Efficient and Real-Time Motion Planning for Robotics Using Projection-Based Optimization

Xuemin Chi[1*], Hakan Girgin[2*], Tobias Löw[3], Yangyang Xie[1], Teng Xue[3], Jihao Huang[1], Cheng Hu[1], Zhitao Liu[1†], and Sylvain Calinon[3†]

***Abstract*— Generating motions for robots interacting with objects of various shapes is a complex challenge, further complicated by the robot's geometry and multiple desired behaviors. While current robot programming tools (such as inverse kinematics, collision avoidance, and manipulation planning) often treat these problems as constrained optimization, many existing solvers focus on specific problem domains or do not exploit geometric constraints effectively. We propose an efficient first-order method, Augmented Lagrangian Spectral Projected Gradient Descent (ALSPG), which leverages geometric projections via Euclidean projections, Minkowski sums, and basis functions. We show that by using geometric constraints rather than full constraints and gradients, ALSPG significantly improves real-time performance. Compared to second-order methods like iLQR, ALSPG remains competitive in the unconstrained case. We validate our method through toy examples and extensive simulations, and demonstrate its effectiveness on a 7-axis Franka robot, a 6-axis P-Rob robot and a 1:10 scale car in real-world experiments. Source codes, experimental data and videos are available on the project webpage: `https://sites.google.com/view/alspg-oc`**

Fig. 1: Chess robot setup. SPG-based IK solver validated on a publicly available 5-DOF chess-playing robot, successfully determines correct pick poses autonomously with 100% accuracy.

I. Introduction

Many robotics tasks are framed as constrained optimization problems. For example, inverse kinematics (IK) seeks a robot configuration that matches a desired pose while respecting constraints like joint limits or stability. Motion planning and optimal control aim to determine trajectories or control commands that satisfy task-specific dynamics and environmental constraints. Model predictive control (MPC) solves real-time optimal control problems by addressing simplified, short-horizon constrained optimization problems.

Several second-order solvers such as SNOPT [1], SLSQP [2], LANCELOT [3], and IPOPT [4]—are commonly used to solve general constrained optimization problems. In robotics, however, most research focuses on solvers tailored to specific problems. For instance, constrained versions of differential dynamic programming (DDP) [5], iterative linear quadratic regulator (iLQR) [6], TrajOpt [7], and CHOMP [8] are used for motion planning. However, many of these solvers are not open-source, making them difficult to benchmark and improve. Moreover, adapting them for real-time feedback applications, such as closed-loop IK and MPC, often requires significant tuning.

We address these challenges by proposing a simple yet powerful solver that can be easily implemented without requiring large memory resources. This solver exploits geometric constraints inherent to many robotic tasks, which can be described using geometric set primitives (see Table I). Examples include joint limits, center-of-mass stability, and avoiding or reaching geometric shapes (e.g., spheres or convex polytopes). These constraints can often be framed as projections rather than full constraint formulations. We argue that leveraging these projections, rather than treating constraints generically, can significantly improve solver performance.

One of the simplest algorithms for handling such projections is projected gradient descent, where the gradient is projected to ensure the next iterate remains inside the constraint set. A more advanced version, spectral projected gradient descent (SPG), has demonstrated strong practical performance and is considered a competitive alternative to second-order solvers in various fields [9]. Extensions of SPG to handle additional constraints using augmented Lagrangian methods have been explored in [10]–[12]. However, the application of these projection-based methods to popular second-order solvers in robotics has not been widely explored [13], resulting in a missed opportunity to fully exploit the potential of projections in this domain.

This work was supported in part by National Natural Science Foundation of China (NSFC:62173297), Zhejiang Key R&D Program (Grant NO. 2022C01035).(† Corresponding authors.)

∗ Equal Contribution.

1 Authors are associated with the Department of Control Science and Engineering, Zhejiang University.

2 Author is with Swiss Cobotics Competence Center, Biel, Switzerland.

3 Authors are with the Idiap Research Institute, Martigny, Switzerland and with EPFL, Lausanne, Switzerland.

This paper makes the following contributions:

- We propose an efficient projection-based optimization method that leverages geometric constraints in robotics tasks and extend Augmented Lagrangian Spectral Projected Gradient Descent (ALSPG) with a direct shooting method to handle multiple nonlinear constraints.
- We introduce and integrate various geometric projections including Euclidean projections, polytopic projections, and learning-based projections, into the ALSPG.
- We validate our approach through numerical simulations on planar arm systems, Franka robot arms, humanoid robots, and autonomous vehicles, providing performance benchmarks and analysis against baseline methods. We further assess the effectiveness of our method through real-world experiments on 6-axis and 7-axis robotic arms and a 1:10 scale car.

II. Related work

In optimization, Euclidean projections and the analytical expressions for various projections are fundamental for efficiently solving constrained optimization problems. The general theory for projecting onto a level set of an arbitrary function using KKT conditions is discussed in [14]. Extensive studies on projection methods and their properties can be found in [15], which provides a comprehensive theoretical background. In [16], Usmanova *et al.* propose an efficient algorithm for projecting onto arbitrary convex constraint sets, demonstrating that exploiting projections in optimization can significantly improve performance. Further, Bauschke and Koch [17] discuss and benchmark algorithms for projecting onto the intersection of convex sets, with Dykstra's alternating projection algorithm [18] being a key method for this task.

The simplest algorithm exploiting projections is projected gradient descent, which performs well in many settings. SPG improves upon this by exploiting curvature information via its spectral stepsizes, leading to faster convergence in many problems. A detailed review of SPG is provided in [11], highlighting its success in constrained optimization tasks. In [19], Torrisi *et al.* propose to use a projected gradient descent algorithm to solve the subproblems of sequential quadratic programming (SQP). They show that their method can solve MPC of an inverted pendulum faster than SNOPT. Our work is closest to theirs with the differences that we use SPG instead of a vanilla projected gradient descent to solve the subproblems of augmented Lagrangian instead of SQP. Additionally, we propose a direct way of handling multiple projections and inequality constraints, which is not trivial in [19].

In [20], Giftthaler and Buchli propose a projection of the updated direction of the control input onto the nullspace of the linearized constraints in iLQR. This approach can only handle simple equality constraints (for example, velocity-level constraints of second-order systems) and cannot treat position-level constraints for such systems, which is a very common and practical class of constraints in real world applications.

III. Problem Formulation

In this section, we present the definitions of projections used in robotic problems to formulate constraints such as constrained inverse kinematics, obstacle avoidance and other manipulation planning problems.

A. Euclidean Projections

The solution $\boldsymbol{x}^*$ to the following constrained optimization problem

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}-\boldsymbol{x}_0\|_2^2 \quad \text{s.t.} \quad \boldsymbol{x}\in\mathcal{C} \tag{1}$$

is called an Euclidean projection of the point $\boldsymbol{x}_0$ onto the set $\mathcal{C}$ and is denoted as $\boldsymbol{x}^*{=}\Pi_{\mathcal{C}}(\boldsymbol{x}_0)$. This operation determines the point $\boldsymbol{x}\in\mathcal{C}$ that is closest to $\boldsymbol{x}_0$ in Euclidean sense. For many sets $\mathcal{C}$, $\Pi_{\mathcal{C}}(\cdot)$ admits analytical expressions that are given in Table I. Even though, usually, these sets are convex (e.g. bounded domains), some nonconvex sets also admit analytical solution(s) that are easy to compute (e.g. being outside of a sphere). In cases $\mathcal{C}$ is non-convex, multiple feasible solutions may exist. In such situations, either a strategy for selecting the solution or a convex decomposition method may be required. Note that many of these sets are frequently used in robotics, from joint/torque limits and avoiding spherical/square obstacles to satisfying virtual fixtures defined in the task space of the robot.

B. Polytopic Projections

The geometries of mobile robots often vary, and they can be represented as polytopes defined by hyperplanes. When considering the robot's geometry, Euclidean projections may not always be applicable. It becomes necessary to account for projections between the robot (modeled as a polytope) and obstacles. Let the robot's shape be convex, with its occupied space denoted as the set $\mathcal{C}_r$. The geometric center of the polytopic robot, denoted $\boldsymbol{p}\in\mathbb{R}^n$, serves as the point of interest for control and collision avoidance. We consider the obstacles or goal regions as convex polytopes in $\mathbb{R}^n$ (with $n=2$ or 3). If the shapes are not convex, the convex-hulls or convex-decomposition can be employed to approximate them as a collection of convex polytopes. The Minkowski sum $\mathcal{M}$ between robot and polytopic set $\mathcal{C}$ is defined as:

$$\mathcal{M}=\mathcal{C}_r\oplus\mathcal{C}=\{\boldsymbol{x}=\boldsymbol{x}_1+\boldsymbol{x}_2 \mid \boldsymbol{x}_1\in\mathcal{C}_r,\boldsymbol{x}_2\in\mathcal{C}\} \tag{2}$$

where $\mathcal{M}$ is the configuration space obstacle for translation movements of robots. If the geometric center $p\in\mathcal{M}$, the robot and the polytope object intersect. The problem of projections between two polytopic sets then reduces to projections between the geometric center $\boldsymbol{p}$ and the Minkowski sum $\mathcal{M}$, since $\mathcal{M}$ is composed of hyperplanes, projecting a point out of a polytope, $\boldsymbol{x}^*{=}\Pi_{\mathcal{M}}(\boldsymbol{x}_0)$, can be resolved using the method outlined I, as shown in Fig. 2. The robot's rotational movements can be accounted for by augmenting the Minkowski set with an additional dimension.

TABLE I: Projections onto bounded domains, affine hyperplane, quadric and second-order cone.

	Bounds	Affine hyperplane	Quadratic	Second-order cone
$\mathcal{C}$	$l \leq x \leq u$	$l \leq \boldsymbol{a}^\top \boldsymbol{x} \leq u$	$l \leq \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{x} \leq u$	$\|\boldsymbol{x}\| \leq t$
$\Pi_{\mathcal{C}}$	$\begin{cases} x & \text{if } l \leq x \leq u \\ u & \text{if } x > u \\ l & \text{if } x < l \end{cases}$	$\begin{cases} \boldsymbol{x} & \text{if } l \leq \boldsymbol{a}^\top \boldsymbol{x} \leq u \\ \boldsymbol{x} - \frac{\boldsymbol{a}(\boldsymbol{a}^\top \boldsymbol{x} - u)}{\|\boldsymbol{a}\|_2^2} & \text{if } \boldsymbol{a}^\top \boldsymbol{x} > u \\ \boldsymbol{x} - \frac{\boldsymbol{a}(\boldsymbol{a}^\top \boldsymbol{x} - l)}{\|\boldsymbol{a}\|_2^2} & \text{if } \boldsymbol{a}^\top \boldsymbol{x} < l \end{cases}$	$\begin{cases} \boldsymbol{x} & \text{if } l \leq \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{x} \leq u \\ \frac{\boldsymbol{x}\sqrt{2u}}{\|\boldsymbol{x}\|} & \text{if } \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{x} > u \\ \frac{\boldsymbol{x}\sqrt{2l}}{\|\boldsymbol{x}\|} & \text{if } l > \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{x} \end{cases}$	$\begin{cases} (\boldsymbol{x}, t), & \text{if } \|\boldsymbol{x}\| \leq t \\ (\boldsymbol{0}, 0), & \text{if } \|\boldsymbol{x}\| \leq -t \\ \frac{\|\boldsymbol{x}\| + t}{2}(\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}, 1) & \text{otherwise} \end{cases}$

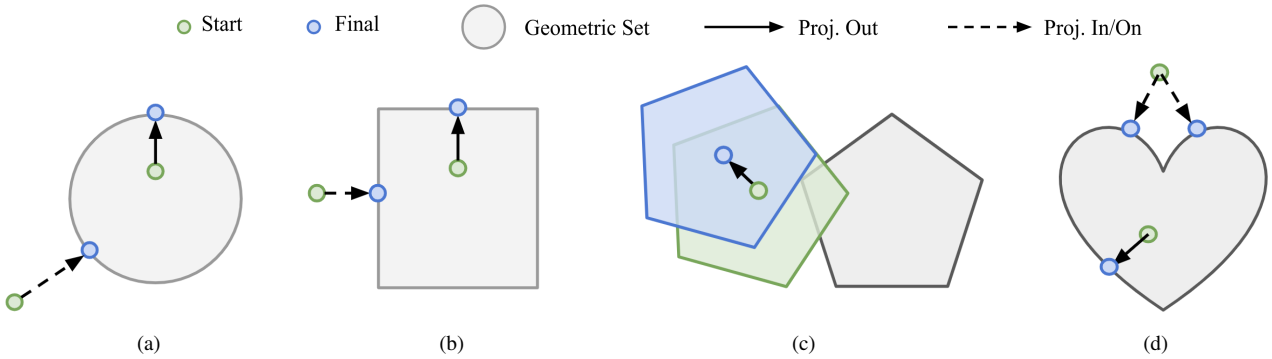


Fig. 2: Geometric Projections. Projecting outside of a set can be utilized for collision avoidance while projecting inside or onto a set can be employed for goal reaching. a) Analytical circle projection. b) Analytical box projection. c) Polytopic projection. d) Implicit projection.

C. Implicit Projections

When the shape of the object is implicit and cannot be expressed using hyperplanes, learning-based techniques can be employed to design the projections. Bernstein polynomial basis functions are efficient for learning the implicit shape. The advantage of this approach lies in the availability of analytical and smooth gradient information. Assuming the order of the polynomials is r and there are c control points, the matrix form of the implicit shape is $\mathcal{S} := \boldsymbol{t}^\top \boldsymbol{M} \Phi$, where $\boldsymbol{t} \in \mathbb{R}^r$ is the time vector, and $\boldsymbol{M} \in \mathbb{R}^{r \times c}$ is the characteristic matrix and $\Phi \in \mathbb{R}^c$ represents the control points. The analytical gradient $\nabla_t \mathcal{S}$ can be computed efficiently. The implicit projection is demonstrated in Fig. 2.

IV. Augmented Lagrangian Spectral Projected Gradient Descent for Robotics

This section gives the SPG algorithm along with the non-monotone line search procedure. These algorithms are easy to implement without big memory requirements and yet result in powerful solvers. Next, we give the ALSPG algorithm based on geometric projections. Finally, the direct shooting approach is adopted to formulate the constrained optimization problems for robotics.

A. Spectral Projected Gradient Descent

SPG is an improved version of a vanilla projected gradient descent using spectral stepsizes. Its excellent numerical results even in comparison to second-order methods have been a point of attraction in the optimization literature [9]. SPG tackles constrained optimization problems in the form of

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{s.t.} \quad \boldsymbol{x} \in \mathcal{C}, \tag{3}$$

by constructing a local quadratic model of the objective function

$$\begin{aligned} f(\boldsymbol{x}) &\approx f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^\top (\boldsymbol{x} - \boldsymbol{x}_k) + \frac{1}{2\gamma_k} \|\boldsymbol{x} - \boldsymbol{x}_k\|_2^2, \\ &= \frac{1}{2\gamma_k} \|\boldsymbol{x} - (\boldsymbol{x}_k - \gamma_k \nabla f(\boldsymbol{x}_k))\|_2^2 + \text{const.}, \end{aligned}$$

and by minimizing it subject to the constraints as

$$\min_{\boldsymbol{x}} \frac{1}{2\gamma_k} \|\boldsymbol{x} - (\boldsymbol{x}_k - \gamma_k \nabla f(\boldsymbol{x}_k))\|_2^2 \quad \text{s.t.} \quad \boldsymbol{x} \in \mathcal{C}, \tag{4}$$

whose solution is an Euclidean projection as described in (1) and given by $\Pi_{\mathcal{C}}(\boldsymbol{x}_k - \gamma_k \nabla f(\boldsymbol{x}_k))$. The local search direction $\boldsymbol{d}_k$ for SPG is then given by

$$\boldsymbol{d}_k = \Pi_{\mathcal{C}}(\boldsymbol{x}_k - \gamma_k \nabla f(\boldsymbol{x}_k)) - \boldsymbol{x}_k, \tag{5}$$

which is used in a non-monotone line search (Algorithm 1) with $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, to find α_k satisfying $f(\boldsymbol{x}_{k+1}) \leq f_{\max} + \alpha_k \gamma_k \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{d}_k$, where $f_{\max} = \max\{f(\boldsymbol{x}_{k-j}) \,|\, 0 \leq j \leq \min\{k, M-1\}\}$. Non-monotone line search allows for increasing objective values for some iterations M preventing getting stuck at bad local minima. A typical value for M is 10. When $M = 1$, it reduces to a monotone line search.

The choice of γ_k affects the convergence properties significantly since it introduces curvature information to the solver. Note that when choosing $\gamma_k = 1$, SPG is equivalent to the widely known projected gradient descent. SPG uses spectral stepsizes obtained by a least-square approximation of the Hessian matrix by $\gamma_k \boldsymbol{I}$. These spectral stepsizes are computed by proposals

$$\gamma_k^{(1)} = \frac{\boldsymbol{s}_k^\top \boldsymbol{s}_k}{\boldsymbol{s}_k^\top \boldsymbol{y}_k} \quad \text{and} \quad \gamma_k^{(2)} = \frac{\boldsymbol{s}_k^\top \boldsymbol{y}_k}{\boldsymbol{y}_k^\top \boldsymbol{y}_k}, \tag{6}$$

where $\boldsymbol{s}_k = \boldsymbol{x}_k - \boldsymbol{x}_{k-1}$ and $\boldsymbol{y}_k = \nabla f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_{k-1})$ [9]. In the case of the quadratic objective function in the form

Algorithm 1: Non-monotone line search

Set $\beta = 10^{-4}$, $\alpha = 1$, $M = 10$, $c = \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{d}_k$,
$f_{\text{max}} = \max\{f(\boldsymbol{x}_{k-j}) | 0 \leq j \leq \min\{k, M-1\}\}$
while $f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k) > f_{max} + \alpha\beta c$ **do**
  $\bar{\alpha} = -0.5\alpha^2 c \Big(f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k) - f(\boldsymbol{x}_k) - \alpha c \Big)^{-1}$
  if $0.1 \leq \bar{\alpha} \leq 0.9$ **then**
    $\alpha = \bar{\alpha}$
  else
    $\alpha = \alpha / 2$
  end
end

Algorithm 2: Spectral Projected Gradient Descent

Initialize $\boldsymbol{x}_k$, γ_k, ϵ=10^{-5}, k=0;
while $\|\Pi_{\mathcal{C}}(\boldsymbol{x}_k - \nabla f(\boldsymbol{x}_k)) - \boldsymbol{x}_k\|_\infty > \epsilon$ **do**
  Find a search direction by $\boldsymbol{d}_k = \Pi_{\mathcal{C}}(\boldsymbol{x}_k - \gamma_k \nabla f(\boldsymbol{x}_k)) - \boldsymbol{x}_k$
  Do non-monotone line search using Algorithm 1 to find $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$
  $\boldsymbol{s}_{k+1} = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ and $\boldsymbol{y}_{k+1} = \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)$
  $\gamma^{(1)} = \frac{\boldsymbol{s}_{k+1}^\top \boldsymbol{s}_{k+1}}{\boldsymbol{s}_{k+1}^\top \boldsymbol{y}_{k+1}}$ and $\gamma^{(2)} = \frac{\boldsymbol{s}_{k+1}^\top \boldsymbol{y}_{k+1}}{\boldsymbol{y}_{k+1}^\top \boldsymbol{y}_{k+1}}$
  if $\gamma^{(1)} < 2\gamma^{(2)}$ **then**
    $\gamma_{k+1} = \gamma^{(2)}$
  else
    $\gamma_{k+1} = \gamma^{(1)} - \frac{1}{2}\gamma^{(2)}$
  end
  $k = k + 1$
end

of $\boldsymbol{x}^\top \boldsymbol{Q} \boldsymbol{x}$, these two values correspond to the maximum and minimum eigenvalues of the matrix $\boldsymbol{Q}$. The initial spectral stepsize can be computed by setting $\bar{\boldsymbol{x}}_0 = \boldsymbol{x}_0 - \gamma_{\text{small}} \nabla f(\boldsymbol{x}_0)$ where γ_{small} is 10^{-4}, and computing $\bar{\boldsymbol{s}}_0 = \bar{\boldsymbol{x}}_0 - \boldsymbol{x}_0$ and $\bar{\boldsymbol{y}}_0 = \nabla f(\bar{\boldsymbol{x}}_0) - \nabla f(\boldsymbol{x}_0)$. Note that this heuristic operation costs one more gradient computation. The final algorithm is given by Algorithm 2.

B. Augmented Lagrangian spectral projected gradient descent

SPG alone is usually not sufficient to solve problems in robotics with complicated nonlinear constraints. In [12], Jia *et al.* provides an augmented Lagrangian framework to solve problems with constraints $\boldsymbol{g}(\boldsymbol{x}) \in \mathcal{C}$ and $\boldsymbol{x} \in \mathcal{D}$, where $\boldsymbol{g}(\cdot)$ is a convex function, $\mathcal{C}$ is a convex set, and $\mathcal{D}$ is a closed nonempty set, both equipped with easy projections.

In this section, we build on the work in [12] with the extension of multiple projections and additional general equality and inequality constraints. The general optimization problem that we are tackling here is

$$\min_{\boldsymbol{x} \in \mathcal{D}} f(\boldsymbol{x}) \quad \text{s.t.} \quad \boldsymbol{g}_i(\boldsymbol{x}) \in \mathcal{C}_i, \quad \forall i \in \{1, \ldots, p\} \tag{7}$$

Algorithm 3: ALSPG

Set $\boldsymbol{\lambda}_0^{\mathcal{C}_i}$=$\boldsymbol{0}$, $\rho_0^{\mathcal{C}_i}$=0.1, $\epsilon_2 = 10^{-4}$
for $k = 0 \leftarrow \infty$ **do**
  $\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x} \in \mathcal{D}} \mathcal{L}(\boldsymbol{x}, \{\boldsymbol{\lambda}^{\mathcal{C}_i}, \rho^{\mathcal{C}_i}\}_{i=1}^p)$ with SPG in Algorithm 2
  foreach $\mathcal{C}_i$ **do**
    $\lambda_{k+1}^{\mathcal{C}_i} = \rho^{\mathcal{C}_i} \Big(\boldsymbol{g}_i(\boldsymbol{x}) + \frac{\boldsymbol{\lambda}^{\mathcal{C}_i}}{\rho^{\mathcal{C}_i}} - \Pi_{\mathcal{C}_i} \Big(\boldsymbol{g}_i(\boldsymbol{x}) + \frac{\boldsymbol{\lambda}^{\mathcal{C}_i}}{\rho^{\mathcal{C}_i}} \Big) \Big)$
    if $V(\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}_{k+1}^{\mathcal{C}_i}, \rho_k^{\mathcal{C}_i}) \leq V(\boldsymbol{x}_k, \boldsymbol{\lambda}_k^{\mathcal{C}_i}, \rho_k^{\mathcal{C}_i})$ **then**
      $\rho_{k+1}^{\mathcal{C}_i} = \rho_k^{\mathcal{C}_i}$
    else
      $\rho_{k+1}^{\mathcal{C}_i} = 10 \rho_k^{\mathcal{C}_i}$
    end
  end
  if $V \left(\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}_{k+1}^{\mathcal{C}_i}, \rho_k^{\mathcal{C}_i} \right) < \epsilon_2$ **then**
    break
  end
end

where $\boldsymbol{g}_i(\cdot)$ are assumed to be arbitrary nonlinear functions. Note that even though the convergence results in [12] apply to the case when these are convex functions and convex sets, we found in practice that the algorithm is powerful enough to extend to more general cases.

We use the following augmented Lagrangian function

$$\mathcal{L}(\boldsymbol{x}, \{\boldsymbol{\lambda}^{\mathcal{C}_i}, \rho^{\mathcal{C}_i}\}_{i=1}^p) = f(\boldsymbol{x}) + \sum_{i=1}^p \frac{\rho^{\mathcal{C}_i}}{2} \left\| \boldsymbol{g}_i(\boldsymbol{x}) + \frac{\boldsymbol{\lambda}^{\mathcal{C}_i}}{\rho^{\mathcal{C}_i}} - \Pi_{\mathcal{C}_i} \Big(\boldsymbol{g}_i(\boldsymbol{x}) + \frac{\boldsymbol{\lambda}^{\mathcal{C}_i}}{\rho^{\mathcal{C}_i}} \Big) \right\|_2^2$$

whose derivative w.r.t. $\boldsymbol{x}$ is given by

$$\nabla \mathcal{L}(\boldsymbol{x}, \{\boldsymbol{\lambda}^{\mathcal{C}_i}, \rho^{\mathcal{C}_i}\}_{i=1}^p) = \nabla f(\boldsymbol{x}) + \sum_{i=1}^p \frac{\rho^{\mathcal{C}_i}}{2} \nabla \boldsymbol{g}_i^\top(\boldsymbol{x}) \Big(\boldsymbol{g}_i(\boldsymbol{x}) + \frac{\boldsymbol{\lambda}^{\mathcal{C}_i}}{\rho^{\mathcal{C}_i}} - \Pi_{\mathcal{C}_i} \Big(\boldsymbol{g}_i(\boldsymbol{x}) + \frac{\boldsymbol{\lambda}^{\mathcal{C}_i}}{\rho^{\mathcal{C}_i}} \Big) \Big),$$

using the property of convex Euclidean projections derivative $\nabla \|\boldsymbol{g}(\boldsymbol{x}) - \Pi(\boldsymbol{g}(\boldsymbol{x}))\|_2^2 = \nabla \boldsymbol{g}(\boldsymbol{x})^\top (\boldsymbol{g}(\boldsymbol{x}) - \Pi(\boldsymbol{g}(\boldsymbol{x}))$, see [15] for details. $\boldsymbol{\lambda}$ is vector of the Lagrangian multipliers and ρ is the penalty parameter. This way, we obtain a formulation that does not need the gradient of the projection function $\Pi_{\mathcal{C}_i}(\cdot)$. One iteration of ALSPG optimizes the subproblem $\arg\min_{\boldsymbol{x} \in \mathcal{D}} \mathcal{L}(\boldsymbol{x}, \{\boldsymbol{\lambda}^{\mathcal{C}_i}, \rho^{\mathcal{C}_i}\}_{i=1}^p)$ given $\{\boldsymbol{\lambda}^{\mathcal{C}_i}, \rho^{\mathcal{C}_i}\}_{i=1}^p$, and then updates these according to the next iterate. Defining the auxiliary function $V(\boldsymbol{x}, \boldsymbol{\lambda}^{\mathcal{C}_i}, \rho^{\mathcal{C}_i}) = \left\| \boldsymbol{g}_i(\boldsymbol{x}) - \Pi_{\mathcal{C}_i} \Big(\boldsymbol{g}_i(\boldsymbol{x}) + \frac{\boldsymbol{\lambda}^{\mathcal{C}_i}}{\rho^{\mathcal{C}_i}} \Big) \right\|$, the algorithm is summarized in Algorithm 3. Note that one can define and tune many heuristics around augmented Lagrangian methods with possible extensions to primal-dual methods.

C. Optimal Control with ALSPG

We consider the following generic constrained optimization problem

$$\min_{\boldsymbol{x} \in \mathcal{C}_{\boldsymbol{x}}, \boldsymbol{u} \in \mathcal{C}_{\boldsymbol{u}}} c(\boldsymbol{x}, \boldsymbol{u}) \quad \text{s.t.} \quad \begin{array}{l} \boldsymbol{x} = \boldsymbol{F}(\boldsymbol{x}_0, \boldsymbol{u}), \\ \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{0}, \end{array} \tag{8}$$

where the state trajectory $\boldsymbol{x}=\left[\boldsymbol{x}_1^\top, \boldsymbol{x}_2^\top, \ldots, \boldsymbol{x}_t^\top, \ldots, \boldsymbol{x}_T^\top\right]^\top$, the control trajectory $\boldsymbol{u} = \left[\boldsymbol{u}_0^\top, \boldsymbol{u}_1^\top, \ldots, \boldsymbol{u}_t^\top, \ldots, \boldsymbol{u}_{T-1}^\top\right]^\top$ and the function $\boldsymbol{F}(\cdot,\cdot)$ correspond to the forward rollout of the states using a dynamics model $\boldsymbol{x}_{t+1}=\boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t)$. We use a direct shooting approach and transform Eq. (8) into a problem in $\boldsymbol{u}$ only by considering

$$\min_{\boldsymbol{u}\in\mathcal{C}_{\boldsymbol{u}}} c(\boldsymbol{F}(\boldsymbol{x}_0,\boldsymbol{u}),\boldsymbol{u}) \quad \text{s.t.} \quad \begin{array}{l}\boldsymbol{F}(\boldsymbol{x}_0,\boldsymbol{u})\in\mathcal{C}_{\boldsymbol{x}},\\ \boldsymbol{h}(\boldsymbol{F}(\boldsymbol{x}_0,\boldsymbol{u}),\boldsymbol{u})=\boldsymbol{0},\end{array} \tag{9}$$

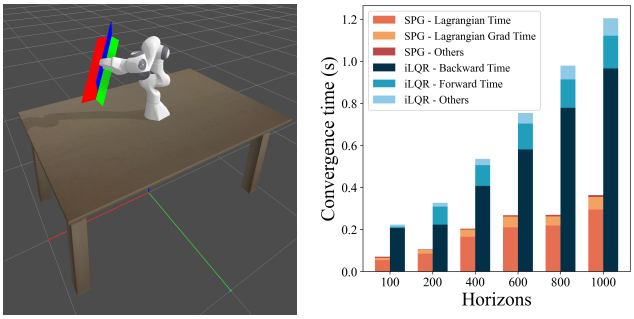
which is exactly in the form of Eq. (7), if $\boldsymbol{g}_1(\boldsymbol{u}) = \boldsymbol{F}(\boldsymbol{x}_0,\boldsymbol{u})$ and $\boldsymbol{g}_2(\boldsymbol{u}) = \boldsymbol{h}(\boldsymbol{F}(\boldsymbol{x}_0,\boldsymbol{u}),\boldsymbol{u})$. The unconstrained version of this problem can be solved with least-square approaches. However, assuming $\boldsymbol{x}_t \in \mathbb{R}^m$, $\boldsymbol{u}_t \in \mathbb{R}^n$, this requires the inversion of a matrix of size $Tn \times Tn$, whereas here we only work with the gradients of the objective function and the functions $\boldsymbol{g}_i(\cdot)$. The component that requires a special attention is $\nabla \boldsymbol{F}(\boldsymbol{x}_0,\boldsymbol{u})$ and in particular, its transpose product with a vector. It turns out that this product can be efficiently computed with a recursive formula (as also described in [19]), resulting in fast SPG iterations. Denoting $\boldsymbol{A}_t = \nabla_{\boldsymbol{x}_t}\boldsymbol{f}(\boldsymbol{x}_t,\boldsymbol{u}_t)$, $\boldsymbol{B}_t = \nabla_{\boldsymbol{u}_t}\boldsymbol{f}(\boldsymbol{x}_t,\boldsymbol{u}_t)$, and $\nabla_{\boldsymbol{u}}\boldsymbol{F}(\boldsymbol{x}_0,\boldsymbol{u})^\top\boldsymbol{y} = \boldsymbol{z}$ with $\boldsymbol{y} = \left[\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_t, \ldots, \boldsymbol{y}_{T-1}\right]$, $\boldsymbol{z} = \left[\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_t, \ldots, \boldsymbol{z}_{T-1}\right]$, one can show that the matrix vector product $\nabla_{\boldsymbol{u}}\boldsymbol{F}(\boldsymbol{x}_0,\boldsymbol{u})^\top\boldsymbol{y}$ can be written as

$$\begin{bmatrix} \boldsymbol{B}_0^\top & \boldsymbol{B}_0^\top\boldsymbol{A}_1^\top & \boldsymbol{B}_0^\top\boldsymbol{A}_1^\top\boldsymbol{A}_2^\top & \ldots & \boldsymbol{B}_0^\top\prod_{t=1}^{T-1}\boldsymbol{A}_t^\top \\ \boldsymbol{0} & \boldsymbol{B}_1^\top & \boldsymbol{B}_1^\top\boldsymbol{A}_2^\top & \ldots & \boldsymbol{B}_1^\top\prod_{t=2}^{T-1}\boldsymbol{A}_t^\top \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \ldots & \boldsymbol{B}_{T-1}^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{T-1} \end{bmatrix}$$
$$= \begin{bmatrix} \boldsymbol{B}_0^\top(\boldsymbol{y}_0 + \boldsymbol{A}_1^\top\boldsymbol{y}_1 + \boldsymbol{A}_1^\top\boldsymbol{A}_2^\top\boldsymbol{y}_2 + \prod_{t=1}^{T-1}\boldsymbol{A}_t^\top\boldsymbol{y}_{T-1}) \\ \boldsymbol{B}_1^\top(\boldsymbol{y}_1 + \boldsymbol{A}_2^\top\boldsymbol{y}_2 + \boldsymbol{A}_2^\top\boldsymbol{A}_3^\top\boldsymbol{y}_3 + \prod_{t=2}^{T-1}\boldsymbol{A}_t^\top\boldsymbol{y}_{T-1}) \\ \vdots \\ \boldsymbol{B}_{T-2}^\top(\boldsymbol{y}_{T-2} + \boldsymbol{A}_{T-1}^\top\boldsymbol{y}_{T-1}) \\ \boldsymbol{B}_{T-1}^\top\boldsymbol{y}_{T-1} \end{bmatrix},$$

where the terms in parantheses can be computed recursively backward by $\bar{\boldsymbol{z}}_{t+1} = (\boldsymbol{y}_{t+1} + \boldsymbol{A}_t^\top\bar{\boldsymbol{z}}_t)$, $\boldsymbol{z}_t = \boldsymbol{B}_{t-1}^\top\bar{\boldsymbol{z}}_t$ and $\bar{\boldsymbol{z}}_{T-1} = \boldsymbol{y}_{T-1}$, without having to construct the big matrix $\nabla_{\boldsymbol{u}}\boldsymbol{F}(\boldsymbol{x}_0,\boldsymbol{u})^\top$. Note that when there are no constraints on the state and $\boldsymbol{h}(\cdot) = 0$, Eq. (9) can be solved directly with the SPG algorithm.

V. Experiments

In this section, we perform extensive simulations and real-world experiments on multiple robotic tasks, including IK problems, motion planning, MPC for a contact-rich pushing task, autonomous navigation and parking tasks. Real-world evaluations were conducted on a 7-axis Franka robot, a 6-axis P-Rob robot, and a 1:10 scale car. The motivation behind these experiments is to show that: 1) the proposed way of solving these robotics problems can be faster than the second-order methods such as iLQR and IPOPT; and 2) exploiting projections whenever we can, instead of leaving the constraints for the solver to treat them as generic constraints, increases the performance significantly.



(a) (b)

Fig. 3: Comparison of iLQR and SPG in terms of convergence time evolution vs the number of horizons.

TABLE II: Comparison of constrained inverse kinematics with and without projections.

Application	Num. of fun. eval.	Num of Jac. eval.
Talos IK without Proj.	6459.4 ± 3756.8	3791.79 ± 1061.05
Talos IK with Proj.	97.64 ± 82.84	883.44 ± 81.7

A. Inverse kinematics

We first evaluate the performance of SPG compared to iLQR[1] in a reach planning task using a 7-axis manipulator without constraints. iLQR is implemented with dynamic programming. SPG is implemented as detailed in the previous section. Both implementations are in Python. The control input is $\ddot{q} \in \mathbb{R}^7$ and the states are joint velocities and positions $(\dot{q}, q) \in \mathbb{R}^{14}$. The results on Fig. 3b show that the convergence time scales linearly with the planning horizons due to the growing number of decision variables. Notably, SPG scales more efficiently than iLQR, demonstrating its potential for real-time applications.

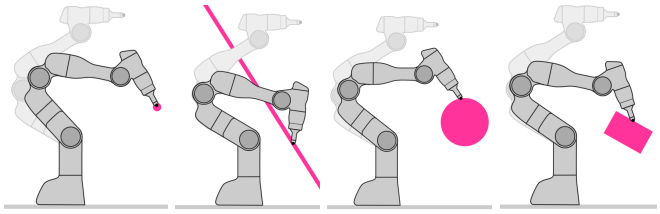
A Constrained inverse kinematics problem can be described in many ways using projections. One typical way is to find a feasible $\boldsymbol{q} \in \mathcal{C}_{\boldsymbol{q}}$ that minimizes a cost to be away from a given initial configuration $\boldsymbol{q}_0$ while respecting general constraints $\boldsymbol{h}(\boldsymbol{q}) = \boldsymbol{0}$ and projection constraints $\boldsymbol{f}(\boldsymbol{q}) \in \mathcal{C}_{\boldsymbol{x}}$

$$\min_{\boldsymbol{q}\in\mathcal{C}_{\boldsymbol{q}}} \|\boldsymbol{q}-\boldsymbol{q}_0\|_2^2 \quad \text{s.t.} \quad \begin{array}{l}\boldsymbol{h}(\boldsymbol{q})=\boldsymbol{0},\\ \boldsymbol{f}(\boldsymbol{q})\in\mathcal{C}_{\boldsymbol{x}},\end{array} \tag{10}$$

where $\boldsymbol{f}(\cdot)$ can represent entities such as the end-effector pose or the center of mass for which the constraints are easier to be expressed as projections onto $\mathcal{C}_{\boldsymbol{x}}$, and $\mathcal{C}_{\boldsymbol{q}}$ can represent the configuration space within the joint limits. Fig. 4 shows a 3-axis planar manipulator with $\boldsymbol{f}(\cdot)$ representing the end-effector position and $\mathcal{C}_{\boldsymbol{x}}$ denoting feasible regions.

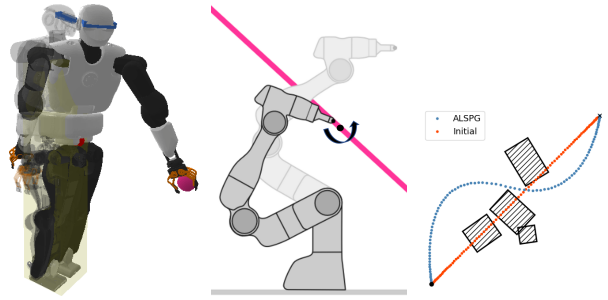
Talos IK: We tested our algorithm on a high-dimensional (32 DoF) IK problem for the TALOS robot (see Fig. 5a), subject to the following constraints: i) the center of mass must remain inside a box; ii) the end-effector must lie within a sphere; and iii) the foot position and orientation are fixed. We compared two versions of the ALSPG algorithm: 1) by casting these constraints as projections onto $\mathcal{C}_{\boldsymbol{x}}$; and 2) by embedding all constraints within the function $\boldsymbol{h}(\cdot)$ to assess

[1]The implementation and code of iLQR can refer to RCFS.



(a) (b) (c) (d)

Fig. 4: Projection view of inverse kinematics problem. (a) Reaching a point (standard IK problem): $\mathcal{C}_{\boldsymbol{x}} = \{\boldsymbol{x} \mid \boldsymbol{x}{=}\boldsymbol{x}_d\}$. (b) Reaching under/above/on a plane (in the halfspace): $\mathcal{C}_{\boldsymbol{x}} = \{\boldsymbol{x} \mid \boldsymbol{a}^\top \boldsymbol{x} + b{=}0\}$. (c) Reaching inside/outside/on a circle: $\mathcal{C}_{\boldsymbol{x}} = \{\boldsymbol{x} \mid \leq r_i^2 \leq \|\boldsymbol{x} - \boldsymbol{x}_d\|_2^2 \leq r_o^2\}$. (d) Reaching inside/outside/on a rectangle: $\mathcal{C}_{\boldsymbol{x}} = \{\boldsymbol{x} \mid \|\boldsymbol{x} - \boldsymbol{x}_d\|_{\infty,\boldsymbol{W}} \leq L\}$.



(a) (b) (c)

Fig. 5: Inverse kinematics and motion planning problems solved with the proposed algorithm. (a) Talos inverse kinematics problem with foot pose, center of mass stability (red point inside yellow rectangular prism) and end-effector inside a (pink) sphere constraints. (b) Robust inverse kinematics solution with $\mathcal{C}_{\boldsymbol{p}} = \{\boldsymbol{p} \mid \boldsymbol{\mu}^\top \boldsymbol{p} + \Psi^{-1}(\eta)\|\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{p}\|_2 \leq 0\}$. (c) Motion planning problem in the presence of 4 scaled and rotated rectangular obstacles.

the direct advantages of exploiting projections in ALSPG. The algorithm was run from 1000 different random initial configurations for both versions, and we compared the number of function and Jacobian evaluations. The results, shown in Table II, demonstrate that using projections significantly improves efficiency.

Robust IK: In this experiment, we would like to achieve a task of reaching and staying in the half-space under a plane whose slope is stochastic because, for example, of the uncertainties in the measurements of the vision system. The constraint can be written as $\boldsymbol{a}^\top \boldsymbol{f}(\boldsymbol{q}) \leq 0$, where $\boldsymbol{a} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. We can transform it into a chance constraint to provide some safety guarantees probabilistically. The idea is to find a joint configuration $\boldsymbol{q}$ such that it will stay under a stochastic hyperplane with a probability of $\eta \geq 0.5$. This inequality can be written as a second-order cone constraint wrt $\boldsymbol{f}(\boldsymbol{q})$ as $\boldsymbol{\mu}^\top \boldsymbol{f}(\boldsymbol{q}) + \Psi^{-1}(\eta)\|\boldsymbol{\Sigma}^{\frac{1}{2}} \boldsymbol{f}(\boldsymbol{q})\|_2 \leq 0$, where $\Psi(\cdot)$ is the cumulative distribution function of zero mean unit variance Gaussian variable. Defining $\boldsymbol{g}(\boldsymbol{q}) = \begin{bmatrix}(\boldsymbol{\Sigma}^{\frac{1}{2}} f(\boldsymbol{q}))^\top & \boldsymbol{\mu}^\top \boldsymbol{f}(\boldsymbol{q})\end{bmatrix}^\top$, the optimization problem can then be defined as

$$\min_{\boldsymbol{q} \in \mathcal{C}_{\boldsymbol{q}}} \|\boldsymbol{q} - \boldsymbol{q}_0\|_2^2 \quad \text{s.t.} \quad \boldsymbol{g}(\boldsymbol{q}) \in \mathcal{C}_{\text{SOC}}, \tag{11}$$

which can be solved efficiently without using second-order cone (SOC) gradients, by using the proposed algorithm. We tested the algorithm on the 3-axis robot shown in Fig. 5b by optimizing for a joint configuration with a probability of $\eta = 0.8$ and then computing continuously the constraint violation for the last 1000 time steps by sampling a line slope from the given distribution. We obtained a constraint violation percentage of around 80%, as expected.

B. Motion planning and MPC on planar push

Non-prehensile manipulation has been widely studied as a challenging task for model-based planning and control, with the pusher-slider system as one of the most prominent examples [21] (see Fig. 6). The reasons include hybrid dynamics with various interaction modes, underactuation and contact uncertainty. In this experiment, we study motion planning and MPC on this planar push system, without any constraints, to compare to a standard iLQR implementation. The cost function includes the control effort and the $L2$ norm measuring the difference between the final and target

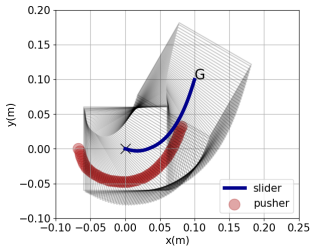
TABLE III: Comparison of MPC with iLQR and ALSPG for planar push.

Method	Conv. time [s]	Num. of fun. eval.	Num. of Jac. eval.
iLQR	14.5 ± 1.3	26689.5 ± 1830.5	6104.0 ± 1455.6
ALSPG	2.9 ± 0.5	225.9 ± 27.4	78.4 ± 8.5

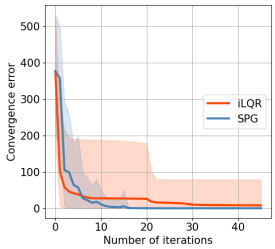
configurations. Fig. 6b illustrates the cost convergence for iLQR and ALSPG across 10 different targets for statistical analysis. Although iLQR seems to converge to medium accuracy faster than ALSPG, because of the difficulties in the task dynamics, it seems to get stuck at local minima very easily. On the other hand, ALSPG performs better in terms of variance and local minima. We applied MPC with iLQR and ALSPG with a horizon of 60 timesteps and stopped the MPC as soon as it reached the goal position with a desired precision. Table III shows this comparison in terms of convergence time (s), number of function evaluations and number of Jacobian evaluations. According to these findings, ALSPG performs better than a standard iLQR, even when there are no constraints in the problem.

C. Motion planning with obstacle avoidance

Obstacle avoidance problems are usually described using geometric constraints. In autonomous parking tasks, obstacles and cars are usually described as 2D rectangular objects. In this experiment, we take a 2D double integrator point car reaching a target pose in the presence of rectangular obstacles (see Fig. 5c). We apply the ALSPG algorithm with and without projections to illustrate the main advantages of having an explicit projection function over direct constraints. The main difference is without projections, the solvers need to compute the gradient of the constraints, whereas with projections, this is not necessary. In order to understand the differences between first-order and second-order methods, we also compared ALSPG-Proj to AL-SLSQP with projections (SLSQP-Proj.), which is the same algorithm except the subproblem is solved by a second-order solver SLSQP from Scipy [22]. The box obstacles allow analytical projections



(a) Pusher-slider system path optimized by the proposed algorithm to go from the state $(0, 0, 0)$ to $(0.1, 0.1, \pi/3)$. Optimal control solved with SPG results in a smooth path for the pusher-slider system.



(b) Convergence error mean and variance plot for iLQR and ALSPG motion planning algorithm for 10 different goal conditions starting from the same initial positions and control commands.

Fig. 6: ALSPG algorithm applied to a pusher-slider system.

TABLE IV: Comparison of MPC with iLQR and ALSPG for planar push.

Method	Conv. time [s]	Num. of fun. eval.	Num. of Jac. eval.
ALSPG with Proj.	**392.0 ± 66.2**	**332.0 ± 60.6**	**187.8 ± 34.6**
SLSQP with Proj.	679.0 ± 260.0	438.4 ± 200.7	389.2 ± 164.3
ALSPG without Proj.	2780.0 ± 530.9	571.8 ± 109.9	399.0 ± 93.9

and distance computation. We performed 5 experiments, each with different settings of 4 rectangular obstacles and compared the convergence properties. The results are given in Table IV. The comparison, with and without projections, reports a clear advantage of using projections instead of plain constraints in the convergence properties. Although the convergence time comparison is not necessarily fair for SPG implementations as the SLSQP solver calls C++ functions, the comparison of ALSPG-Proj. and SLSQP-Proj. shows that ALSPG-Proj. still achieves lower convergence time. Additionally, we compare it against the optimization-based collision avoidance (OBCA) algorithm [23] that is based on distance computation and IPOPT. The bicycle model is used $\dot{c}_x = v\cos\theta, \dot{c}_y = v\sin\theta, \dot{\theta} = \frac{v}{L}\tan\delta, \dot{v} = a$. where $L = 2.7$ m is the wheelbase length. The system control inputs $\boldsymbol{u} = [\delta, a]$. Other parameters remain the same as the baseline. As shown in Table V, the results further validate the efficiency of our approach.

D. Autonomous Navigation on a 1:10 Scale Car

To further assess the effectiveness of our approach, we tested the ALSPG algorithm on a 1:10 scale vehicle executing a navigation task. The experiment was conducted on a 1:10 car, using an Intel NUC14 ProU7 as the onboard computer. The sensor suite includes a Hokuyo UST-10LX LiDAR with a maximum scan frequency of 40 Hz. Odometry is provided by the VESC. Sensor fusion combines data from the LiDAR, the IMU embedded in the VESC, and odometry, utilizing the Cartographer to localize the vehicle and obtain its state. A pure-pursuit controller is implemented to track the given trajectory. The projections are constructed through polytopic projections and convex decomposition from section IV. The results shown in Fig. 7 demonstrate the effectiveness of our method.

TABLE V: Comparison between IPOPT and ALSPG on 100 random parking tests.

Scenarios	Algorithm	Solver	Com. time			
			mean	std	min	max
Vertical Parking	OBCA	IPOPT	655	167	380	1027
	Polytopic Proj.	ALSPG	**222**	**130**	**52**	**885**
	Polytopic Proj.	SLSQP	837	221	204	1319
Parallel Parking	OBCA	IPOPT	708	199	368	1346
	Polytopic Proj.	ALSPG	**470**	**136**	**199**	**909**
	Polytopic Proj.	SLSQP	942	249	227	1894

E. MPC for Real-Time Tracking on 7-axis Manipulator

We tested the ALSPG algorithm on the MPC problem of tracking an object with box constraints on the end-effector position of a Franka robot (see Fig. 9). An Aruco marker on the object is tracked by a camera held by another robot. In this experiment, the goal is to show the real-time applicability of the proposed algorithm for a constrained problem in the presence of disturbances. In Fig. 8, the error of the constraints and the objective function using formulation (8) is given for 1 min. time period of MPC with a short horizon of 50 timesteps. Between 20s and 30s, the robot is disturbed by the user thanks to the compliant torque controller run on the robot. We can see that the algorithm drives smoothly the error to zero, see the accompanying video.

F. Chess Robot

We validated our algorithm through a three-month experimental study using a 6-axis P-Rob robot from F&P Robotics to solve an inverse kinematics problem (position and orientation) with one degree of freedom (DoF) in orientation left unconstrained. The task involved grasping chess pieces, where the robot autonomously determined its orientation around the z-axis using SPG, compensating for workspace limitations that prevented full 6-DoF positioning, as shown in Fig. 1. The quaternion error, expressed via log-mapping, introduced nonlinearity and complexity, while joint limit projections ensured feasibility. After three months of public demonstrations, the system achieved a 100% success rate.

VI. Conclusion

In this work, we presented a fast first-order constrained optimization framework based on geometric projections, and applied it to various robotics problems ranging from inverse kinematics to motion planning. We showed that many of the geometric constraints can be rewritten as a logical combination of geometric primitives onto which the projections admit analytical expressions. We built an augmented Lagrangian method with spectral projected gradient descent as a subproblem solver for constrained optimization. We demonstrated: 1) the advantages of using projections when compared to setting up the geometric constraints as plain constraints with gradient information to the solvers; and 2) the advantages of using spectral projected gradient descent based motion planning compared to a standard second-order iLQR and IPOPT algorithm through different robot experiments. Sample-based MPC have been increasingly popular in recent years thanks to their fast practical implementations, despite

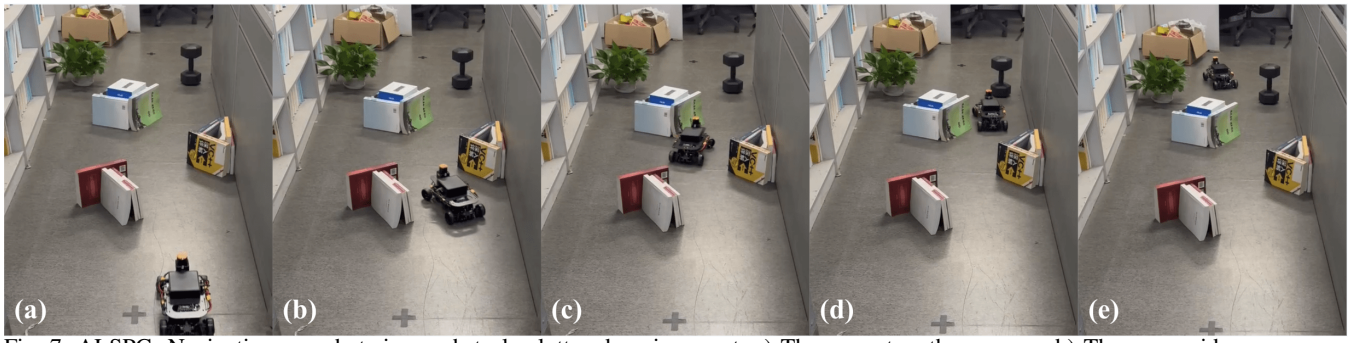


Fig. 7: ALSPG: Navigation snapshots in an obstacle-cluttered environment. a) The car enters the passage. b) The car avoids non-convex and triangular obstacles. c) A box-shaped obstacle blocks the left side. d-e) The car navigates avoiding the box obstacle to reach the goal.

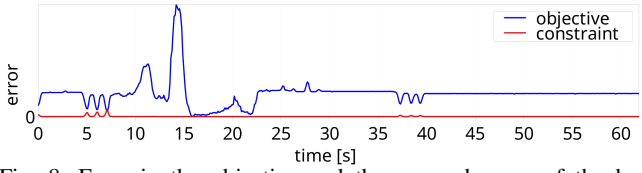


Fig. 8: Error in the objective and the squared norm of the box constraint value during 1 min execution of MPC on Franka robot.

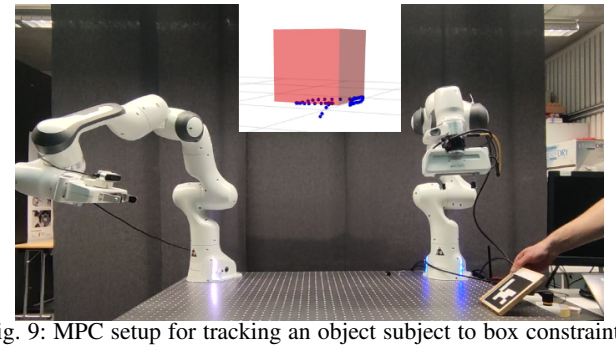


Fig. 9: MPC setup for tracking an object subject to box constraints.

their lack of theoretical guarantees. In contrast, second-order methods for MPC require a lot of computational power but with somewhat better convergence guarantees. We argue that ALSPG, being already in between these two methodologies in terms of these properties, promises great future work to combine it with sample-based MPC to further increase its advantages on both sides.

References

[1] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization." *SIAM J. Optimization*, vol. 12, no. 4, pp. 979–1006, 2002.

[2] D. Kraft, "A software package for sequential quadratic programming," *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, 1988.

[3] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, LANCELOT*: a Fortran package for large-scale nonlinear optimization (Release A)*. Heidelberg, Berlin, New-York: Springer-Verlag, 1992.

[4] A. Wachter, "An interior point algorithm for large-scale nonlinear optimization with applications in process engineering," Ph.D. dissertation, Carnegie Mellon University, 2002.

[5] Y. Aoyama, G. Boutselis, A. Patel, and E. A. Theodorou, "Constrained differential dynamic programming revisited," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2021, pp. 9738–9744.

[6] J. N. Nganga, H. Li, and P. M. Wensing, "Second-order differential dynamic programming for whole-body mpc of legged robots," *IFAC-PapersOnLine*, vol. 56, no. 3, pp. 499–504, 2023.

[7] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Proc. Robotics: Science and Systems (RSS)*, vol. 9, no. 1, 2013, pp. 1–10.

[8] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2009, pp. 489–494.

[9] E. G. Birgin, J. M. Martínez, and M. Raydan, "Spectral projected gradient methods: review and perspectives," *Journal of Statistical Software*, vol. 60, pp. 1–21, 2014.

[10] R. Andreani, E. G. Birgin, J. M. Martínez, and M. L. Schuverdt, "On augmented lagrangian methods with general lower-level constraints," *SIAM Journal on Optimization*, vol. 18, no. 4, pp. 1286–1309, 2008.

[11] E. G. Birgin and J. M. Martínez, *Practical augmented Lagrangian methods for constrained optimization*. SIAM, 2014.

[12] X. Jia, C. Kanzow, P. Mehlitz, and G. Wachsmuth, "An augmented lagrangian method for optimization problems with structured geometric constraints," *Mathematical Programming*, pp. 1–51, 2022.

[13] M. Schmidt, E. Berg, M. Friedlander, and K. Murphy, "Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm," in *Artificial intelligence and statistics*. PMLR, 2009, pp. 456–463.

[14] H. H. Bauschke, "Projection algorithms and monotone operators," Ph.D. dissertation, Theses (Dept. of Mathematics and Statistics)/Simon Fraser University, 1996.

[15] H. H. Bauschke, P. L. Combettes, *et al.*, *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2011, vol. 408.

[16] I. Usmanova, M. Kamgarpour, A. Krause, and K. Levy, "Fast projection onto convex smooth constraints," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 476–10 486.

[17] H. H. Bauschke and V. R. Koch, "Projection methods: Swiss army knives for solving feasibility and best approximation problems with halfspaces," *Contemporary Mathematics*, vol. 636, pp. 1–40, 2015.

[18] J. P. Boyle and R. L. Dykstra, "A method for finding projections onto the intersection of convex sets in hilbert spaces," in *Advances in order restricted statistical inference*. Springer, 1986, pp. 28–47.

[19] G. Torrisi, S. Grammatico, R. S. Smith, and M. Morari, "A projected gradient and constraint linearization method for nonlinear model predictive control," *SIAM Journal on Control and Optimization*, vol. 56, no. 3, pp. 1968–1999, 2018.

[20] M. Giftthaler and J. Buchli, "A projection approach to equality constrained iterative linear quadratic optimal control," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 61–66.

[21] T. Xue, H. Girgin, T. Lembono, and S. Calinon, "Demonstration-guided optimal control for long-term non-prehensile planar manipulation," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2023, pp. 4999–5005.

[22] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, *et al.*, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.

[23] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, 2020.