Propagative Distance Optimization for Motion Planning

Yu Chen[1], Jinyun Xu[1], Yilin Cai[2], Ting-Wei Wong[1], Zhongqiang Ren[3], Howie Choset[1], and Guanya Shi[1]

***Abstract*— This paper focuses on the motion planning problem for serial articulated robots with revolute joints under kinematic constraints. Many motion planners leverage iterative local optimization methods but are often trapped in local minima due to non-convexity of the problem. A key reason for the non-convexity is the trigonometric term when parameterizing the kinematics using joint angles. Recent distance-based formulations can eliminate these trigonometric terms by formulating the kinematics based on distances, and has shown superior performance against classic joint angle based formulations in domains like inverse kinematics (IK). However, distance-based kinematics formulations have not yet been studied for motion planning, and naively applying them for motion planning may lead to poor computational efficiency. In particular, IK seeks one configuration while motion planning seeks a sequence of configurations, which greatly increases the scale of the underlying optimization problem. This paper proposes Propagative Distance Optimization for Motion Planning (PDOMP), which addresses the challenge by (i) introducing a new compact representation that reduces the number of variables in the distance-based formulation, and (ii) leveraging the chain structure to efficiently compute forward kinematics and Jacobians of the robot among waypoints along a path. Test results show that PDOMP runs up to 10 times faster than the sampling-based and angle-based-optimization baseline methods.**

I. Introduction

Motion planning is a fundamental problem in robotics that seeks paths from a starting configuration to a goal configuration under various constraints, including collision avoidance, stability, and kinematic and dynamic feasibility. This paper focuses on the motion planning problem for serial articulated robots with revolute joints, which has been addressed by optimization [1], [2], search [3], [4], sampling [5], [6], [7], [8], [9] and learning-based approaches [10]. For the purpose of obtaining high-quality paths subject to various constraints in a high-dimensional configuration space, motion planning is often achieved by iterative local optimization, which can either generate a path from scratch or refine initial solutions provided by other planners. However, these methods can get slowed down or even trapped into local minima, resulting in infeasible or suboptimal solutions.

One way to handle the trigonometric constraint is to eliminate them by using distance-based kinematic formulations of the serial articulated robots [11], [12], [13], [14], [15], [16], [17]. The main idea is to attach spatial points to the robot and the obstacles, and formulate the kinematic model

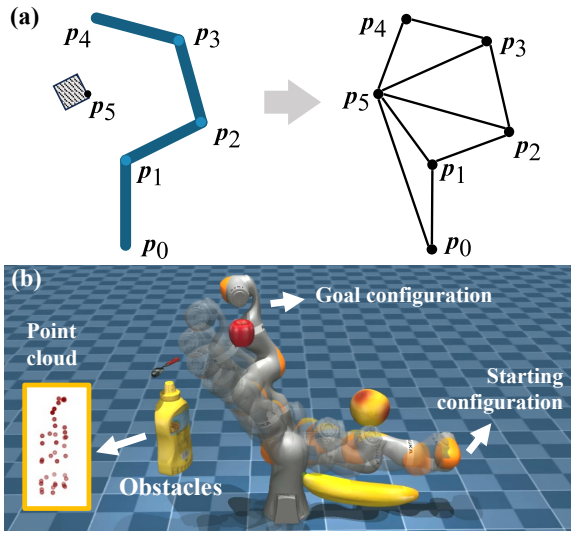

Fig. 1: PDOMP solves a motion planning problem using distance-based formulations which represent the robot and obstacles through attached points, and optimizes the distances between them to find a collision-free start-goal path.

based on the distances among these points instead of joint angles. Distance-based kinematic formulations were mainly used to solve inverse kinematic (IK) problems [11], [18], [19], and have demonstrated superior performance compared to joint-angle-based approaches [16], [14]. However, they have not yet been studied for motion planning problems, and naively applying them for motion planning may lead to poor computational efficiency. Specifically, IK seeks one configuration while motion planning seeks a path, i.e., a sequence of configurations, which greatly increases the scale of the underlying optimization problem. Most existing distance-based formulations exhibit quadratic or cubic complexity with respect to the scale of problem, and are thus prevented from being used in motion planning.

This paper addresses this challenge and proposes a new approach called PDOMP (Propagative Distance Optimization for Motion Planning) that extends distance-based formulations to motion planning in a computationally efficient way. PDOMP gains computational efficiency for the following two reasons. First, this paper introduces a new angle representation for distance-based methods, which reduces the number of distances (in other words variables) required to represent a joint angle, and expedites the optimization. Second, while our prior work on IK [20] introduced a propagation method to leverage the chain structure of serial robots to efficiently compute forward kinematics and Jacobians for a single configuration, PDOMP further extends this idea to

[1]Roboitcs Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA. {yuchen3, jinyunx, andrewwong, choset, guanyas}@andrew.cmu.edu
[2]Georgia Institute of Technology, Atlanta, GA 30332 USA. yilincai@gatech.edu
[3]Shanghai Jiao Tong University, Shanghai, 200240 China. zhongqiang.ren@sjtu.edu.cn

achieve propagation between waypoints (i.e., configurations) along the entire path during motion planning, which further expedites the computation.

We compare PDOMP with both optimization- and sampling-based methods as baselines on various robots in simulation. The results show that our method runs up to 10 times faster than the baselines. We then conduct an ablation study, which shows that, the new representation of joint angles speeds up the distance-based optimization by a factor ranging from 3.1 to 3.8, while the propagation technique boosts speed by 100 to 400 times.

II. Related Works

A. Distance-Based Kinematic Formulations

Distance-based formulations assign spatial points to the robot links and obstacles, represeting the joint angles using the distances between these points. The effectiveness of distance-based formulations was verified in inverse kinematics (IK). Josep et al. [11] used a distance matrix to formulate the kinematic model of serial robots [18] and solved the IK problem using matrix completion approach. Han et al. [12] parameterized the robot kinematic model with a combination of anchored diagonal lengths and triangle orientations. Marić, Giamou, et al. [16] used sparse bounded-degree sum of squares relaxations [21] to solve IK problems for spherical joint robots. Marić et al. later proposed a distance-geometric framework called Riemannian Trust Region (R-TR) [14] to address the constrained IK using Riemannian optimization.

B. Motion Planning for Serial Articulated Robots

Search-based methods [3], [4] runs systematic graph search in a discretized representation of the robot's configuration space (i.e., C-space), which enjoys completeness and solution quality guarantees but suffers from the high-dimensionality of the C-space. Sampling-based motion planners, such as Rapidly-exploring Random Tree (RRT) [5], Probabilistic Roadmap Method (PRM) [6], and their variants [7], [8], [9], construct a graph in the configuration space by randomly sampling points from the C-space, and connecting the sampled points with feasible paths. These methods are efficient in high-dimensional spaces but often produce paths of poor quality without fine tuning, resulting in jagged or non-smooth trajectories. Local optimization methods iteratively refine a given initial trajectory based on gradient information, which can overcome high-dimensionality of the C-space but may get trapped into local minima. To name a few, Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [1] is a first-order optimization-based method that uses functional gradient techniques to iteratively improve the quality of the initial path. CHOMP often finds high-quality paths but suffers from slow computational speed due to the slow convergence of the first-order optimization. Trajectory Optimization for Motion Planning (TrajOpt) [2] is a second-order optimization method that uses sequential quadratic programming (SQP) to solve the trajectory optimization problem, which speeds up the computation.

III. Formulations

This section first introduces basic concepts and notations in Sec. III-A and reviews the kinematic formulation and distance graph construction using the spatial points attached to the robot in Sec. III-B, then formulates the constraints at each waypoint in Sec. III-C. After presenting the single-waypoint formulation, we proceed to model the decision sequence in Sec. III-E. For further details on Sections III-B and III-C, please refer to our previous work on IK [20].

A. Basic Concepts and Notations

Our method aims to find a discrete path with N waypoints for a serial articulated robot with M degrees of freedom (DoF)s that connects the starting configuration to the goal configuration. Each waypoint in the path is represented by variables $\boldsymbol{\omega}_k \in \mathbb{R}^M$ $(k = 1, \ldots, N)$, defined in the configuration space $\mathcal{C}$ (which is also referred to as joint space hereafter), where k is the index of the waypoint. We consider a linear system $\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \mathbf{s}_k$, where $\mathbf{s}_k$ is the control input, i.e., the change of variables ω_k, at each waypoint. We consider kinematic constraints, collision avoidance constraints, terminal state constraints, and the objectives of minimizing the path length in joint space and ensuring similarity to the initial path. Notably, our method has the potential to include additional constraints, such as robot dynamics.

The spatial Cartesian frame of the robot link is denoted by $\mathcal{F}_i$, with the frame axes represented as $\mathbf{x}_i$, $\mathbf{y}_i$, and $\mathbf{z}_i$, where i is the index of robot link. The frames are attached to the robot links in the same way as the Denavit–Hartenberg (DH) parameter. Specifically, the origin of $\mathcal{F}_i$ is located at the corresponding robot joint of the ith link. The relative transformation between $\mathcal{F}_{i_2}$ and $\mathcal{F}_{i_1}$ is denoted by ${}^{i_2}\mathbf{T}_{i_1} \in SE(3)$. The transformation of $\mathcal{F}_i$ with respect to the world frame is $\mathbf{T}_i \in SE(3)$. We use a_{i-1} to denote the revolute radius of $\mathcal{F}_i$ with respect to the axis $\mathbf{z}_{i-1}$, and α_{i-1} to denote the angle between $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$ about the common normal. We use d_i to represent the offset of $\mathcal{F}_i$ relative to $\mathcal{F}_{i-1}$ along $\mathbf{z}_i$. We use $s_{(\cdot)}$ and $c_{(\cdot)}$ to denote the sine and cosine functions of $(\cdot)$, respectively.

B. Graph of Distance

An example of a serial robot and an obstacle is shown in Fig. 2a. Instead of using joint angles as decision variables, we attach spatial points to the robot and obstacles and use the distances between spatial points to parameterize the robot kinematics. We attach points to the robot's base and joint frames to fully capture the spatial transformations of all the rigid bodies. Obstacles are represented as clusters of points, inspired by the common use of LiDAR or depth cameras that detect and represent environmental obstacles as point clouds.

These spatial points are encoded into a distance graph $G = (V, E)$. The vertices V in the distance graph represent the positions of the attached spatial points, while the edges E represent constraints related to robot kinematics and collision avoidance. For computational efficiency, edges are

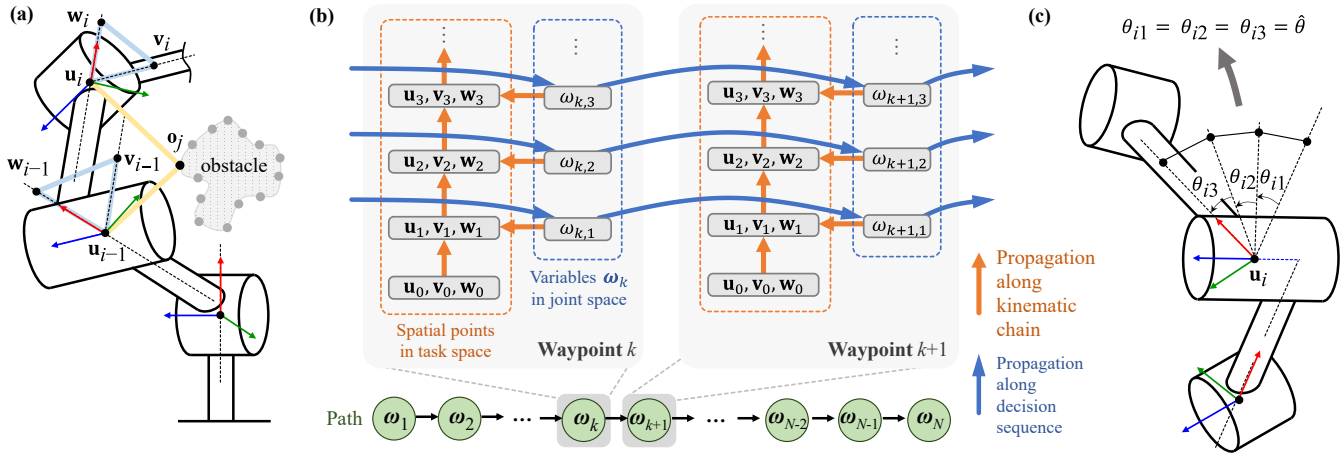

Fig. 2: Overview of the (a) constraints formulation based on spatial points attached on the robot link frames and obstacles, (b) propagative computation process along kinematic chain and decision sequence, and (c) the compact representation approach.

only connected between two vertices if they satisfy one of the following conditions:

- Both vertices are attached to neighboring robot links. These edges represent the kinematics constraints.
- One vertex is attached to the robot and the other is attached to an obstacle. These edges represent the collision avoidance constraint.

C. Constraints at One Waypoint

1) Robot kinematics constraints: The constraints of robot kinematics formulate the relative transformation between robot joints that are connected to each other. We formulate the kinematic constraints by modifying the proximal DH convention [22] with Euclidean distances between points. We attach three points to the frame of the ith joint $\mathcal{F}_i$. The first point $\mathbf{u}_i$ is attached to the origin to represent the spatial position of joint i. The second point $\mathbf{v}_i$ is of distance l_i^{uv} away from $\mathbf{u}_i$ in the direction of $\mathbf{x}_{i-1}$. The third point $\mathbf{w}_i$ is of distance l_i^{uw} away from $\mathbf{u}_i$ in the direction of $\mathbf{x}_i$. We define L_i as the squared distance between $\mathbf{v}_i$ and $\mathbf{w}_i$. By defining $l_i^{uv} = l_i^{uw} = 1/\sqrt{2}$, the relationship between the joint angle θ_i and the squared distance L_i is

$$\cos\theta_i = 1 - L_i \tag{1}$$

The relative transformation ${}^{i-1}\mathbf{T}_i$ is:

$${}^{i-1}\mathbf{T}_i = \begin{bmatrix} 1-L_i & -(2L_i - L_i^2)^{\frac{1}{2}} & 0 & a \\ c_\alpha(2L_i - L_i^2)^{\frac{1}{2}} & c_\alpha(1-L_i) & -s_\alpha & d_i s_\alpha \\ s_\alpha(2L_i - L_i^2)^{\frac{1}{2}} & s_\alpha(1-L_i) & c_\alpha & d_i c_\alpha \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$\mathbf{T}_i$ is computed by recursively multiplying from the base frame to ${}^{i-1}\mathbf{T}_i$. :

$$\mathbf{T}_i = \mathbf{T}_0 \prod_{p=1}^{i} {}^{p-1}\mathbf{T}_p \tag{3}$$

The position of the point attached on the ith joint frame origin $\mathbf{u}_i$ can be extracted from $\mathbf{T}_i$.

We also consider the joint limits as box constraints of the squared distance $L_i \in [L_i^{\min}, L_i^{\max}]$. We convert the box constraint to equality constraints leveraging the Sigmoid function intorduced in [23] and the slack variable ω_i to bound the squared distance L_i within $(L_i^{\min}, L_i^{\max})$, which is a close approximation to $[L_i^{\min}, L_i^{\max}]$:

$$L_i = s(\omega_i) = \left(L_i^{\max} - L_i^{\min}\right)\sigma(\omega_i) + L_i^{\min} \tag{4}$$

where $\sigma(\omega_i) = 1/\left(1 + e^{-\omega_i}\right) : \mathbb{R} \to (0, 1)$.

2) Collision avoidance constraints: Collision avoidance constraints ensure that the robot does not overlap with obstacles. These constraints are formulated as inequality constraints on the distances between points attached to the obstacles and the robot. For any spatial points attached on the obstacle, denoted as o_j:

$$d_{ij}^{\text{joint}} = r_i - ||\mathbf{u}_i - \mathbf{o}_j|| \leq 0 \tag{5a}$$

$$d_{ij}^{\text{link}} = 2a_i - (||\mathbf{u}_i - \mathbf{o}_j|| + ||\mathbf{u}_{i-1} - \mathbf{o}_j||) \leq 0 \tag{5b}$$

where j indicates the index of obstacle points. Eq. 5a and Eq. 5b represent collision avoidance constraints between points attached to the obstacle and joint i and link i, respectively. We summarize Eq. 5a and Eq. 5b among all waypoints as $d(\boldsymbol{\omega}_k) \leq 0$.

D. Compact Representation of Joint Angles

We now introduce a new angle representation that reduces redundant variables in the distance-based method, reducing the scale of the optimization problem.

1) Redundancy in the formulation: The L_i in Eq. 2 is a bijection of the classical joint-angle-based DH convention with the precondition that the joint angle lies within $[0, \pi]$. However, this precondition is usually too strong, as the joint angle limits in many robotic arms exceed this range. As a result, Eq. 1 no longer holds. In [20], this issue is addressed by angle decomposition. As shown in Fig. 2c, the angle decomposition technique divides a joint angle, which may exceed the range of $[0, \pi]$, into multiple sub-angles that can be translated into distance representations. For a joint angle θ_i, let $\theta_i^{\min} = 0$ and $k = \theta_i^{\max}/\pi > 0$, such that $\theta_i \in [0, k\pi]$. The angle θ is then divided into $\lceil k \rceil$ sub-angles θ_{im}, each of which lies within the range $[0, \pi]$: $\theta_i = \sum_{m=1}^{\lceil k \rceil} \theta_{im}$ ($\theta_{im} \in \left[\theta_i^{\min}/\lceil k \rceil, k\pi/\lceil k \rceil\right] \subseteq [0, \pi]$), where $m = 1, \ldots, \lceil k \rceil$ is the index of sub-angles divided by the angle θ_i.

For a sub-angle θ_{im}, the corresponding squared distance L_{im} and slack variable ω_{im} can then be applied by inserting

θ_{im} into Eq. 1. Decomposing a single joint angle into multiple sub-angles increases the dimensionality of the kinematic model, which can slow down computation.

2) Compact Representation: For all sub-angles θ_{im}, we assume they are equal and use only one variable, $\hat{\theta}$, to represent them. This assumption holds during the iterative optimization process under the following conditions:

- The optimizer is a general first- or second-order iterative local optimizer, which determines its optimization direction and step size using the Jacobian, Hessian, or hyper-parameters that are independent to variables.
- The initial values of the sub-angles, divided by the same joint angle, are the same.

Now we show that this assumption always holds during the optimization. For a sub-angle θ_{im} and its next neighbouring sub-angle $\theta_{i,m+1}$, the derivative of a cost function g with respect to them can be written as:

$$\frac{\partial g}{\partial \theta_{i,m+1}} = \sum \left(\frac{\partial g}{\partial {}^{m}\mathbf{T}_{m+1}} \odot \frac{\partial {}^{m}\mathbf{T}_{m+1}}{\partial \theta_{i,m+1}} \right) \tag{6}$$

$$\frac{\partial g}{\partial \theta_{im}} = \sum \left(\frac{\partial g}{\partial {}^{m-1}\mathbf{T}_{m}} \odot \frac{\partial {}^{m-1}\mathbf{T}_{m}}{\partial \theta_{im}} \right) \tag{7}$$

where g can be a cost function such as the one stated in Eq. 5a, Eq. 5b, or other types of cost functions, including end-effector pose cost and center-of-mass cost in [20]. Here $\sum(\cdot)$ indicates summing the elements in the matrix $(\cdot)$.

Utilizing the spatial transformation stated in Eq. 3, we rewrite $\partial g/\partial {}^{m}\mathbf{T}_{m+1}$ and $\partial g/\partial {}^{m-1}\mathbf{T}_{m}$ as:

$$\frac{\partial g}{\partial {}^{m}\mathbf{T}_{m+1}} = {}^{m-1}\mathbf{T}_{m}{}^{\top} \cdot \left(\mathbf{T}_{m-1}^{\top} \cdot \frac{\partial g}{\partial \mathbf{T}_{m+1}} \right) \tag{8}$$

$$\frac{\partial g}{\partial {}^{m-1}\mathbf{T}_{m}} = \left(\mathbf{T}_{m-1}^{\top} \cdot \frac{\partial g}{\partial \mathbf{T}_{m+1}} \right) \cdot {}^{m}\mathbf{T}_{m+1}{}^{\top} \tag{9}$$

For simplicity, let $K^n = {}^{n-1}\mathbf{T}_{n}{}^{\top}$ and $P^n = \partial {}^{n-1}\mathbf{T}_{n}/\partial \theta_{in}$. We then use $Q \in \mathbb{R}^{4\times4}$ to denote $\mathbf{T}_{m-1}^{\top} \cdot (\partial g/\partial \mathbf{T}_{m+1})$. We use q_{xy} to denote the element in Q, where x and y are row and column index. As a result, we have $\partial g/\partial \theta_{i,m+1} = \sum((K^m \cdot Q) \odot P^{m+1})$ and $\partial g/\partial \theta_{im} = \sum((Q \cdot K^{m+1}) \odot P^m)$. Note that $\theta_{i,m+1}$ and θ_{im} share the same revolute axis, and that $\mathbf{T}_{m+1}$ shares the same origin with $\mathbf{T}_m$, we have $\alpha_m = 0$, $a_m = 0$, and $d_{m+1} = 0$. If $\theta_{m+1} = \theta_m = \hat{\theta}$, we get the equivalence relationship between $\sum((K^m \cdot Q) \odot P^{m+1})$ and $\sum((Q \cdot K^{m+1}) \odot P^m)$:

$$\begin{aligned} &\sum((K^m \cdot Q) \odot P^{m+1}) = \sum((Q \cdot K^{m+1}) \odot P^m) \\ =& - q_{11} \cdot s(2\hat{\theta}) - q_{12} \cdot c(2\hat{\theta}) - q_{21} \cdot c\alpha_{m-1} \cdot c(2\hat{\theta}) \\ &- q_{22} \cdot c\alpha_{m-1} \cdot s(2\hat{\theta}) - q_{31} \cdot s\alpha_{m-1} \cdot c(2\hat{\theta}) \\ &- q_{32} \cdot s\alpha_{m-1} \cdot s(2\hat{\theta}) \end{aligned} \tag{10}$$

which indicates $\partial g/\partial \theta_{i,m+1}$ and $\partial g/\partial \theta_{im}$ are equal if the value of the corresponding sub-angles are the same:

$$(\partial g/\partial \theta_{i,m+1})|_{\theta_{i,m+1}=\hat{\theta}} = (\partial g/\partial \theta_{im})|_{\theta_{im}=\hat{\theta}} \tag{11}$$

The second order derivative of g with respect to $\hat{\theta}$ can be computed by Eq. 10. The second order derivative remains the same if $\theta_{i,m+1} = \theta_{im} = \hat{\theta}$. Given that all sub-angles are bijections to squared distance L_{im} and slack variable ω_{im}, same conclusion can be drawn for distance representations.

In an iterative local optimization algorithm, by setting the initial value of all sub-angles divided from the same joint angle to be equal to $\hat{\theta}_{\text{iter}=0}$, we obtain the same first and second derivatives for these sub-angles. Since the update rule depends solely on the Jacobian, Hessian, or hyper-parameters that are independent of the variables, the values of the sub-angles will remain identical in subsequent iterations.

This compact representation of variables eliminates the redundant variables introduced by the angle decomposition technique. It reduces the size of the Jacobian and Hessian matrices, thereby improving computational efficiency, especially when the range of joint angles is large and requires division into many sub-angles.

E. Constraints of the Path

We focus on three kinds of constraints for the path. The path length objective is defined as the squared sum of decision variables along the path, instead of joint angles:

$$J_{\text{length}} = \sum_{1:N-1} \mathbf{s}_k^{\top} \mathbf{s}_k \tag{12}$$

The similarity between the optimized path and the initial solution is defined as:

$$J_{\text{initial}} = \sum_{1:N-1} \left(\mathbf{s}_k - \mathbf{s}_k^{\text{initial}} \right)^{\top} \left(\mathbf{s}_k - \mathbf{s}_k^{\text{initial}} \right) \tag{13}$$

which is often used when an initial solution from global planners is provided. To ensure the last waypoint reaches the goal, we have a terminal waypoint constraint:

$$c(\boldsymbol{\omega}_N) = (\boldsymbol{\omega}_N - \boldsymbol{\omega}_N^*)^{\top} (\boldsymbol{\omega}_N - \boldsymbol{\omega}_N^*) = 0 \tag{14}$$

F. Propagation Computation Along the Path

As shown in Fig. 2b, we compute the robot configuration in each waypoint propagatively along both the decision sequence and the kinematic chain. This propagation occurs hierarchically. First, we propagate through the decision sequence to compute the joint space variables, $\boldsymbol{\omega}_1$. Then, for each waypoint, we compute task-space constraints, such as collision avoidance, based on the joint space values.

The forward rollout of the decision sequence starts from the initial waypoint, $\boldsymbol{\omega}_1$, and iteratively computes $\boldsymbol{\omega}_k$ as $\boldsymbol{\omega}_{k-1} + \mathbf{s}_{k-1}$. Within each waypoint, propagation begins from three spatial points, $\mathbf{u}_0$, $\mathbf{w}_0$, and $\mathbf{v}_0$, which are attached to the base frame. Since these points are typically stationary relative to the world frame, their positions are assumed to be known. In our approach, we group $\mathbf{u}_i$, $\mathbf{w}_i$, and $\mathbf{v}_i$ into a "unit", solving for the variables in the ith unit before moving to the $(i+1)$th unit. Furthermore, the variables computed in the ith unit reuse pre-computed results from the $(i-1)$th unit. The propagation technique is also applied to Jacobian computation, where we propagate backward along the decision sequence and kinematic chain using reverse accumulation. Importantly, this Jacobian computation occurs after the forward rollout, enabling the reuse of intermediate results from the forward pass, which enhances computational efficiency.

The propagation process within a waypoint maps between the joint and task space of the kinematic model, which is in fact the forward kinematics and Jacobian computation

process that has already been stated in [20]. Here we denote them as h_{FK} and h_{Jacob}:

$$d(\boldsymbol{\omega}_k) = h_{\mathrm{FK}}(\boldsymbol{\omega}_k) \quad (15a) \qquad \nabla_{\boldsymbol{\omega}_k} d(\boldsymbol{\omega}_k) = h_{\mathrm{Jacob}}(\boldsymbol{\omega}_k) \quad (15b)$$

IV. Algorithm

The motion planning problem is formulated as a local optimization problem over $\mathbf{s}_{1:N-1}$:

$$\begin{aligned} \min_{\mathbf{s}_{1:N-1}} \quad & J = \beta_1 J_{\mathrm{initial}} + \beta_2 J_{\mathrm{length}} \\ \text{s.t.} \quad & \boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \mathbf{s}_k,\ c(\boldsymbol{\omega}_N) = 0,\ d(\boldsymbol{\omega}_{1:N}) \le 0 \end{aligned} \tag{16}$$

where β_1 and β_2 are the weight parameters.

The equality constraint $\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \mathbf{s}_k$ is inherently handled in the forward rollout. Let $d'(\boldsymbol{\omega}) = \max(0, d(\boldsymbol{\omega}_{1:N}))$, we use the augmented Lagrangian method to offload the constraints into the objective function:

$$\begin{aligned} L_\rho =& J + \boldsymbol{\lambda}^\top c(\boldsymbol{\omega}_N) + \boldsymbol{\mu}^\top d'(\boldsymbol{\omega}_{1:N}) + \frac{\rho}{2} c(\boldsymbol{\omega}_N))^\top c(\boldsymbol{\omega}_N)) \\ & + \frac{\rho}{2} d'(\boldsymbol{\omega}_{1:N}))^\top d'(\boldsymbol{\omega}_{1:N})) \end{aligned} \tag{17}$$

where $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are the Lagrangian multipliers and ρ is the adjust penalty parameter. The constrained IK problem is formulated as finding $\mathbf{s}^*_{1:N-1} = \mathrm{argmin}\ L_\rho$.

The Jacobian of L_ρ with respect to the variable $\boldsymbol{\omega}_k$ is

$$\begin{aligned} \nabla_{\boldsymbol{\omega}_k} L_\rho =& (\boldsymbol{\mu}^\top + \rho d'(\boldsymbol{\omega}_k)) \nabla_{\boldsymbol{\omega}_k} d(\boldsymbol{\omega}_k) \\ & + \mathbb{1}(k = N) \cdot (\boldsymbol{\lambda}^\top + \rho c(\boldsymbol{\omega}_k)) \nabla_{\boldsymbol{\omega}_k} c(\boldsymbol{\omega}_k) \end{aligned} \tag{18}$$

and the Jacobian of L_ρ with respect to the variable $\mathbf{s}_k$ is

$$\nabla_{\mathbf{s}_k} L_\rho = \nabla_{\boldsymbol{\omega}_{k+1}} L_\rho + 2\beta_1(\mathbf{s}_k - \mathbf{s}_k^{\mathrm{initial}}) + 2\beta_2 \mathbf{s}_k \tag{19}$$

As shown in Algorithm 1, we iteratively minimize L_ρ and update the Lagrangian multiplier $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ and ρ. The Hessian matrix computation and solution update rule are generated by Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) solver [24]. After solving for $\boldsymbol{\omega}^*_k = \mathrm{argmin} L_\rho$, we update Lagrangian multipliers $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ and ρ. Our method checks the L_1 normalization terminal state constraint $c(\boldsymbol{\omega})$ and collision avoidance constraints $d'(\boldsymbol{\omega})$ and terminate when both of them are below a certain threshold c_{tol} and d'_{tol}.

Algorithm 1 PDOMP

$\boldsymbol{\lambda} \leftarrow \mathbf{0}$, $\boldsymbol{\mu} \leftarrow \mathbf{0}$, $\rho \leftarrow 1$, $\phi \leftarrow 10$.
Initialize $\mathbf{s}$ with linear interpolation.
while $|d'(\boldsymbol{\omega}^*)| \ge d_{\mathrm{tol}}$ **or** $|c(\boldsymbol{\omega}^*)| \ge c_{\mathrm{tol}}$ **do**
  for $k = 2, ..., N$ **do**
    $\boldsymbol{\omega}_k \leftarrow \boldsymbol{\omega}_{k-1} + \mathbf{s}_{k-1}$
    Compute $\mathbf{u}_k$ using Eq. 15a.
    Compute $d(\boldsymbol{\omega}_k)$ with $\mathbf{u}_k$ using Eq. 5b and Eq. 5a.
  end for
  Compute $c(\boldsymbol{\omega}_N)$ using Eq. 14.
  Compute J_{length} and J_{initial} using Eq. 12 and Eq. 13.
  Compute L_ρ using Eq. 17.
  $\nabla_{\boldsymbol{\omega}_N} c(\boldsymbol{\omega}_N) \leftarrow 2(\boldsymbol{\omega}_N - \boldsymbol{\omega}^*_N)$.
  for $k = N, ..., 2$ **do**
    Compute $\nabla_{\boldsymbol{\omega}_k} d(\boldsymbol{\omega}_k)$ using Eq. 15b.
    Compute $\nabla_{\boldsymbol{\omega}_k} L_\rho$ using Eq. 18.
    if $k < N$ **then**
      Compute $\nabla_{\mathbf{s}_k} L_\rho$ using Eq. 19.
    end if
  end for
  Update $\mathbf{s}^*$ with L-BFGS based on L_ρ and $\nabla_{\mathbf{s}} L_\rho$.
  $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \rho c(\boldsymbol{\omega}^*)$, $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \rho d'(\boldsymbol{\omega}^*)$
  $\rho \leftarrow \phi\rho$
end while
Recover $\boldsymbol{\theta}^*$ using Eq.1.
return $\boldsymbol{\theta}^*$

V. Experimental Results

A. Experimental Settings and Baselines

We test all the methods on 3 popular robots in simulation: Franka, UR10, and KUKA. We set up various scenarios with 1 to 5 random obstacles, which are different objects sampled from YCB dataset as obstacles [25], with their corresponding point clouds provided. For each number of obstacles and each robot, we generate 100 scenarios for experiments. An example of the scenarios is shown in Fig. 1a. Each scenario is generated as follows. We first place the robot at the origin of the world frame and randomly sample a path of N waypoints from a uniform distribution over the space of joint angles. We then randomly generate obstacles within the workspace of the robot that are collision-free with respect to the random path to ensure the generated problem instance is feasible. The MoveIt! collision checker [26] is used to check for collisions. The point cloud of the obstacles is downsampled using a voxel grid filter with a grid leaf size of 0.1m.

We compare our method with CHOMP [1] and RRT-Connect [7]. We set $N = 100$ for both our method and CHOMP, and use linear interpolation for initialization. The parameters β_1, β_2 in Eq. 16 are set to 0 and 1 respectively. The constraint penalty tolerance c_{tol} and d'_{tol} are set to 10^{-4}. Additionally, we tested both our method and CHOMP using the solutions from RRT-Connect as the initial guess, denoted as "RRTConnect+Ours" and "RRTConnect+CHOMP," respectively. The objective function of these two methods is a weighted sum of the prior path similarity and the path length objective. We set β_1 in Eq. 16 as 1, and β_2 as 0.01 for the KUKA robot, and 0.1 for the UR10 and Franka robots. Both CHOMP and RRT-Connect are from the Moveit! motion planning library. All methods are implemented in C++ and tested on a desktop computer with an Intel Core i9 CPU and 128 GB of RAM.

B. PDOMP vs Baselines

We measure the performance of all methods using three criteria. First, we report the failure rate as the percentage of experiments that fail to provide a solution within the time limit, which is set to 10 seconds, or result in a solution that collides with the obstacles. The Moveit! collision checker [26] is used to verify collisions along the entire continuous path. Second, we report the logarithm of runtime, $\log_{10} T$, where T is the median value in milliseconds for all collision-free solutions. For the methods RRTConnect+Ours and RRT-Connect+CHOMP, the runtime is the sum of the runtimes of

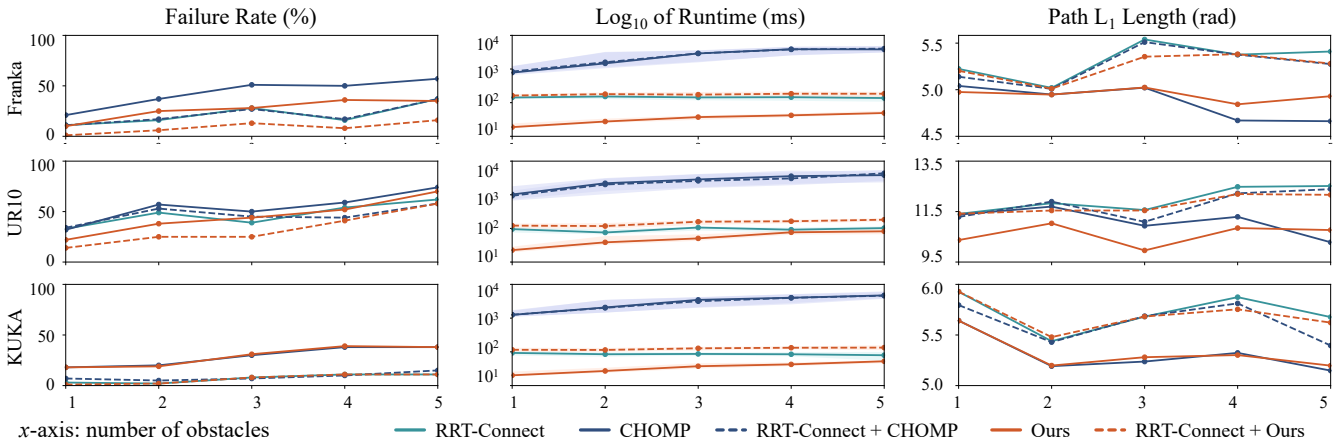

Fig. 3: We compare our method with baselines on three robot platforms: Franka, UR10, and KUKA. Three criteria are applied to measure the performance: failure rate, path length, and algorithm runtime. For all criteria, lower values indicate better performance.

TABLE I: Ablation studies on runtime of the algorithm

Method	Propagation	Compact Representation	Runtime - KUKA (ms)	Runtime - UR10 (ms)	Runtime - Franka (ms)
Full Pipeline	✓	✓	5.14 ± 1.58	17.67 ± 9.98	2.25 ± 0.38
Baseline 1	×	✓	912.95 ± 79.77	1831.76 ± 262.15	988.98 ± 137.96
Baseline 2	×	×	2834.08 ± 311.87	6975.25 ± 1429.96	3091.67 ± 358.24

RRTConnect and the respective local planners. Lastly, we report the median value of L_1 path length for all methods.

As shown in Fig. 3, our method with linear initialization is up to 100 times faster compared to CHOMP, and up to 10 times faster than RRT-Connect. When comparing with CHOMP, our method demonstrates a lower failure rate and a comparable path length. When comparing with RRT-Connect, our method achieves a shorter path length at the cost of a higher failure rate. This is because RRT-Connect, as a global planner, is more likely to find a feasible solution but tends to produce jagged paths.

When using RRT-Connect to provide an initial solution, our method achieves up to a 50% lower failure rate than RRT-Connect, while maintaining comparable runtime and slightly shorter path lengths. In contrast, RRT-Connect+CHOMP fails to achieve similar improvement: its failure rate is not lower than that of RRT-Connect, and its runtime is much longer.

This result highlights the advantages of our method: when building a path from scratch, our method achieves much faster speeds than the baselines. When updating an initial path provided by global planners, our method can improve the quality of the initial solution with almost negligible additional runtime.

C. Ablation Study of PDOMP

We then demonstrate the effectiveness of the two techniques introduced in this paper: The propagative computation (Sec. III-B) and compact representation of variables (Sec. III-D). We compare our method with two baselines: Baseline 1 eliminates the propagation process and instead uses finite differences to compute the Jacobian. Baseline 2 further eliminates the compact representation of variables and employs the same representation method as in [20]. All methods are tested on KUKA, UR10, and Franka robots. We set the number of obstacles to 1 and tested 50 random scenarios. The number of waypoints N is set to 10.

The comparison of runtime is shown in Tab. I. Baseline 1 is 3.1 to 3.8 times faster than Baseline 2 with similar success rate and path length. This is due to the application of compact variable representation. The full pipeline is up to 100 to 400 times faster than Baseline 1 due to the application of the propagation technique. The propagative computation greatly improves speed due to (1) its time efficiency, achieved through lower time complexity compared to finite difference [20], and (2) its space efficiency, enabled by reusing intermediate variables that avoids memory allocation overhead.

VI. Conclusion

This paper proposes a novel motion planning method PDOMP that extends distance-based formulations to motion planning in a computationally efficient way. PDOMP introduces a new angle representation for distance-based methods that reduces the number of variables needed. PDOMP also extends the propagation approach to achieve propagation computation between waypoints along the path. Both of these two techniques improve the runtime efficiency of the algorithm and lead to a fast motion planner. Although PDOMP still faces the limitation that its speed decreases as the number of obstacles increases, we believe this can be addressed through parallel collision checking approaches, which presents an exciting direction for future work. Furthermore, extending PDOMP to diverse robotic platforms (e.g., floating base robots and parallel robots) and constraints (e.g., dynamic constraints and end effector pose constraints) will be further considered in the future.

REFERENCES

[1] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 489–494.

[2] 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 *Robotics: science and systems*, vol. 9, no. 1. Berlin, Germany, 2013, pp. 1–10.

[3] B. J. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for manipulation with motion primitives," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 2902–2908.

[4] M. S. Saleem, R. Sood, S. Onodera, R. Arora, H. Kanazawa, and M. Likhachev, "Search-based path planning for a high dimensional manipulator in cluttered environments using optimization-based primitives," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8301–8308.

[5] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

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

[7] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[8] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," 2011.

[9] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 521–528.

[10] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *arXiv preprint arXiv:2303.04137*, 2023.

[11] J. M. Porta, L. Ros, and F. Thomas, "Inverse kinematics by distance matrix completion," 2005.

[12] L. Han and L. Rudolph, "Inverse kinematics for a serial chain with joints under distance constraints." in *Robotics: Science and systems*, 2006.

[13] F. Marić, M. Giamou, I. Petrović, and J. Kelly, "Inverse kinematics as low-rank euclidean distance matrix completion," *arXiv preprint arXiv:2011.04850*, 2020.

[14] F. Marić, M. Giamou, A. W. Hall, S. Khoubyarian, I. Petrović, and J. Kelly, "Riemannian optimization for distance-geometric inverse kinematics," *IEEE Transactions on Robotics*, vol. 38, no. 3, pp. 1703–1722, 2021.

[15] M. Giamou, F. Marić, D. M. Rosen, V. Peretroukhin, N. Roy, I. Petrović, and J. Kelly, "Convex iteration for distance-geometric inverse kinematics," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1952–1959, 2022.

[16] F. Marić, M. Giamou, S. Khoubyarian, I. Petrović, and J. Kelly, "Inverse kinematics for serial kinematic chains via sum of squares optimization," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7101–7107.

[17] J. M. Porta, L. Ros, F. Thomas, and C. Torras, "A branch-and-prune solver for distance constraints," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 176–187, 2005.

[18] G. M. Crippen, T. F. Havel *et al.*, *Distance geometry and molecular conformation*. Research Studies Press Taunton, 1988, vol. 74.

[19] M. J. Sippl and H. A. Scheraga, "Cayley-menger coordinates." *Proceedings of the National Academy of Sciences*, vol. 83, no. 8, pp. 2283–2287, 1986.

[20] Y. Chen, Y. Cai, J. Xu, Z. Ren, G. Shi, and H. Choset, "Propagative distance optimization for constrained inverse kinematics," *arXiv preprint arXiv:2406.11572*, 2024.

[21] T. Weisser, J. B. Lasserre, and K.-C. Toh, "Sparse-bsos: a bounded degree sos hierarchy for large scale polynomial optimization with sparsity," *Mathematical Programming Computation*, vol. 10, pp. 1–32, 2018.

[22] J. J. Craig, *Introduction to robotics*. Pearson Educacion, 2006.

[23] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *International workshop on artificial neural networks*. Springer, 1995, pp. 195–201.

[24] J. E. Dennis, Jr and J. J. Moré, "Quasi-newton methods, motivation and theory," *SIAM review*, vol. 19, no. 1, pp. 46–89, 1977.

[25] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *2015 international conference on advanced robotics (ICAR)*. IEEE, 2015, pp. 510–517.

[26] I. A. Sucan and S. Chitta. "moveit". [Online]. Available: moveit.ros.org