Shadow Program Inversion with Differentiable Planning: A Framework for Unified Robot Program Parameter and Trajectory Optimization

Benjamin Alt[1,2,*,†], Claudius Kienle[1,*], Darko Katic[1], Rainer Jäkel[1] and Michael Beetz[2]

***Abstract*— This paper presents Shadow Program Inversion with Differentiable Planning (SPI-DP), a novel first-order optimizer capable of optimizing robot programs with respect to both high-level task objectives and motion-level constraints. To that end, we introduce Differentiable Gaussian Process Motion Planning for N-DoF Manipulators (dGPMP2-ND), a differentiable collision-free motion planner for serial N-DoF kinematics, and integrate it into an iterative, gradient-based optimization approach for generic, parameterized robot program representations. SPI-DP allows first-order optimization of planned trajectories and program parameters with respect to objectives such as cycle time or smoothness subject to e.g. collision constraints, while enabling humans to understand, modify or even certify the optimized programs. We provide a comprehensive evaluation on two practical household and industrial applications.**

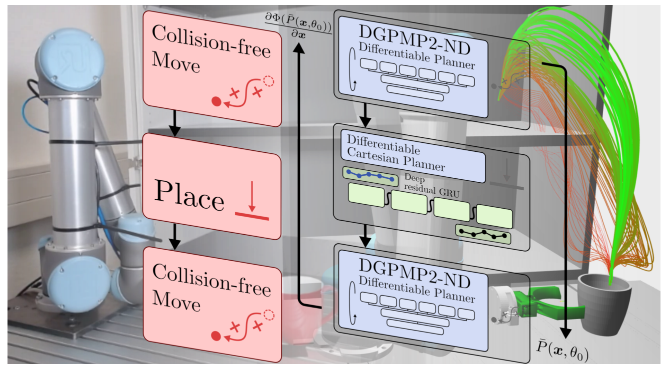

Fig. 1. Shadow Program Inversion with Differentiable Planning (SPI-DP) enables the optimization of robot programs (left, red) by first-order iterative optimization over a differentiable surrogate (right, gray). A differentiable collision-free motion planner (dGPMP2-ND) ensures that the resulting motion trajectories are optimal with respect to task objectives and motion-level constraints.

I. Introduction

Intuitive robot programming has eased the use of robots to solve real-world applications. However, the cost of automation is often driven by the iterative optimization of robot trajectories and program parameters, particularly for complex manipulation tasks. This optimization is done by skilled programmers through time-consuming trial and error. "Programming by optimization" [1] allows a human programmer to specify a rough program skeleton, which is then completed by an optimization algorithm. This approach is particularly useful in tactile applications like force-controlled insertion or handling of deformable objects. However, applying general-purpose optimization methods to robot programs is challenging as the success of a robot skill depends on the parameterization of the preceding skills: An optimizer must *jointly* optimize the parameters of complete skill sequences or hierarchically composed subprograms. Moreover, robots must not only achieve task-level goals such as cycle time requirements, but also respect motion-level constraints such as collision-freeness or proximity to a human demonstration. Existing approaches focus exclusively on either trajectory optimization [2]–[5] or parameter optimization [6]–[9] and typically optimize individual skills, rather than jointly optimizing complete robot programs.

In this paper, we propose Shadow Program Inversion with Differentiable Planning (SPI-DP), a robot program optimizer which combines both trajectory and parameter optimization with respect to task- and motion-level constraints. We make the following contributions:

1) We present **Differentiable Gaussian Process Motion Planning for N-DoF Manipulators (dGPMP2-ND)**, a **differentiable motion planner** for serial N-degree of freedom (DoF) manipulators, capable of propagating gradients through the collision-free planning procedure.[1]
2) We introduce **SPI-DP**, an approach for **unifying program parameter and trajectory optimization** capable of optimizing robot programs with respect to a wide range of objective functions, including task-specific metrics and goals demonstrated by humans.
3) We provide a **real-world evaluation** of the proposed framework on household pick-and-place as well as industrial peg-in-hole applications.

To our knowledge, SPI-DP is the first approach to combine parameter and trajectory optimization for robot programs in one unified framework.

II. Related Work

1) Robot program parameter optimization: In the context of "programming by optimization" [1], a wide array of optimizers have been proposed, with a majority employing zero-order algorithms such as evolutionary or mutation-based algorithms [10]–[14], particle swarms [11], [15], [16] or Bayesian optimization [8], [16]–[18]. First-order optimizers

This work was supported by the German Federal Ministry of Education and Research (grant 01MJ22003B), the DFG CRC EASE (CRC #1320) and the EU project euROBIN (grant 101070596).

*Authors contributed equally to this paper
†Corresponding author: benjamin.alt@uni-bremen.de
[1]ArtiMinds Robotics, Karlsruhe, Germany
[2]Institute for Artificial Intelligence, University of Bremen, Germany

[1]The source code for dGPMP2-ND is available at https://github.com/benjaminalt/dgpmp2-nd.

propose to leverage gradient information for fast, stable convergence. Differentiable programming (∂P) proposes to represent programs as differentiable computation graphs (DCGs), which permit gradient computation for program parameters via automatic differentiation [19]–[21]. In robotics, ∂P has been primarily used to optimize control parameters in conjunction with differentiable physics engines [22]–[26], or as differentiable policies in reinforcement learning [27]–[30]. Our approach proposes to represent robot programs as DCGs, comprising differentiable planners and artificial neural networks (ANNs), for first-order optimization of program parameters.

2) Trajectory optimization: Program parameter optimization has typically been considered separately from the optimization of motion trajectories. First-order methods such as CHOMP [2] promise fast convergence due to the exploitation of gradient information [4], [31]–[34]. Gaussian Process Motion Planning (GPMP) [35]–[38] represents robot trajectories as a Gaussian process (GP) and realizes optimization by iteratively minimizing an objective function comprising smoothness and collision constraints. We generalize Differentiable Gaussian Process Motion Planning (dGPMP2) [37], a first-order extension of GPMP, to N-DoF serial kinematics, add additional constraints such as human demonstrations and integrate it as a gradient-preserving path planner inside a first-order program optimizer. To our knowledge, we contribute the first gradient-based framework for jointly optimizing robot program parameters and motion trajectories.

III. Shadow Program Inversion: A Primer

Our proposed framework is based on differentiable Shadow Program Inversion (SPI), a model-based first-order optimizer for robot program parameters [7], [39], [40], which is briefly outlined below. On the basis of SPI and a differentiable N-DoF motion planner (see Sec. IV), we present a novel double-loop first-order optimizer capable of jointly optimizing program parameters and motion trajectories (see Sec. V).

A. Differentiable Shadow Programs

The core of our framework is the concept of a *shadow program*: A differentiable "twin" of a robot program which serves as its surrogate for learning and optimization. The *source program*, which is written by the programmer and ultimately executed on the robot, and its shadow used for learning and optimization, are representationally decoupled: The source program can be expressed in any parameterized representation, such as the textual or skill-based representations used by modern robot programming frameworks [41]–[43].

Given a source program $P(\boldsymbol{x}, \theta_0)$, we seek to optimize the program parameters $\boldsymbol{x}$, given initial robot state θ_0. P is modeled as a function $P : \mathbb{R}^N \times \Theta \rightarrow \Theta^T$, mapping a real-valued N-dimensional parameter vector $\boldsymbol{x} \in \mathbb{R}^N$ and joint angles θ in state space Θ to the robot trajectory $\boldsymbol{\theta} \in \Theta^T$, where T is the number of timesteps. The state space is composed of robot joint configurations and end-effector wrenches: $\Theta = \mathcal{C} \times \mathbb{R}^6$.

The *shadow program* $\bar{P}$ is a generative model of P, trained to approximate the real-world robot trajectory $\boldsymbol{\theta}$ for a given set of program parameters $\boldsymbol{x}$ and initial state θ_0 [7]. A central property of $\bar{P}$ is that it is differentiable, allowing the computation of the gradient of a task-specific objective Φ over the trajectory with respect to the program's input parameters, and, as a consequence, the optimization of $\boldsymbol{x}$ using a gradient-based optimizer.

The shadow program architecture is modular, to reflect the typically skill-based structure of most source programs. An exemplary shadow program composed of two skills is illustrated in Fig. 4. For the purpose of this paper, it is important to note that shadow skills are generative models of robot skills, predicting the expected real-world robot trajectory given the skill's parameters and the current robot state. A *differentiable planner* bootstraps a naive *prior trajectory*, which is then refined by a neural sequence-to-sequence model to reflect expected real-world deviations from the plan (see [7] for details).

B. Robot Program Parameter Optimization

Differentiable shadow programs enable parameter optimization for near-arbitrary source programs in any parameterized representation. Consider a skill-based robot program (here in pseudocode) for an industrial peg-in-hole task:

```
MoveArm(approach_pose)
SpiralSearch(spiral_extents, contact_force)
Insert(depth, pushing_force)
```

`SpiralSearch` and `Insert` are skills from a skill library, and `approach_pose` is a program parameter corresponding to the end-effector pose from which the robot starts searching for a hole in the workpiece. The duration of the search depends on the position of the hole relative to the approach pose. Consequently, the cycle time of this program can be optimized by adapting `approach_pose` to be, on average, directly above the hole. SPI solves such parameter optimization problems by first-order optimization over the shadow program, using $\bar{P}$ as a differentiable surrogate for P [7]. The optimized parameters can be transferred back to the source program P, validated and adjusted by a human programmer, and executed on the robot.

C. Joint Parameter and Trajectory Optimization

For many tasks, program parameters and low-level motion trajectories must be jointly optimized. One example is the optimization of grasp poses to maximize grasp success: The approach and depart motions must also be optimized for collision-freeness, smoothness and other task constraints whenever the grasp pose is changed. We integrate a differentiable motion planner into the shadow program representation, which ensures that trajectories predicted in a forward pass comply with motion-level constraints such as collision-freeness, smoothness or proximity to a human demonstration. The differentiable motion planner is described in detail in Section IV.

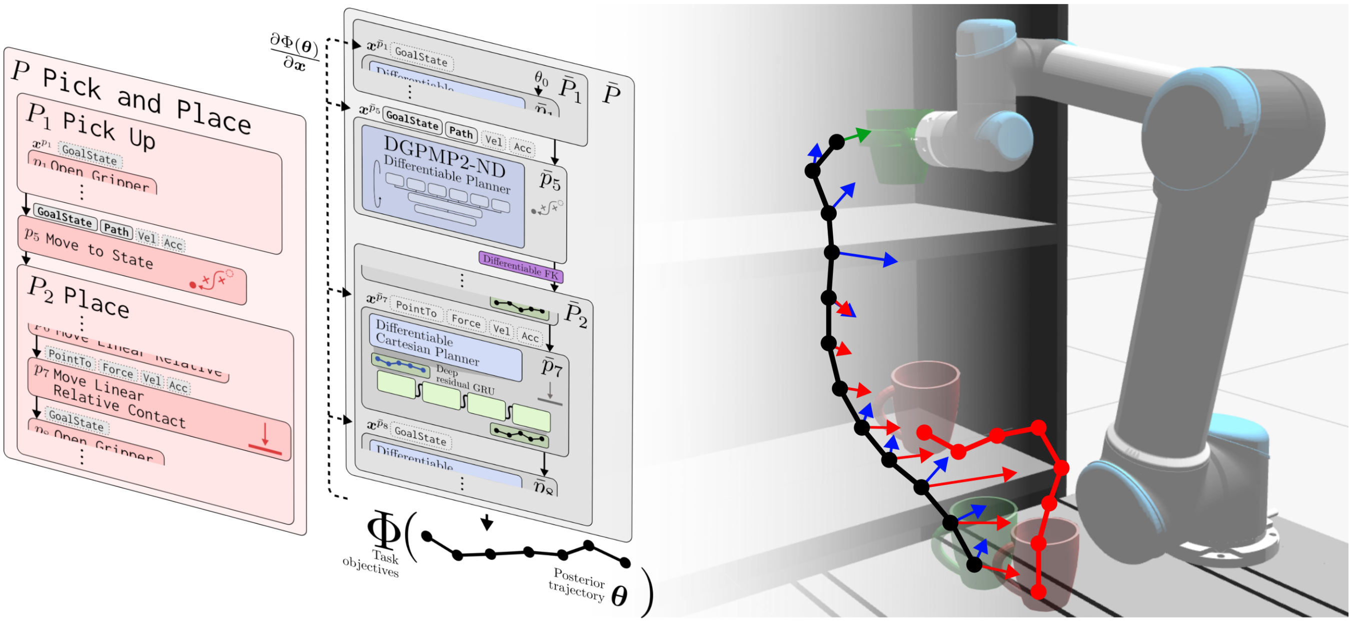

Fig. 2. Left: Shadow Program Inversion with Differentiable Planning (SPI-DP) optimizes robot programs (left, red) with respect to nearly arbitrary task objectives (Φ). Program parameters, such as the `Force` parameter of a placing action, can be optimized jointly with the low-level motion trajectories to respect task-level objectives and motion-level constraints. By performing gradient-based optimization over a differentiable surrogate ("shadow" $\bar{P}$, gray), the framework is applicable to near-arbitrary parameterized program representations, including most robot programming languages. Right: dGPMP2-ND plans collision-free motions for N-DoF serial kinematics within SPI-DP's optimization loop. Trajectories (black) are optimized with respect to collision (blue), goal (green) and human demonstration (red) constraints, among others.

IV. DGPMP2-ND: Differentiable Motion Planning for N-DoF Manipulators

The gradient-based optimization of programs containing collision-free motion skills requires a differentiable planner. We propose dGPMP2-ND, a differentiable collision-free motion planner for N-DoF manipulators, which generates trajectories that conform to motion constraints such as collision-freeness, smoothness, adherence to joint limits or precision at a target pose. To that end, we extend and modify dGPMP2 [37] by implementing differentiable collision checking for three-dimensional collision worlds and N-DoF serial kinematics, adding a joint limit constraint as well as a factor rewarding similarity to a human demonstration.

A. Differentiable Gaussian Motion Planning

dGPMP2 casts motion planning as inference on a factor graph [37] and minimizes a *cost functional* $\mathcal{F}(\boldsymbol{\theta})$ over trajectory $\boldsymbol{\theta}$ via an iterative optimization procedure [36].

Figure 3 illustrates dGPMP2-ND. While dGPMP2 plans in Cartesian space, dGPMP2-ND plans joint-space trajectories. This permits to integrate joint limit constraints, while still supporting end-effector pose constraints by applying a differentiable forward kinematics on the joint-space trajectory. At each planner iteration $j, 1 \leq j \leq j_{max}$, a set of *factors* is evaluated. Given the current joint trajectory $\boldsymbol{\theta}^j$, each factor computes an *error* $\boldsymbol{h}(\boldsymbol{\theta})$, a *Jacobian* $\boldsymbol{H}$ indicating the direction of steepest descent to minimize the error, and an *inverse covariance* $\boldsymbol{\Sigma}^{-1}$ to weight the different factors. We propose six such factors:

1) A *Gaussian process (GP) prior* factor, which penalizes points on the joint trajectory that deviate from the mean defined by a GP prior (see [36] for details). For each point on the trajectory, the Jacobian $\boldsymbol{H}_{GP}$ indicates the direction toward the GP mean.
2) A *start state prior*, which penalizes the deviation of the first point on $\boldsymbol{\theta}^j$ from a predefined start configuration. For the first point on $\boldsymbol{\theta}^j$, the Jacobian $\boldsymbol{H}_{start}$ indicates the direction toward the start configuration.
3) A *goal state prior*, which penalizes the deviation of the last point on $\boldsymbol{\theta}^j$ from a predefined goal configuration. For the last point on $\boldsymbol{\theta}^j$, the Jacobian $\boldsymbol{H}_{goal}$ indicates the direction toward the goal configuration.
4) A *collision factor* (see Sec. IV-B).
5) A *joint limit factor* (see Sec. IV-C).
6) A *demonstration prior* (see Sec. IV-D).

The GP, start and goal state priors remain unchanged from the original dGPMP2 formulation, albeit extended to the N-DoF case. We contribute novel differentiable collision and joint limit factors, as well as a differentiable Cartesian demonstration prior.

At each iteration, the linear system

$$(\mathcal{K}^{-1} + \boldsymbol{H}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{H})\delta\boldsymbol{\theta} = -\mathcal{K}^{-1}(\boldsymbol{\theta}^i - \boldsymbol{\mu}) - \boldsymbol{H}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{h}(\boldsymbol{\theta}^j)$$

is solved for $\delta\boldsymbol{\theta}$, where $\boldsymbol{H}$ is the combined Jacobian, $\boldsymbol{\Sigma}^{-1}$ is the combined inverse covariance, $\mathcal{K}^{-1}$ is the inverse kernel matrix of the GP and $\boldsymbol{h}(\boldsymbol{\theta}^j)$ is the combined error function. All matrices are combined by concatenating the matrices of the individual factors along the row axis. The trajectory is then incrementally updated: $\boldsymbol{\theta}^{j+1} = \boldsymbol{\theta}^j + r * \delta\boldsymbol{\theta}$, where r is

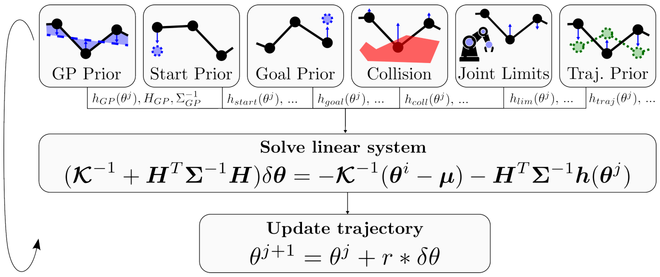

Fig. 3. Differentiable Gaussian Process Motion Planning for N-DoF Manipulators (dGPMP2-ND) permits motion planning with respect to joint-space and Cartesian constraints and objectives. It realizes trajectory optimization by iterative solving of a linear system, permitting the backpropagation of gradients through the planner.

the update rate.

As dGPMP2-ND is used as a differentiable planner inside another iterative optimization process, the total number of iterations required by dGPMP2-ND until convergence must be kept as small as possible. To that end, r is initalized at a high value (r=0.3), leaving it constant while the collision error $h_{coll} > 0$, and decay it by factor 0.1 at each iteration. Moreover, optimization is stopped before j_{max} is reached when h_{coll} has not decreased for 25 iterations or the total h has decreased by less than 5% for 25 iterations.

B. Differentiable N-DoF Collision Factor

Bhardwaj et al. [37] propose a collision factor for 2D environments and a point robot. We extend their approach to 3D environments, joint-space trajectories, and N-DoF serial robot kinematics. Before planning, we precompute a 3D signed distance field (SDF) of the environment, where each voxel contains the signed distance from the voxel center to the next obstacle. For all states on joint trajectory $\boldsymbol{\theta}$, we compute the Cartesian poses and Jacobians of all links using differentiable forward kinematics [44]. For each link and each time step, we identify the SDF voxel that intersects with the collision mesh of the link and has the smallest distance to the collision environment. To compute a differentiable error, we then take for each identified voxel the weighted mean of the 26 surrounding voxels, resulting in a vector pointing away from the nearest collision. The resulting Jacobian equals the matrix multiplication of the Jacobian for each link and the Jacobian for the differentiable error.

C. Joint Limit Factor

To ensure that joint-limit constraints of the manipulator are met, we extend dGPMP2 by a joint limit factor. For each state on the joint trajectory $\boldsymbol{\theta}$, the *joint limit error*

$$h_{lim} = \begin{cases} \theta - \theta_{lim} & \text{if} \quad \theta > \theta_{lim} \\ -\theta_{lim} - \theta & \text{if} \quad \theta < -\theta_{lim} \\ 0 & \text{otherwise} \end{cases}$$

penalizes trajectory states which exceed or fall below the joint limits $\boldsymbol{\theta}_{lim}$. $\boldsymbol{H}_{lim}$ is the identity matrix for values outside the limits, zero otherwise.

D. Demonstration prior

For many planning problems, human demonstrations can be leveraged to guide the planner toward good solutions, speeding up convergence. We extend dGPMP2 by a prior factor which penalizes trajectories that deviate from a reference trajectory, such as a human demonstration. For every state on the joint trajectory $\boldsymbol{\theta}$, we compute the Cartesian end-effector pose $\boldsymbol{p}$ and Jacobian $\boldsymbol{H}_{traj}$. The demonstration prior error $\boldsymbol{h}_{traj}$ is the pointwise difference between $\boldsymbol{p}$ and the corresponding point on the reference trajectory.

Taken together, dGPMP2-ND permits collision-free motion planning by iterative optimization, while respecting additional motion-level constraints such as joint limits or adherence to a reference trajectory. dGPMP2-ND is differentiable end-to-end, permitting the differentiation of the resulting trajectories with respect to input parameters such as target poses.

V. Joint Trajectory and Parameter Optimization

For many real-world robot tasks, motion trajectories and program parameters cannot be optimized in isolation. Grasping is a canonical example: Grasping an object with a given grasp pose imposes constraints on the approach motion, while e.g. collision objects in the environment make some approach motions, and therefore grasp poses, impossible. Grasp poses and approach motions must be jointly optimized in order to achieve task-level objectives (a stable grasp) while obeying motion-level constraints (collision-free approach). With dGPMP2-ND, gradient-based optimization over differentiable shadow programs permits the joint optimization of motion trajectories and program parameters. Fig. 4 shows the integration of dGPMP2-ND as a differentiable collision-free motion planner into the shadow program architecture. Shadow programs are differentiable, predictive models of robot programs; with dGPMP2-ND, collision-free planning becomes part of their forward pass.

A. Shadow Program Inversion with Differentiable Planning

With the integration of dGPMP2-ND in the shadow program architecture, even complex multi-skill robot programs involving collision-free planning skills are represented as differentiable computation graphs. This enables the computation of $\frac{\partial \Phi(\boldsymbol{\theta})}{\partial \boldsymbol{x}}$, the gradient of some task-level objective function Φ of the predicted trajectory w.r.t. the program parameters $\boldsymbol{x}$, and the optimization of $\boldsymbol{x}$ by a first-order optimizer. We call this procedure *Shadow Program Inversion with Differentiable Planning (SPI-DP)*. For each iteration i,

1) a **forward pass** through the shadow program is performed, yielding a prediction of $\boldsymbol{\theta}$ given initial inputs $\boldsymbol{x}$ and start state θ_0. This includes multiple iterations of dGPMP2-ND as an inner-loop trajectory optimizer for each shadow skill involving $\mathcal{C}$-space planning;
2) a **backward pass** is performed to compute $\frac{\partial \Phi(\boldsymbol{\theta})}{\partial \boldsymbol{x}}$ via automatic differentiation [45];
3) the **input parameters are incrementally updated** via gradient descent to minimize Φ. We use Adam [46] with

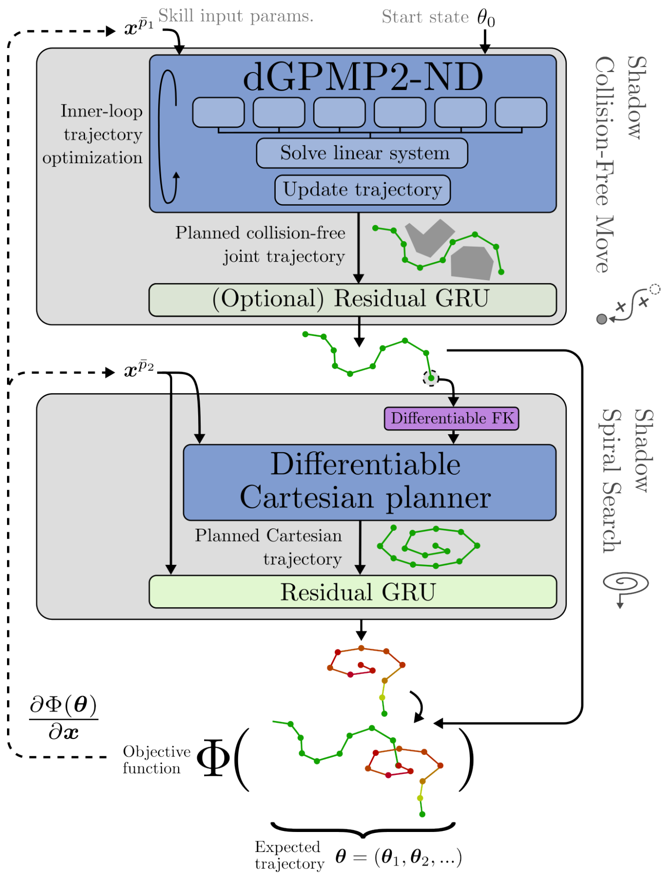

Fig. 4. A differentiable shadow program for a search-based insertion task composed of two skills. By combining differentiable planners (blue) and trained neural networks (light green), program parameters $\boldsymbol{x}$ can be optimized with respect to task-level objectives Φ while respecting motion-level constraints such as collision-freeness. A forward pass (top to bottom) predicts the expected real-world trajectory given program parameters $\boldsymbol{x}$ and robot state θ. The gradients of Φ are backpropagated and $\boldsymbol{x}$ is incrementally optimized.

a relatively high learning rate, such as 0.001, for fast convergence.

We find that the inner-loop dGPMP2-ND converges in less than 20 iterations for most planning problems we encountered. In each outer-loop iteration j, planning results are cached and used as the initial trajectories for dGPMP2-ND in the next iteration $j+1$, avoiding redundant planning. We find that one optimization iteration of a complex source program with 15 skills takes about 19 seconds on an Nvidia RTX 4090 graphics processing unit (GPU).

B. Task-Level Objective Functions

As the learnable components of shadow skills are trained offline to accurately predict the expected trajectory, the task objective Φ does not need to be known at training time. Given trained shadow skills, program parameters can be optimized for near-arbitrary differentiable objective functions Φ over the expected trajectory $\boldsymbol{\theta}$. For industrial applications, the process metrics cycle time, path length and success probability are most salient:

$$\Phi_{cyc}(\boldsymbol{\theta}) = \sum_{i=1}^{|\boldsymbol{\theta}|} \log(\boldsymbol{\theta}_{i,EOS})$$
$$\Phi_{path}(\boldsymbol{\theta}) = \sum_{i=2}^{|\boldsymbol{\theta}|} \|\boldsymbol{\theta}_{i,pos} - \boldsymbol{\theta}_{i-1,pos}\|$$
$$\Phi_{succ}(\boldsymbol{\theta}) = \sum_{i=1}^{|\boldsymbol{\theta}|} \log(\boldsymbol{\theta}_{i,succ})$$

Both the cycle time Φ_{cyc} and the success probability Φ_{succ} are defined as the binary cross-entropy of the end-of-sequence and task success flags with a target label of 1. For Φ_{succ}, this pushes the success probability of every trajectory point to 1 resulting in higher success probability of the execution. Φ_{cyc} pushes the end-of-sequence flag of every trajectory point toward 1, such that the trajectory has fewer points, which is equivalent to a reduced cycle time. Φ_{path} calculates the overall path length of the trajectory, independent of the end-effector velocity.

With dGPMP2-ND, SPI-DP optimizes program parameters subject to motion-level constraints such as collision-freeness, joint limits or proximity to a human demonstration (see VI-A and VI-B for details).

VI. Experiments

A. Household Pick-and-Place with Human Demonstration

We evaluate our framework on a household table cleaning scenario, in which a robot is tasked to pick up a cup from a table and place it into a cupboard, while guaranteeing collision-freeness (see VI-A.1). The motions are conditioned on one single human demonstration. In a second set of experiments, we demonstrate the zero-shot transfer to a different object (a wine glass) and the simultaneous optimization of the target pose (see VI-A.2). The experiments test three hypotheses:

- H1 **Motion-level optimization:** dGPMP2-ND is capable of planning collision-free, smooth pick-and-place motions that adhere to a single human demonstration for variable target poses and different object geometries;
- H2 **Task-level optimization:** SPI-DP can optimize the entire robot program parameters for KPIs resulting in reduced overall cycle-time while respecting imposed contact-force limits;
- H3 **Joint optimization:** SPI-DP is capable of jointly optimizing robot programs with respect to motion-level (collision-freeness, human demonstration) and task-level (cycle time, contact force) constraints.

The setup consists of a UR5 robotic arm with a flange-mounted ATI Gamma force-torque sensor and a SCHUNK pneumatic gripper. 10 demonstrations of a human transferring the cup from random pick-up poses to random target poses are collected with an Intel RealSense RGB-D camera. A human demonstration consists of the sampled 6D pose trajectory of the center of the cup.

1) Cup Pick-and-Place: A robot program consisting of approach, grasp, transfer, place and depart skills is optimized to place the cup at one of four target poses on two different shelf levels inside the cupboard. In this experiment, the

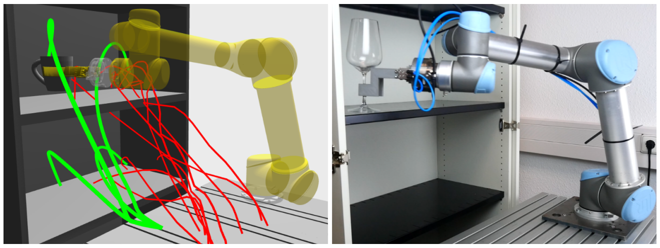

Fig. 5. Left: Experiment VI-A.1: 3D rendering of the collision world, 4 exemplary optimization results (green) and 10 human demonstrations (red). Right: Experiment VI-A.2: Real-world execution of an optimized program.

approach motion is optimized to respect the motion-level constraints illustrated in Fig. 3, with a collision environment consisting of the robot, table, cup and cupboard, and a Cartesian trajectory prior given by a human demonstration. A total of 40 trials are performed, one for each combination of target pose and human demonstration. The results are shown in Fig. 5 (left). All optimized motions were collision-free, even if the human demonstration contained a collision. The target pose was reached with a mean accuracy of 0.6 mm.

2) Wine Glass Pick-and-Place with Target Pose Optimization: The same robot program is optimized again, but the manipulated object is swapped for a wine glass and the gripper geometry is changed accordingly. In addition to the transfer motion, we also optimize the target pose (a parameter of the transfer skill) to minimize the cycle time of the overall program. The real-world experiment setup is shown in Fig. 5 (right). Again, 40 trials are performed, one for each combination of initial target pose and human demonstration. All optimized motions were collision-free and closely adhered to the human demonstration. SPI-DP optimized the target pose parameter resulting in motions which were 2.8 cm shorter on average. It moved the target pose as close to the shelf as possible, reducing cycle time by 40 % while avoiding collisions.

B. Engine Block Poka-Yoke Testing with Force Control

This experiment tests the scalability of joint motion- and task-level optimization for complex industrial robot programs. The task consists of a poka-yoke quality assurance task, in which a UR5 robot arm approaches three holes on an engine block and executes a force-controlled search motion to probe the hole (see Fig. 6. To simulate stochastic process noise, the engine block is moved on a linear axis by a random offset at every iteration. SPI-DP ensures collision-free motions from one hole to another and at the same time optimizes program parameters such as target poses, search patterns, velocities and contact forces with respect to the task-level objectives of cycle time minimization and maximization of the probability of task success.

20 trials are performed, for each of which the linear axis was moved randomly by up to 4 mm and the robot program was executed once with randomly initialized parameters, and once with optimized parameters. The results are shown in Table I. After optimization, the probability of finding each of the three holes is increased by 83%, 67% and 186%, respectively. In addition, optimization improved the search pattern, search dynamics and contact forces, reducing search duration by 62%. At the same time, motion-level optimization by dGPMP2-ND resulted in collision-free motions for all evaluations. The noticeably better optimization results for *Hole 3* stem from its wider diameter and the greater surface area surrounding it, compared to the other two holes.

TABLE I
EXPERIMENT VI-B: RESULTS

	Hole found		Duration (s)	
	unoptimized	optimized	unoptimized	optimized
Hole 1	6 / 20	11 / 20	2.29	1.39
Hole 2	6 / 20	10 / 20	1.95	0.77
Hole 3	7 / 20	20 / 20	1.97	0.48

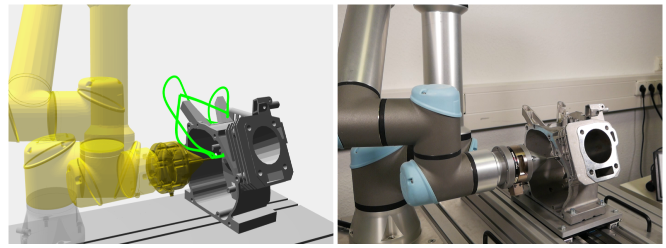

Fig. 6. Experiment VI-B: 3D rendering of the collision world and collision free trajectory planned with SPI-DP (left) and real-world execution (right).

VII. Conclusion and Outlook

We introduce Shadow Program Inversion with Differentiable Planning (SPI-DP), a first-order robot program optimizer capable of jointly optimizing robot program parameters and motion trajectories. To that end, we present dGPMP2-ND, a differentiable motion planner for n-DoF robotic manipulators. SPI-DP integrates dGPMP2-ND into a first-order general-purpose optimizer, which optimizes the parameters of a given robot program with respect to task-level objectives, while simultaneously ensuring that motion-level constraints are respected. SPI-DP is evaluated on two representative use cases from service and industrial robotics. Both experiments show that SPI-DP enables the optimization of program parameters such as target poses or search regions while ensuring collision-freeness, smoothness and kinematic feasibility. To our knowledge, SPI-DP is the first gradient-based optimizer capable of jointly optimizing program parameters and motion trajectories for arbitrary parameterized robot programs. Limitations of SPI-DP include its relative sensitivity with respect to hyperparameters, particularly the GP and collision factor covariances. We suggest the investigation of metaheuristics [47], [48] and meta-optimization approaches [49]–[51] for future work, which reliably steer the optimizer toward stable solutions or optimize planner hyperparameters for efficient convergence. Moreover, we seek to evaluate dGPMP2-ND on large-scale standardized planning benchmarks and investigate its integration as a differentiable planner for reinforcement learning [27]–[30] and Task and Motion Planning [22], [52], [53].

References

[1] H. H. Hoos, "Programming by optimization," *Communications of the ACM*, vol. 55, no. 2, pp. 70–80, Feb. 2012.

[2] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *ICRA*, May 2009, pp. 489–494.

[3] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic Trajectory Optimization for Motion Planning," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 4569–4574.

[4] T. Osa, "Multimodal trajectory optimization for motion planning," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 983–1001, July 2020.

[5] A. Dastider, H. Fang, and M. Lin, "RETRO: Reactive Trajectory Optimization for Real-Time Robot Motion Planning in Dynamic Environments," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 8764–8770.

[6] M. Racca, V. Kyrki, and M. Cakmak, "Interactive Tuning of Robot Program Parameters via Expected Divergence Maximization," in *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '20. New York, NY, USA: Association for Computing Machinery, Mar. 2020, pp. 629–638.

[7] B. Alt, D. Katic, R. Jäkel, A. K. Bozcuoglu, and M. Beetz, "Robot Program Parameter Inference via Differentiable Shadow Program Inversion," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 4672–4678.

[8] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian optimization with safety constraints: Safe and automatic parameter tuning in robotics," *Machine Learning*, vol. 112, no. 10, pp. 3713–3747, Oct. 2023.

[9] N. Kumar, T. Silver, W. McClinton, L. Zhao, S. Proulx, T. Lozano-Pérez, L. P. Kaelbling, and J. Barry, "Practice Makes Perfect: Planning to Learn Skill Parameter Policies," May 2024.

[10] J. A. Marvel, W. S. Newman, D. P. Gravel, G. Zhang, Jianjun Wang, and T. Fuhlbrigge, "Automated learning for parameter optimization of robotic assembly tasks utilizing genetic algorithms," in *2008 IEEE International Conference on Robotics and Biomimetics*, Feb. 2009, pp. 179–184.

[11] J. Kulk and J. S. Welsh, "Evaluation of walk optimisation techniques for the NAO robot," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*, Oct. 2011, pp. 306–311.

[12] F. Wu, W. Weimer, M. Harman, Y. Jia, and J. Krinke, "Deep Parameter Optimisation," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '15. New York, NY, USA: Association for Computing Machinery, July 2015, pp. 1375–1382.

[13] B. R. Bruce, J. M. Aitken, and J. Petke, "Deep Parameter Optimisation for Face Detection Using the Viola-Jones Algorithm in OpenCV," in *Search Based Software Engineering*, ser. Lecture Notes in Computer Science, F. Sarro and K. Deb, Eds. Cham: Springer International Publishing, 2016, pp. 238–243.

[14] J. Sohn, S. Lee, and S. Yoo, "Amortised Deep Parameter Optimisation of GPGPU Work Group Size for OpenCV," in *Search Based Software Engineering*, ser. Lecture Notes in Computer Science, F. Sarro and K. Deb, Eds. Cham: Springer International Publishing, 2016, pp. 211–217.

[15] R. Krohling, "Gaussian swarm: A novel particle swarm optimization algorithm," in *IEEE Conference on Cybernetics and Intelligent Systems, 2004.*, vol. 1, Dec. 2004, pp. 372–376 vol.1.

[16] G. Bolet, G. Georgakoudis, K. Parasyris, K. W. Cameron, D. Beckingsale, and T. Gamblin, "An Exploration of Global Optimization Strategies for Autotuning OpenMP-based Codes," in *2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2024, pp. 741–750.

[17] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian optimization for learning gaits under uncertainty," *Annals of Mathematics and Artificial Intelligence*, vol. 76, no. 1, pp. 5–23, Feb. 2016.

[18] M. Mayr, F. Ahmad, K. Chatzilygeroudis, L. Nardi, and V. Krueger, "Skill-based Multi-objective Reinforcement Learning of Industrial Robot Tasks with Planning and Knowledge Integration," *arXiv:2203.10033 [cs]*, Mar. 2022.

[19] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic Differentiation in Machine Learning: A Survey," *Journal of Machine Learning Research*, vol. 18, no. 153, pp. 1–43, 2018.

[20] C. C. Margossian, "A Review of automatic differentiation and its efficient implementation," *WIREs Data Mining and Knowledge Discovery*, vol. 9, no. 4, July 2019.

[21] M. Blondel and V. Roulet, "The Elements of Differentiable Programming," Mar. 2024.

[22] M. Toussaint, K. Allen, K. Smith, and J. Tenenbaum, "Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning," in *Robotics: Science and Systems XIV*, vol. 14, June 2018.

[23] J. Degrave, M. Hermans, J. Dambre, and F. Wyffels, "A Differentiable Physics Engine for Deep Learning in Robotics," *Frontiers in Neurorobotics*, vol. 13, 2019.

[24] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand, "DiffTaichi: Differentiable Programming for Physical Simulation," in *International Conference on Learning Representations*, Sept. 2019.

[25] Y.-L. Qiao, J. Liang, V. Koltun, and M. Lin, "Scalable Differentiable Physics for Learning and Control," in *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Nov. 2020, pp. 7847–7856.

[26] K. M. Jatavallabhula, M. Macklin, D. Fox, A. Garg, and F. Ramos, "Bayesian Object Models for Robotic Interaction with Differentiable Probabilistic Programming," in *Proceedings of The 6th Conference on Robot Learning*. PMLR, Mar. 2023, pp. 1563–1574.

[27] W. Wan, Y. Wang, Z. Erickson, and D. Held, "Differentiable Trajectory Optimization as a Policy Class for Reinforcement and Imitation Learning," Oct. 2023.

[28] M. A. Z. Mora, M. Peychev, S. Ha, M. Vechev, and S. Coros, "PODS: Policy Optimization via Differentiable Simulation," in *Proceedings of the 38th International Conference on Machine Learning*. PMLR, July 2021, pp. 7805–7817.

[29] P. Kolaric, D. K. Jha, A. U. Raghunathan, F. L. Lewis, M. Benosman, D. Romeres, and D. Nikovski, "Local Policy Optimization for Trajectory-Centric Reinforcement Learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 5094–5100.

[30] L. Wang, Y. Zhang, D. Zhu, S. Coleman, and D. Kerr, "Supervised Meta-Reinforcement Learning With Trajectory Optimization for Manipulation Tasks," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 16, no. 2, pp. 681–691, Apr. 2024.

[31] T. A. Howell, S. Le Cleac'h, S. Singh, P. Florence, Z. Manchester, and V. Sindhwani, "Trajectory Optimization with Optimization-Based Dynamics," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6750–6757, July 2022.

[32] T. A. Howell, K. Tracy, S. Le Cleac'h, and Z. Manchester, "CALIPSO: A Differentiable Solver for Trajectory Optimization with Conic and Complementarity Constraints," in *Robotics Research*, A. Billard, T. Asfour, and O. Khatib, Eds. Cham: Springer Nature Switzerland, 2023, pp. 504–521.

[33] M. Xu, T. L. Molloy, and S. Gould, "Revisiting Implicit Differentiation for Learning Problems in Optimal Control," in *Thirty-Seventh Conference on Neural Information Processing Systems*, Nov. 2023.

[34] W. Jin, S. Mou, and G. J. Pappas, "Safe pontryagin differentiable programming," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, ser. NIPS '21. Red Hook, NY, USA: Curran Associates Inc., June 2024, pp. 16 034–16 050.

[35] M. Mukadam, X. Yan, and B. Boots, "Gaussian Process Motion planning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 9–15.

[36] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-Time Gaussian Process Motion Planning via Probabilistic Inference," *The International Journal of Robotics Research*, vol. 37, no. 11, pp. 1319–1340, Sept. 2018.

[37] M. Bhardwaj, B. Boots, and M. Mukadam, "Differentiable Gaussian Process Motion Planning," in *arXiv:1907.09591 [Cs]*. Paris: IEEE, Mar. 2020.

[38] L. C. Cosier, R. Iordan, S. N. T. Zwane, G. Franzese, J. T. Wilson, M. Deisenroth, A. Terenin, and Y. Bekiroglu, "A Unifying Variational Framework for Gaussian Process Motion Planning," in *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2024, pp. 1315–1323.

[39] B. Alt, D. Katic, R. Jäkel, and M. Beetz, "Heuristic-Free Optimization of Force-Controlled Robot Search Strategies in Stochastic Environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022, pp. 8887–8893.

[40] C. Kienle, B. Alt, O. Celik, P. Becker, D. Katic, R. Jäkel, and G. Neumann, "MuTT: A Multimodal Trajectory Transformer for Robot Skills," 2024.

[41] S. R. Schmidt-Rohr, R. Jäkel, and G. Dirschl, "ArtiMinds Robot Programming Suite," ArtiMinds Robotics GmbH, 2013.

[42] "Simulator for industrial robots and offline programming - RoboDK," https://robodk.com/, May 2020.

[43] W. T. White, "Introducing Intrinsic Flowstate," May 2023.

[44] G. Sutanto, A. Wang, Y. Lin, M. Mukadam, G. Sukhatme, A. Rai, and F. Meier, "Encoding Physical Constraints in Differentiable Newton-Euler Algorithm," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*. PMLR, July 2020, pp. 804–813.

[45] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," Oct. 2017.

[46] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference for Learning Representations*. San Diego: arXiv, 2015.

[47] M. Gendreau and J.-Y. Potvin, Eds., *Handbook of Metaheuristics*, ser. International Series in Operations Research & Management Science. Boston, MA: Springer US, 2010, vol. 146.

[48] J. Swan, S. Adriaensen, A. E. I. Brownlee, K. Hammond, C. G. Johnson, A. Kheiri, F. Krawiec, J. J. Merelo, L. L. Minku, E. Özcan, G. L. Pappa, P. García-Sánchez, K. Sörensen, S. Voß, M. Wagner, and D. R. White, "Metaheuristics "In the Large"," *European Journal of Operational Research*, vol. 297, no. 2, pp. 393–406, Mar. 2022.

[49] X. Chen and E. Hazan, "Online Control for Meta-optimization," *Advances in Neural Information Processing Systems*, vol. 36, pp. 36 768–36 780, Dec. 2023.

[50] T. Gautam, S. Pfrommer, and S. Sojoudi, "Meta-Learning Parameterized First-Order Optimizers Using Differentiable Convex Optimization," in *2023 62nd IEEE Conference on Decision and Control (CDC)*, Dec. 2023, pp. 2284–2291.

[51] M. Feurer, J. Springenberg, and F. Hutter, "Initializing Bayesian Hyperparameter Optimization via Meta-Learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, Feb. 2015.

[52] J. Envall, R. Poranne, and S. Coros, "Differentiable Task Assignment and Motion Planning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2023, pp. 2049–2056.

[53] Y. Lee, P. Huang, K. M. Jatavallabhula, A. Z. Li, F. Damken, E. Heiden, K. Smith, D. Nowrouzezahrai, F. Ramos, and F. Shkurti, "STAMP: Differentiable Task and Motion Planning via Stein Variational Gradient Descent," Jan. 2024.