall i = 1, 2, \ldots n \tag{2}$$
$$h_j(\boldsymbol{\xi}, \mathbf{p}) = 0, \forall j = 1, 2, \ldots m \tag{3}$$

The optimal solution $\boldsymbol{\xi}^*$ satisfies the following Karush–Kuhn Tucker (KKT) conditions.

$$\nabla f(\boldsymbol{\xi}^*, \mathbf{p}) + \sum_i \lambda_i \nabla g_i(\boldsymbol{\xi}^*, \mathbf{p}) + \sum_j \mu_j \nabla h_j(\boldsymbol{\xi}^*, \mathbf{p}) = 0 \tag{4a}$$
$$g_i(\boldsymbol{\xi}^*, \mathbf{p}) \leq 0, \forall i \tag{4b}$$
$$h_j(\boldsymbol{\xi}^*, \mathbf{p}) = 0 \tag{4c}$$
$$\lambda_i \geq 0, \lambda_i g_i(\boldsymbol{\xi}^*, \mathbf{p}) = 0, \forall i. \tag{4d}$$

The gradients in (4a) are taken with respect to $\boldsymbol{\xi}$. The variables λ_i, μ_j are called the Lagrange multipliers. Now, consider a scenario where the optimal solution $\boldsymbol{\xi}^*$ for the parameter $\mathbf{p}$ needs to be adapted for the perturbed set $\overline{\mathbf{p}} = \mathbf{p} + \Delta\mathbf{p}$. As mentioned earlier, one possible approach is to resolve the optimization with $\boldsymbol{\xi}^*$ as the warm-start initialization. Alternately, for $\Delta\mathbf{p}$ with a small magnitude, an analytical perturbation model can be constructed. To be precise, we can compute the first-order differential of the r.h.s. of (4a)–(4d) to obtain analytical gradients in the following form [5–7].

$$(\nabla_{\mathbf{p}}\boldsymbol{\xi}^*, \nabla_{\mathbf{p}}\lambda_i, \nabla_{\mathbf{p}}\mu_i^*) = \mathbf{F}(\boldsymbol{\xi}^*, \mathbf{p}, \lambda_i, \mu_j) \tag{5}$$

Multiplying the gradients with $\Delta\mathbf{p}$ gives us an analytical expression for the new solution and Lagrange multipliers corresponding to the perturbed parameter set [7].

1.2. Contribution

Algorithmic Contribution: A critical bottleneck in using the gradient map of the form (5) to compute perturbed solutions is that the mapping between $\Delta\mathbf{p}$ and λ_i is highly discontinuous. In other words, even a small $\Delta\mathbf{p}$ can lead to large changes in the so-called active-set of the inequality constraints. Thus it becomes necessary to develop additional active-set prediction mechanisms [7]. In this paper, we bypass this complication by instead focusing on the parametric optimization with only bound constraints on the variable set. Argmin differentiation of such problems has a simpler structure, which we leverage to develop a line-search based algorithm to incrementally adopt joint trajectories to larger changes in the parameter/tasks. To give some example of "large perturbation", our algorithm can adapt the joint trajectories of Franka–Panda arm to a perturbation of up to 30 cm in the goal position. This is almost 30% of the workspace of the Franka arm.

Application Contribution: For the first time, we apply the Argmin differentiation concept to the problem of joint trajectory optimization for the manipulators under end-effector task constraints. We consider a diverse class of cost functions to handle (i) perturbations in joint configurations or (ii) end-effector way-points in orientation-constrained end-effector trajectories. We present an extensive benchmarking of our algorithm's performance as a function of the perturbation magnitude. We also show that our algorithm outperforms the warm-start trajectory optimization approach in computation time by several orders of magnitude while achieving similar quality as that measured by task residuals and smoothness of the resulting trajectory.

1.3. Related Works

The concept of Argmin differentiation has been around for a few decades, although often under the name of sensitivity analysis [8,9]. However, of late it has seen a resurgence, especially in the context of end-to-end learning of control policies [10,11]. Our proposed work is more closely related to those that use Argmin differentiation for motion planning or feedback control. In this context, a natural application of Argmin differentiation is

in bi-level trajectory optimization where the gradients of the optimal solution from the lower level are propagated to optimize the cost function at the higher level. This technique has been applied to both manipulation and navigation problems in existing works [6,12]. Alternately, Argmin differentiation can also be used for the correction of prior-computed trajectories [7,13].

To the best of our knowledge, we are not aware of any work that uses Argmin differentiation for the adaptation of task-constrained manipulator joint trajectories. The closest to our approach is [5] that uses it to accelerate the inverse kinematics problem. Along similar lines, [7] considers a very specific example of perturbation in the end-effector goal position. In contrast to these two cited works, we consider a much more diverse class of task constraints. Furthermore, our formulation also has important distinctions with [7] at the algorithmic level. Authors in [7] use the log-barrier function for including inequality constraints as penalties in the cost function. In contrast, we note that in the context of the task-constrained trajectory optimization considered in this paper, the joint angle limits are the most critical. The velocity and acceleration constraints can always be satisfied through time-scaling based pre-processing [14]. Thus, by choosing a way-point parametrization for the joint trajectories, we formulate the underlying optimization with just box constraints on the joint angles. This, in turn, allows us to treat this constraint through simple projection (Line 4 in Algorithm 1) without disturbing the structure of the cost function and the resulting Jacobian and Hessian matrices obtained by Argmin differentiation.

Algorithm 1 Line-Search Based Joint Trajectory Adaptation to Task Perturbation

1: Initialize ${}^{k}\boldsymbol{\xi}^{*}$ as the solution for the prior parameter ${}^{k}\mathbf{p}$, the Hessian ${}^{k}\nabla^{2}_{\xi} f({}^{k}\boldsymbol{\xi}, \mathbf{p})$, the gradient $\nabla_{\xi,p_i} f({}^{k}\boldsymbol{\xi}, \mathbf{p})$, and ${}^{k}\Delta\mathbf{p} = \overline{\mathbf{p}} - {}^{k}\mathbf{p}$

2: **while** $\eta > 0$ **do**

$$\max \eta \tag{6a}$$

$$f({}^{k}\boldsymbol{\xi}^{*}(\mathbf{p} + \eta\Delta\mathbf{p}), \mathbf{p} + \Delta\mathbf{p}) \leq f({}^{k}\boldsymbol{\xi}^{*}, \mathbf{p} + \Delta\mathbf{p}) \tag{6b}$$

3:

$${}^{k+1}\boldsymbol{\xi}^{*} = {}^{k}\boldsymbol{\xi}^{*} + \eta\nabla_{\mathbf{p}}\boldsymbol{\xi}^{*}\Delta^{k}\mathbf{p} \tag{7}$$

4:

$${}^{k+1}\boldsymbol{\xi}^{*} = Project(\boldsymbol{\xi}_{lb}, \boldsymbol{\xi}_{ub}) \tag{8}$$

5: Update ${}^{k+1}\overline{\mathbf{p}} = ForwardRoll({}^{k+1}\boldsymbol{\xi}^{*})$

6: Update ${}^{k+1}\Delta\mathbf{p} = \overline{\mathbf{p}} - {}^{k+1}\mathbf{p}$.

7: Update Hessian $\nabla^{2}_{\xi} f({}^{k+1}\boldsymbol{\xi}, {}^{k+1}\mathbf{p})$.

8: Update Jacobian $\nabla_{\xi,p} f({}^{k+1}\boldsymbol{\xi}, {}^{k+1}\mathbf{p})$

9: **end while**

2. Proposed Approach

2.1. Symbols and Notations

We will use lower case normal font letters to represent scalars, while bold font variants will represent vectors. Matrices are represented by upper case bold fonts. The subscript t will be used to denote the time stamp of variables and vectors. The superscript T will represent the transposing of a matrix.

2.2. Argmin Differentiation for Unconstrained Parametric Optimization

We consider the optimal joint trajectories to be the solution of the following bound-constrained optimization with parameter $\mathbf{p}$.

$$\boldsymbol{\xi}^{*}(\mathbf{p}) = \arg\min_{\boldsymbol{\xi}} f(\boldsymbol{\xi}, \mathbf{p}) \tag{9a}$$

$$\boldsymbol{\xi}_{lb} \leq \boldsymbol{\xi} \leq \boldsymbol{\xi}_{ub} \tag{9b}$$

We are interested in computing the Jacobian of $\boldsymbol{\xi}^*(\mathbf{p})$ with respect to $\mathbf{p}$. If we ignore the bound-constraints for now, we can follow the approach presented in [4] to obtain them in the following form.

$$\nabla_{\mathbf{p}}\boldsymbol{\xi} = -(\nabla^2_{\boldsymbol{\xi}} f(\boldsymbol{\xi}, \mathbf{p}))^{-1} \left[\nabla_{\boldsymbol{\xi}, p_1} f(\boldsymbol{\xi}, \mathbf{p}), \quad \dots \quad \nabla_{\boldsymbol{\xi}, p_n} f(\boldsymbol{\xi}, \mathbf{p})\right] \tag{10}$$

Using (10), we can derive a local model for the optimal solution corresponding to a perturbation $\Delta\mathbf{p}$ as

$$\boldsymbol{\xi}^*(\overline{\mathbf{p}}) = \boldsymbol{\xi}^*(\mathbf{p}) + \nabla_{\mathbf{p}}\boldsymbol{\xi}^* \overbrace{(\overline{\mathbf{p}} - \mathbf{p})}^{\Delta\mathbf{p}}, \tag{11}$$

Intuitively, (11) signifies a step of length $\Delta\mathbf{p}$ along the gradient direction. However, for (11) to be valid, the step-length needs to be small. In other words, the perturbed parameter $\overline{\mathbf{p}}$ needs to be in the vicinity of $\mathbf{p}$. Although it is difficult to mathematically characterize the notion of "small", in the following, we attempt a practical definition based on the notion of optimal cost.

Definition 1. *A valid* $|\Delta\mathbf{p}|$ *is one that satisfies the following relationship*

$$f(\boldsymbol{\xi}^*(\overline{\mathbf{p}} = \mathbf{p} + \Delta\mathbf{p}), \mathbf{p} + \Delta\mathbf{p}) \leq f(\boldsymbol{\xi}^*, \mathbf{p} + \Delta\mathbf{p}) \tag{12}$$

The underlying intuition in (12) is that the perturbed solution should lead to a lower cost for the parameter $\mathbf{p} + \Delta\mathbf{p}$ as compared to $\boldsymbol{\xi}^*$ for the same perturbed parameter.

2.3. Line Search and Incremental Adaption

Algorithm 1 couples the concept from the definition (11) with a basic line-search to incrementally adapt (11) to a large $\Delta\mathbf{p}$. The algorithm begins by initializing the optimal solution ${}^k\boldsymbol{\xi}$ and the parameter ${}^k\mathbf{p}$ with prior values for iteration $k = 0$. These variables are then used to initialize the Hessian and Jacobian matrices. The core computations takes place in line 2, wherein we compute the least amount of scaling that needs to be done to step length ${}^k\Delta\mathbf{p} = {}^k\overline{\mathbf{p}} - \mathbf{p}$ to guarantee a reduction in the cost. At line 3, we update the optimal solution based on step-length $\eta^k\Delta\mathbf{p}$ obtained in line 2, followed by a simple projection at line 4 to satisfy the minimum and maximum bounds. At line 5, we perform the called forward roll-out of the solution to update the parameter set. For example, if the parameter $\mathbf{p}$ models position of the end-effector at the final time instant of a trajectory, then line 5 computes how close the ${}^{k+1}\boldsymbol{\xi}^*$ takes the end-effector to the perturbed goal position $\overline{\mathbf{p}}$. On lines 7 and 8, we update the Hessian and the Jacobian matrices based on the updated parameter set and optimal solution.

3. Task Constrained Joint Trajectory Optimization

This section formulates various examples of the task-constrained trajectory optimization problem and uses the previous section's results for optimal adaptation of joint trajectories under task perturbation. To formulate the underlying costs, we adopt the way-point parametrization and represent the joint angles at time t as $\mathbf{q}_t$. Furthermore, we will use $(\mathbf{x}_e(\mathbf{q}_t), \mathbf{o}_e(\mathbf{q}_t))$ to describe the end-effector position and orientation in terms of Euler angles, respectively.

3.1. Orientation Constrained Interpolation between Joint Configurations

The task here is to compute an interpolation trajectory between a given initial $\mathbf{q}_0$ and a final joint configuration $\mathbf{q}_m$ while maintaining a specified orientation $\mathbf{o}_d$ for the end-effector at all times. We model it through the following cost function.

$$\sum_t f_s(\mathbf{q}_{t-k:t}) + \left\| \begin{matrix} \mathbf{q}_{t_1} - \mathbf{q}_0 \\ \mathbf{q}_{t_m} - \mathbf{q}_m \end{matrix} \right\|^2_2 + \sum_t \|\mathbf{o}_e(\mathbf{q}_t) - \mathbf{o}_d\|^2_2 \tag{13}$$

The first term the cost function models smoothness in terms of joint angles from $t-k$ to t [15]. For example, for $k=1$, the smoothness is defined as the first-order finite difference of the joint positions at subsequent time instants. Similarly, $k=2,3$, will model higher order smoothness through second and third-order finite differences respectively. We consider all three finite-differences in our smoothness cost term. The second term ensures that the interpolation trajectory is close to the given initial and final points. The final term in the cost function maintains the required orientation of the end-effector.

We can shape (13) in the form of (9a) by defining $\boldsymbol{\xi} = (\mathbf{q}_{t_1}, \mathbf{q}_{t_2}, \dots \mathbf{q}_{t_m})$. The bounds will correspond to the maximum and minimum limits on the joint angles at each time instant. We define the parameter set as $\mathbf{p} = (\mathbf{q}_0, \mathbf{q}_m)$. That is, we are interested in computing the adaptation when either or both of $\mathbf{q}_0$ and $\mathbf{q}_m$ gets perturbed.

Applications

Adaptation of $\boldsymbol{\xi}^*$ of (13) for different $\mathbf{q}_0, \mathbf{q}_m$ has applications in learning from demonstration setting where the human just provides the information about the initial and/or final joint configuration, and the manipulator then computes a smooth interpolation trajectory between the boundary configurations by adapting a prior computed trajectory.

Figure 1 presents an example of adaptation discussed above. The prior computed trajectory is shown in blue. This is then adapted to two different final joint configurations. The trajectory computed through Algorithm 1 is shown in green, while that obtained by resolving the optimization problem (with warm-starting) is shown in red.

3.2. Orientation-Constrained Trajectories through Way-Points

The task in this example is to make the end-effector move though given way-points while maintaining the orientation at $\mathbf{o}_d$. Let $\mathbf{x}_{d_t}$ represent the desired way-point of the end-effector at time t. Thus, we can formulate the following cost function for the current task.

$$\sum_t f_s(\mathbf{q}_{t-k:t}) + \sum_t \|\mathbf{o}_e(\mathbf{q}_t) - \mathbf{o}_d\|_2^2 + \sum_t \|\mathbf{x}_e(\mathbf{q}_t) - \mathbf{x}_{d_t}\|_2^2 \tag{14}$$

The first two terms in the cost function are the same as the previous example. The changes appear in the final term which minimizes the l_2 norm of the distance of the end-effector with the desired way-point. The defintion of $\boldsymbol{\xi}$ remains the same as before. However, the parameter set is now defined as $\mathbf{p} = (\mathbf{x}_{d_1}, \mathbf{x}_{d_2}, \dots \mathbf{x}_{d_m})$.

Application

Collision Avoidance As shown in Figure 2, a key application of the adaptation problem discussed above is in collision avoidance. A reactive planner such as [16] can provide new via-points for the manipulator to avoid collision. Our Algorithm 1 can then use the cost function (14) to adapt the prior trajectory shown in blue to that shown in green. For comparison, the trajectory obtained with resolve of the trajectory optimization is shown in red.

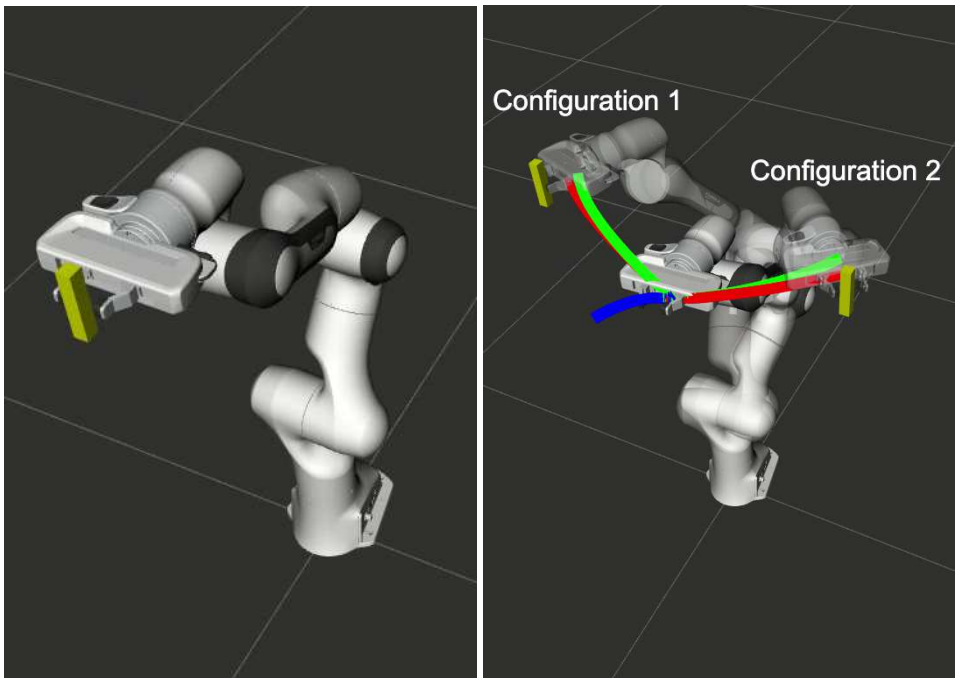

Figure 1. Prior trajectory shown in blue is used to adapt the joint motions to move towards two different final joint configurations while maintaining the horizontal orientation of the end-effector at all times.

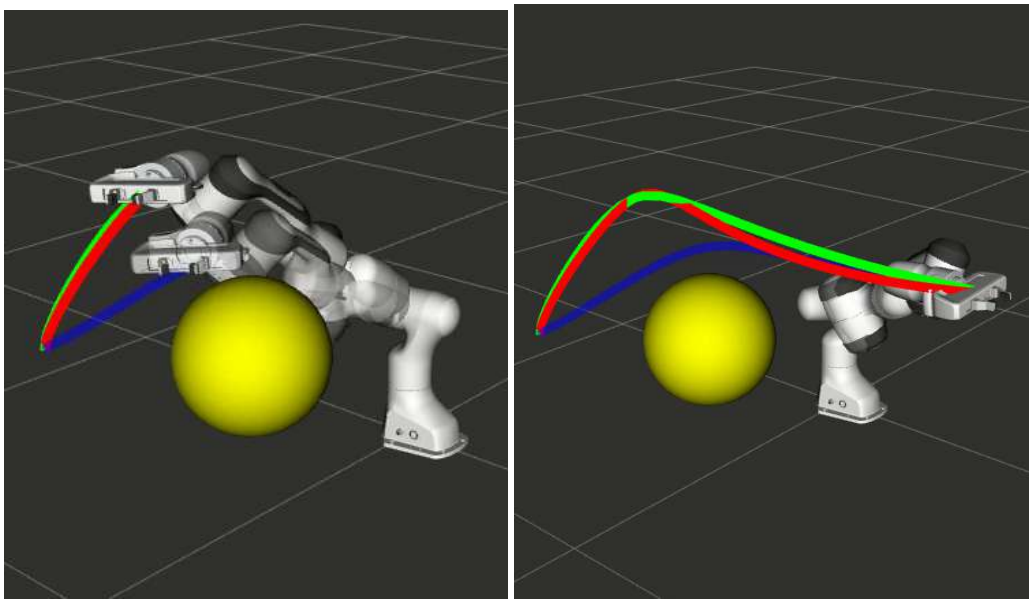

Figure 2. Collision avoidance by perturbing the mid-point of the prior computed end-effector trajectory.

Human–Robot Handover: Algorithm 1 with cost function (14) also finds application in human–robot handover tasks. An example is shown in Figure 3, where the manipulator adapts the prior trajectory (blue) to a new estimate of the handover position. As before, the trajectory obtained with Algorithm 1 is shown in green, while the one shown in red corresponds to a re-solve of the trajectory optimization with warm-start initialization.

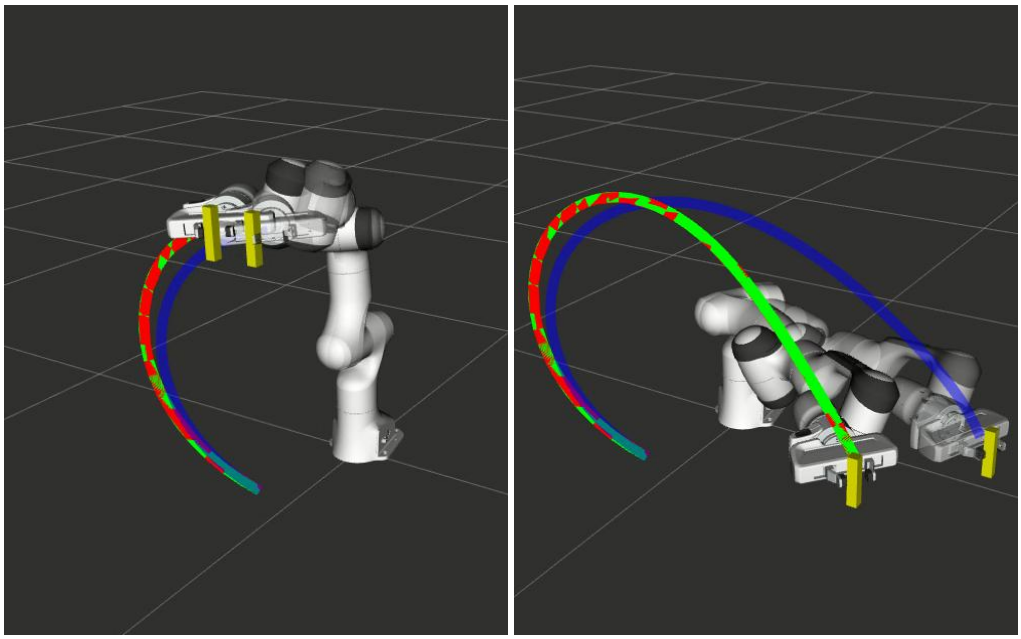

Figure 3. Perturbation in the final position of the end-effector.

4. Benchmarking

4.1. Implementation Details

The objective of this section is to compare the trajectories computed by Algorithm 1 with that obtained by re-solving the trajectory optimization for the perturbed parameters with warm-start initialization. We consider the same three benchmarks presented in Figures 1–3 implemented on a 7dof Franka Panda Arm, but for a diverse range of perturbations magnitude. For each benchmark, we created a data set of 180 trajectories by generating random perturbations in the task parameters. For the benchmark of Figure 1, the parameters are the joint angles, but in the following we use the forward kinematics to derive equivalent representation for the parameters in terms of end-effector position values.

Each joint trajectory is parameterized by a 50-dimensional vector of way-points. Thus, the underlying task constrained trajectory optimization involves a total of 350 variables. We use Scipy-SLSQP [3] to obtain the prior trajectory and also to re-solve the trajectory optimization for the perturbed parameters. We did our implementation in Python using Jax-Numpy [17] to compute the necessary Jacobian and Hessian matrices. We also used the just-in-time compilation ability of JAX to create an on-the-fly compiled version of our codes. The line-search in Algorithm 1 (line 2) was done through a parallelized search over a set of discretized η values. The entire implementation was done on a 32 GB RAM i7-8750 desktop with RTX 2080 GPU (8GB). To foster further research in this field and ensure reproducibility, we open-source our implementation for review at https://rebrand.ly/argmin-planner (First released on 15 September 2020).

4.2. Quantitative Results

Orientation Metric: For this analysis, we compared the pitch and roll angles at each time instant along trajectories obtained with Algorithm 1 and the resolving approach. Specifically, we computed the maximum of the absolute difference (or L_∞ norm) of the two orientation trajectories. The yaw orientation in all these benchmarks was a free variable and is thus not included in the analysis. The results are summarized in Figures 4–6. The histogram plot in these figures are generated for the medium perturbation ranges (note the figure legends). For the Figure 1 benchmark related to cost function (13), Figure 4 shows that all the trajectories obtained by Algorithm 1 have L_∞ norm of the orientation difference less than 0.1 rad. For the benchmark of Figure 2, which we recall involves perturbing the via-point of the end-effector trajectory, the histograms of Figure 5 show similar trends. All the trajectories computed by Algorithm 1 managed a similar orientation difference. For the benchmark of Figure 3 pertaining

to the perturbation of the final position, 69.41% of the trajectories obtained by Algorithm 1 managed to maintain a orientation difference of 0.1 rad with the resolving approach.

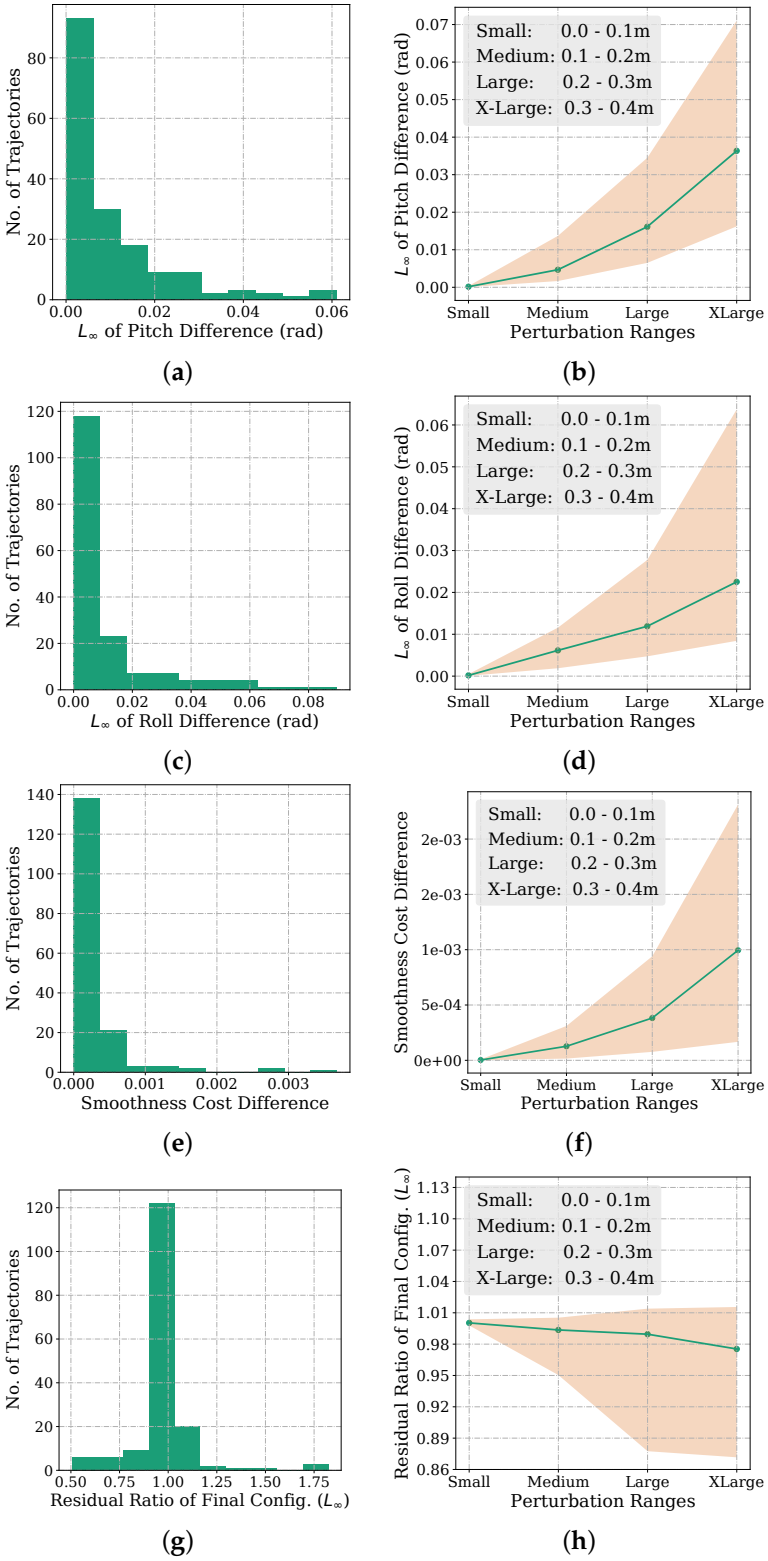

Figure 4. Performance of Algorithm 1 for different perturbation ranges on the benchmark of Figure 1 that involves perturbing the final joint configuration (recall cost function (13)). Note that the perturbation in the final joint is converted to position values by forward kinematics. The (**a**,**c**,**e**,**g**) column shows the histogram of orientation, smoothness and task residual ratio metrics for the medium range perturbation. The (**b**,**d**,**f**,**h**) column quantifies the metrics for different perturbation ranges.

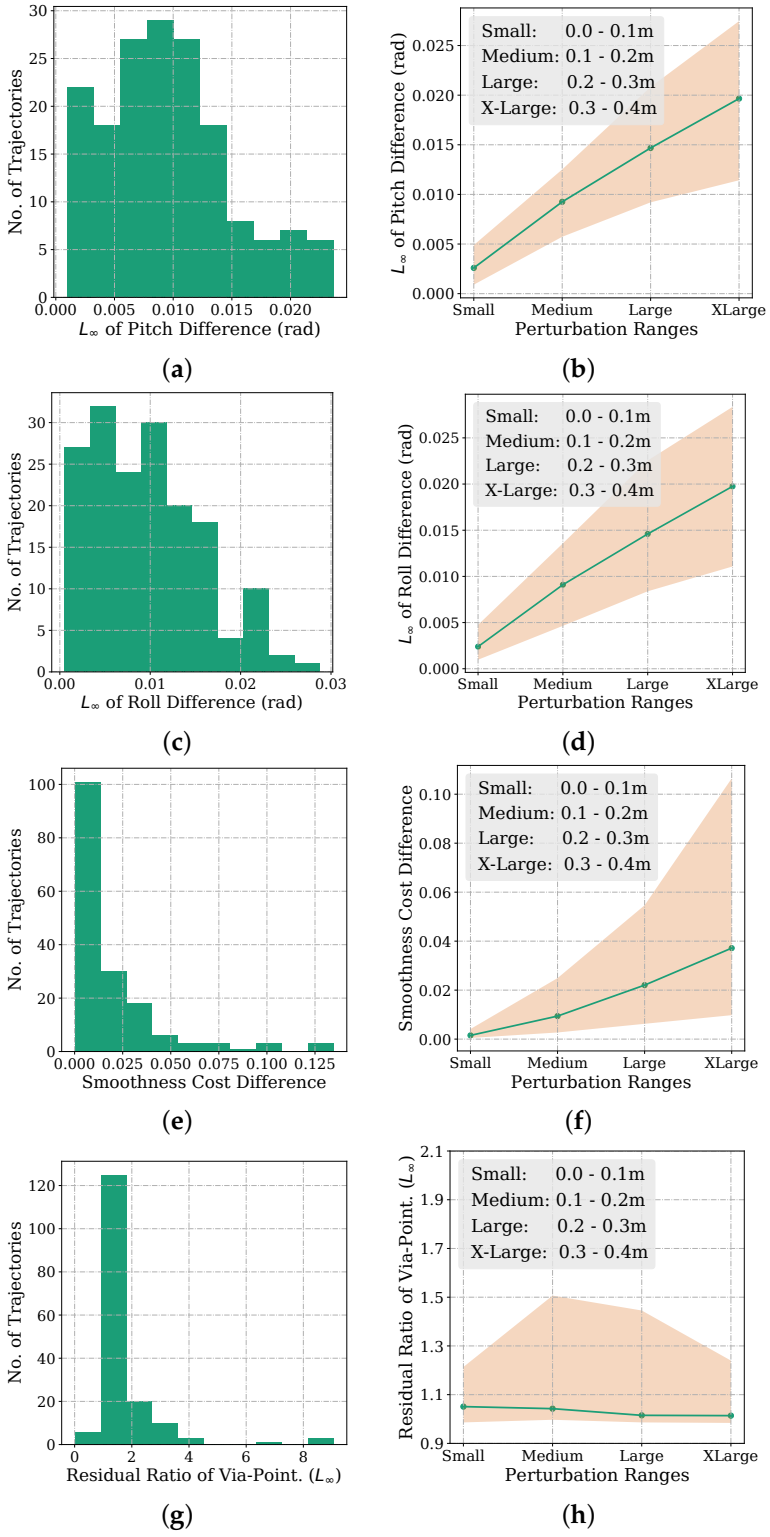

Figure 5. Performance of Algorithm 1 for different perturbation ranges on the benchmark of Figure 2 that involves perturbing the via-point of the end-effector trajectory (recall cost function (14)). The (**a**,**c**,**e**,**g**) and (**b**,**d**,**f**,**h**) columns show similar benchmarking as those of Figure 4.

Task residuals ratio metric: For this analysis, we compare the task residual between trajectories obtained from Algorithm 1 and the resolving approach. For example, for the benchmark of Figure 1, we want the manipulator final configuration to be close to the specified value (recall cost (13)) while maintaining the desired orientation at each time instant. Thus, we

compute the L_∞ residual of $\mathbf{q}_t - \mathbf{q}_m$ for Algorithm 1 and compare it with that obtained from the resolving approach. Now, as previously, and to be consistent with the other benchmarks, we convert the residual of the joint angles to position values through forward kinematics. Similar analysis follow for the other benchmarks as well. For the ease of exposition, we divide the task residual of Algorithm 1 by that obtained with the resolving approach. A ratio greater than 1 implies that the former led to a higher task residual than the latter and vice-versa. Similarly, a ratio closer to 1 implies that both the approaches performed equally well.

The results are again summarized in Figures 4–6. From Figure 4, we notice that 97.05% of trajectories have a residual ratio less than 1.2. For the experiment involving via-point perturbation in Figure 5, the performance drops to 62.50% for the same value of residual ratio. Meanwhile, as shown in Figure 6, around 82.94% of the trajectories have a residual ratio less than 1.2 in the case of the final position perturbation benchmark of Figure 3.

Velocity Smoothness Metric: For this analysis, we computed the difference in the velocity smoothness cost (L_2 norm of first-order finite difference) between the trajectories obtained with Algorithm 1 and the resolving approach. The results are again summarized in Figures 4–6. For all the benchmarks, in around 65% of the examples, the difference was less than 0.05. This is 35% of the average smoothness cost observed across all the trajectories from both the approaches.

Scaling with Perturbation Magnitude: The line plots in Figures 4–6 represent the first quartile, median and the third quartile of the three metrics discussed above for different perturbation ranges.

For the benchmark of Figure 1, trajectories from Algorithm 1 maintains an orientation difference of less than 0.1 rad, with the trajectories of the resolving approach for perturbations as large as 40 cm. The difference in smoothness cost for the same range is also small, with the median value being in the order of 10^{-3}. The median task residuals achieved by Algorithm 1 is only 2% higher than that obtained by the resolving approach. For the benchmark of Figure 2, the performance remains same on the orientation metric, but the median difference in smoothness cost and task residual ration increases to 0.04 and 9% for the largest perturbation range. The benchmark of Figure 6 follows a similar trend in orientation and smoothness metric, but performs significantly worse in task residuals. For the largest perturbation range, Algorithm 1 leads to 50% higher median task residuals. However, importantly, for perturbation up to 30 cm, the task residual ratio is close to 1, suggesting that Algorithm 1 performed as well as the resolving approach for these perturbations.

Computation Time: Table 1 contrasts the average timing of our Algorithm 1 with the approach of resolving the trajectory optimization with warm-start initialization. As can be seen, our Argmin differentiation based approach provides a worst-case speed up of 160x on the benchmark of Figure 3. For the rest of the benchmarks, this number varies between 500 to 1000. We believe that this massive gain in computation time offsets whatever little performance degradation in terms of orientation, smoothness, and task residual metric that Algorithm 1 incurs compared to re-solving the problem using warm-start. Note that the high computation time of the re-solving approach is expected, given that we are solving a difficult non-convex function over a long horizon of 50 steps resulting in 350 decision variables. Even highly optimized planners like [1] show similar timings on closely related benchmarks [18].

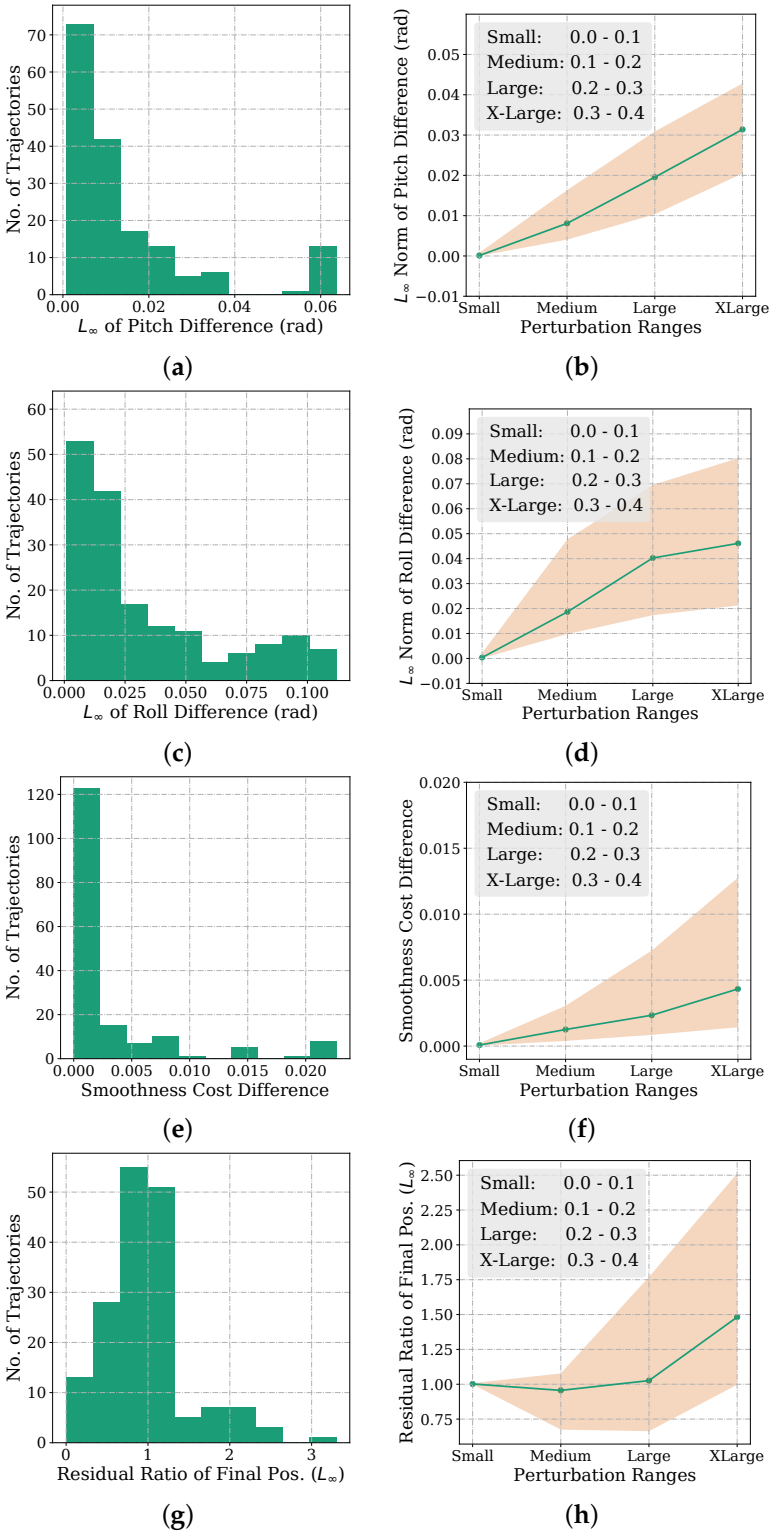

Figure 6. Performance of Algorithm 1 for different perturbation ranges on the benchmark of Figure 6 that involves perturbing the final end-effector position. (recall cost function (14)). The (**a**,**c**,**e**,**g**) and (**b**,**d**,**f**,**h**) columns show similar benchmarking as those of Figure 4.

Table 1. Computation times comparison between Algorithm 1 and resolving trajectory optimization approach on three benchmarks.

Benchmarks	SciPy-SLSQP		Our Algorithm 1
	Wall Time (s)	Wall Time w/o Jacobian and Function Evaluation Overhead (s)	Wall Time (s)
Final Configuration Perturbation (Figure 1)	43.91	41.09	**0.039**
Via Point Perturbation (Figure 2)	53.05	34.74	**0.09**
Final Position Perturbation (Figure 3)	35.91	29.09	**0.18**

5. Conclusions and Future Work

We presented a fast, near real-time algorithm for adapting joint trajectories to task perturbation as high as 40 cm in the end-effector position, almost half the radius of the Franka Panda arm's horizontal workspace used in our experiments. By consistently producing trajectories similar to those obtained by resolving the trajectory optimization problem but in a small fraction of a time, our Algorithm 1 opens up exciting possibilities for reactive motion control of manipulators in applications like human–robot handover.

Our algorithm is easily extendable to other kind of manipulators. The only requirement is that we should know the forward kinematics of the manipulator. This would allow us to get the algebraic expressions for functions $\mathbf{o}_e(\mathbf{q})$ and $\mathbf{x}_e(\mathbf{q})$ in cost function (13) and (14), respectively. In our implementation, we derived the forward kinematics and $\mathbf{o}_e(\mathbf{q})$ and $\mathbf{x}_e(\mathbf{q})$ through the DH representation of the manipulator. The DH table is available for many commercial manipulators, e.g., UR5e besides the Franka Panda Arm used in our simulation.

Our algorithm does not depend on any specific sensing modality. For example, in collision avoidance applications, we assume that obstacle information is used by some higher level planners that provides intermediate collision-free points to the manipulator, which then uses the ArgMin differentiation to replan its prior trajectories.

There are several ways to improve our algorithm. First, the joint bounds can also be included as penalties in the cost function itself, in addition to being handled by projection (Line 11 in Algorithm 1). This would ensure that the gradient and Hessian of the optimal cost is aware of the joint limit bounds. Second, we can consider a low dimensional polynomial representation of the trajectories. For example, the joint trajectories can be represented by a 10th order Bernstein polynomial with the coefficients acting as the variables of the optimization problem. This would drastically reduce the computation cost of obtaining the Hessian of the optimal cost as compared to current way-point paramaterization of the joint trajectory that requires around 50 variables to represent one joint trajectory.

In future works, we will extend our formulation to problems with dynamic constraints, such as torque bounds. We conjecture that by coupling the way-point parametrization with a multiple-shooting like approach, we can retain the constraints as simple box-bounds on the decision variables and consequently retain the computational structure of the Algorithm 1. We are also currently evaluating our algorithm's performance on applications such as autonomous driving.

Author Contributions: Conceptualization, S.S., M.B., A.K.S. and K.M.K.; Methodology, S.S., M.B., A.K.S. and K.M.K.; Software, S.S., M.B. and H.M.; Validation, S.S., M.B. and H.M.; formal analysis, S.S. and M.B.; investigation, S.S., M.B. and A.K.S.; resources, S.S., M.B. and A.K.S.; data curation, S.S., M.B. and H.M.; writing—original draft preparation, A.K.S., S.S., M.B., K.K. and K.M.K.; writing—review and editing, S.S., M.B., K.K. and K.M.K.; visualization, S.S., M.B. and H.M.; supervision, A.K.S. and K.M.K.; project administration, A.K.S. and K.M.K.; funding acquisition, A.K.S. and K.K. All authors have read and agreed to the published version of the manuscript.

Funding: The work was supported in part by thw European Social Fund via the "ICT programme" measure, by grant PSG753 from the Estonian Research Council, by AI & Robotics Estonia (AIRE),

the Estonian candidate for European Digital Innovation Hub, funded by the Ministry of Economic Affairs and Communications in Estonia.

Data Availability Statement: Codes to reproduce the results are available at https://rebrand.ly/argmin-planner (accessed on 1 August 2020).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Berenson, D.; Srinivasa, S.S.; Ferguson, D.; Kuffner, J.J. Manipulation planning on constraint manifolds. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 625–632.
2. Lembono, T.S.; Paolillo, A.; Pignat, E.; Calinon, S. Memory of motion for warm-starting trajectory optimization. *IEEE Robot. Autom. Lett.* **2020**, *5*, 2594–2601. [CrossRef]
3. Virtanen, P.; Gommers, R.; Oliphant, T.E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nat. Methods* **2020**, *17*, 261–272. [CrossRef] [PubMed]
4. Gould, S.; Fernando, B.; Cherian, A.; Anderson, P.; Cruz, R.S.; Guo, E. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv* **2016**, arXiv:1607.05447.
5. Hauser, K. Learning the problem-optimum map: Analysis and application to global optimization in robotics. *IEEE Trans. Robot.* **2016**, *33*, 141–152. [CrossRef]
6. Tang, G.; Sun, W.; Hauser, K. Time-Optimal Trajectory Generation for Dynamic Vehicles: A Bilevel Optimization Approach. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 7644–7650.
7. Reiter, A.; Gattringer, H.; Müller, A. Real-time computation of inexact minimum-energy trajectories using parametric sensitivities. In Proceedings of the International Conference on Robotics in Alpe-Adria Danube Region, Torino, Italy, 21–23 June 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 174–182.
8. Geffken, S.; Büskens, C. Feasibility refinement in sequential quadratic programming using parametric sensitivity analysis. *Optim. Methods Softw.* **2017**, *32*, 754–769. [CrossRef]
9. Pirnay, H.; López-Negrete, R.; Biegler, L.T. Optimal sensitivity based on IPOPT. *Math. Program. Comput.* **2012**, *4*, 307–331. [CrossRef]
10. Amos, B.; Jimenez, I.; Sacks, J.; Boots, B.; Kolter, J.Z. Differentiable MPC for end-to-end planning and control. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2018; pp. 8289–8300.
11. Agrawal, A.; Barratt, S.; Boyd, S.; Stellato, B. Learning convex optimization control policies. In Proceedings of the 2nd Conference on Learning for Dynamics and Control, PMLR, Berkeley, CA, USA, 11–12 June 2020; pp. 361–373.
12. Landry, B.; Lorenzetti, J.; Manchester, Z.; Pavone, M. Bilevel Optimization for Planning through Contact: A Semidirect Method. *arXiv* **2019**, arXiv:1906.04292.
13. Kalantari, H.; Mojiri, M.; Dubljevic, S.; Zamani, N. Fast l_1 model predictive control based on sensitivity analysis strategy. *IET Control. Theory Appl.* **2020**, *14*, 708–716. [CrossRef]
14. Pham, Q.C. A general, fast, and robust implementation of the time-optimal path parameterization algorithm. *IEEE Trans. Robot.* **2014**, *30*, 1533–1540. [CrossRef]
15. Toussaint, M. A tutorial on Newton methods for constrained trajectory optimization and relations to SLAM, Gaussian Process smoothing, optimal control, and probabilistic inference. In *Geometric and Numerical Foundations of Movements*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 361–392.
16. Flacco, F.; Kröger, T.; De Luca, A.; Khatib, O. A depth space approach to human-robot collision avoidance. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, Guangzhou, China, 11–14 December 2012; pp. 338–345.
17. Bradbury, J.; Frostig, R.; Hawkins, P.; Johnson, M.J.; Leary, C.; Maclaurin, D.; Wanderman-Milne, S. JAX: Composable Transformations of Python+NumPy Programs. 2018. Available online: http://github.com/google/jax (accessed on 1 August 2020).
18. Qureshi, A.H.; Dong, J.; Baig, A.; Yip, M.C. Constrained Motion Planning Networks X. *arXiv* **2020**, arXiv:2010.08707.