RangedIK: An Optimization-based Robot Motion Generation Method for Ranged-Goal Tasks

Yeping Wang[1], Pragathi Praveena[1], Daniel Rakita[2] and Michael Gleicher[1]

***Abstract*— Generating feasible robot motions in real-time requires achieving multiple tasks (*i.e.*, kinematic requirements) simultaneously. These tasks can have a specific goal, a range of equally valid goals, or a range of acceptable goals with a preference toward a specific goal. To satisfy multiple and potentially competing tasks simultaneously, it is important to exploit the flexibility afforded by tasks with a range of goals. In this paper, we propose a real-time motion generation method that accommodates all three categories of tasks within a single, unified framework and leverages the flexibility of tasks with a range of goals to accommodate other tasks. Our method incorporates tasks in a weighted-sum multiple-objective optimization structure and uses barrier methods with novel loss functions to encode the valid range of a task. We demonstrate the effectiveness of our method through a simulation experiment that compares it to state-of-the-art alternative approaches, and by demonstrating it on a physical camera-in-hand robot that shows that our method enables the robot to achieve smooth and feasible camera motions.**

I. INTRODUCTION

In real-time robotics applications, the robot needs to calculate how to move at each update to satisfy multiple kinematic requirements simultaneously. As exemplified in Figure 1, a writing application may have several kinematic requirements, which the robot must fulfill to generate accurate and feasible motions. Following Nakamura et al. [1], we consider the kinematic requirements that the robot must fulfill as *tasks*. These tasks can be classified into three categories according to their flexibility: (1) A task can have a *specific* goal with limited flexibility, which may cause the robot to lose the capability to achieve other tasks when attempting to accomplish it; (2) A task can have a *range* of equally valid goals, which provides broad flexibility for a robot to accommodate other tasks; and (3) A task can have a *range* of acceptable goals while also showing preference for a *specific* goal, which offers flexibility when needed for more critical tasks while still targeting a specific goal when possible. We refer to tasks in the last two categories as *ranged-goal tasks*, in contrast to the *specific-goal tasks* in the first category.

In this paper, we introduce a real-time robot motion synthesis method that is able to accommodate specific-goal tasks, ranged-goal tasks with equally valid goals, and ranged-goal tasks with preferred goals within a single, unified framework. We address the real-time multiple-task motion generation problem through a generalized Inverse Kinematics solver, called *RangedIK*, which incorporates a set of specific-goal or ranged-goal tasks in a weighted-sum non-linear optimization structure. *RangedIK* supports flexible ranged-goal tasks in joint space or Cartesian space by utilizing barrier methods with novel loss functions to encode the valid range of a task in optimization. The multiple-objective optimization structure enables *RangedIK* to leverage the flexibility in ranged-goal tasks to improve the accuracy, smoothness, and feasibility of robot motions.

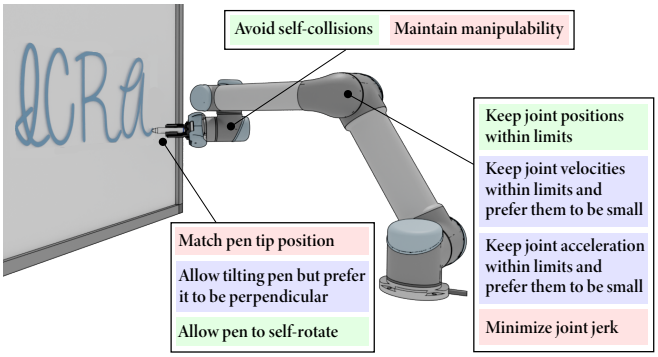

Fig. 1. Multiple tasks need to be achieved to generate accurate, smooth, and feasible robot motions in real-time for the whiteboard writing application. These tasks can be classified into three categories according to their flexibility: tasks that have a specific goal (red), tasks that have a range of equally valid goals (green), and tasks that have a range of acceptable goals with a preference toward a specific goal (blue). In this paper, we present a real-time motion synthesis method that can accommodate these three categories of tasks within a single, unified framework.

This paper builds on prior work [2], [3], which provides a per-instant pose optimization method known as *RelaxedIK* that optimizes single poses to achieve certain accuracy objectives without sacrificing motion feasibility. This paper extends the *RelaxedIK* method by leveraging the flexibility offered by ranged-goal tasks. The contributions of this paper are threefold: (1) a method to incorporate ranged-goal tasks in a multiple-objective optimization-based motion generation structure using novel parametric loss functions (§IV-A and §IV-B); (2) a set of specific-goal or ranged-goal task functions to generate accurate, smooth, and feasible robot motions (§IV-C); and (3) empirical evidence that ranged-goal tasks can create flexibility for improved accomplishment of other tasks (§V). We provide an open-source implementation of our proposed method[1].

To assess the efficacy of our method, we compare *RangedIK* to *RelaxedIK* [2], [3] and *Trac-IK* [4] on appli-

[1]Yeping Wang, Pragathi Praveena, and Michael Gleicher are with the Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI 53706, USA
[yeping|pragathi|gleicher]@cs.wisc.edu

[2]Daniel Rakita is with the Department of Computer Science, Yale University, New Haven, CT 06520, USA daniel.rakita@yale.edu

This work was supported by a University of Wisconsin Vilas Associates Award and National Science Foundation award 1830242.

[1]https://github.com/uwgraphics/relaxed_ik_core/tree/ranged-ik

cations with Cartesian tolerances, which create ranged-goal tasks to maintain end-effector poses within the tolerances (§V). Our results suggest that *RangedIK* generates more accurate, smooth and feasible motions than *RelaxedIK*, which does not utilize the flexibility offered by ranged-goal tasks. Our evaluation also shows that both *RangedIK* and *Trac-IK* generate valid motions within the specified tolerance, but our approach produces smoother and more feasible motions than *Trac-IK*. To showcase the generality of our method, we also demonstrate that our method enables a camera-in-hand robot to generate stable videos (§VI). Finally, we conclude this paper with a discussion of the limitations and implications of this work (§VII).

II. RELATED WORK

Our work builds upon prior works in semi-constrained motion planning, inverse kinematics, and barrier methods.

A. Semi-Constrained Motion Planning

Motion planning algorithms are widely used to enable end-effector path following. Semi-constrained motion planners have been presented to generate robot motions in applications that have Cartesian tolerances, *i.e.*, where accurate end-effector movements in all six degrees of freedom are not required. *Descartes* [5], [6] is an open-source search-based semi-constrained motion planner released by the ROS-Industrial community. Recently, Malhan et al. [7] presented an iterative graph construction method to find trajectories that satisfy constraints in Cartesian or joint space. Besides the graph-based methods mentioned above, sampling-based methods have also been used in semi-constrained motion planners [8], [9].

While semi-constrained motion planners can effectively generate motions for path following, they are not appropriate in time-sensitive, real-time scenarios such as teleoperation. In this work, we use single pose optimization for real-time motion generation. Prior works have shown that single pose optimization is effective for generating motions in real-time scenarios such as teleoperation [10] and camera control [11].

B. Inverse Kinematics

Semi-constrained motion planners often call an Inverse Kinematics (IK) solver *repeatedly* to get the joint positions that produce desired end-effector poses. Some emerging IK solvers can handle Cartesian tolerances and only need to be called once by the semi-constrained motion planners. *Trac-IK* [4] handles Cartesian tolerances by redefining Cartesian errors. Such an approach enables Trac-IK to be incorporated with a motion planner [12], but results in choppy motions when generating motion in real-time (more explanations are in §V-B). In this paper, we present a method that generates smooth motions not only for applications with Cartesian tolerances, but also other ranged-goal tasks.

In our work, we use the term *"task"* from task-priority IK [1], [13], [14], [15], which utilizes joint redundancy to handle a stack of tasks (kinematic requirements). The task priority framework has been extended to incorporate inequality tasks [16], [17]. Task-priority IK solvers can handle multiple tasks simultaneously by projecting a lower-priority task into the null-space of a higher-priority task, but this approach requires the robot to have sufficient kinematic redundancy for the null-space projections. Our method, on the other hand, can manage multiple and potentially competing tasks without the need for such redundancy requirements.

Task-priority IK approaches utilize a strict task hierarchy, which can sometimes be too conservative [18] or challenging to establish [19]. Meanwhile, non-strict task hierarchies are more flexible and are usually handled by weighted-sum strategies in optimization. Rakita et al. [2] demonstrated the benefits of using multiple-objective optimization to combine motion accuracy tasks with feasibility tasks such as self-collision avoidance. Building on this prior work, our approach leverages the flexibility of ranged-goal tasks to generate accurate, smooth, and feasible motions.

C. Barrier Methods

In non-linear optimization, barrier methods convert an inequality constraint to a positively-weighted "barrier" in the loss function to prevent solutions from leaving feasible regions [20]. Barrier functions have been widely used in robotics for real-time optimization-based controllers [21], [22], model predictive controllers [23], [24] and motion planners [25], [26]. In this work, we apply barrier functions to incorporate ranged-goal tasks in a multiple-objective optimization-based IK structure.

A strict definition of barrier functions requires them to go to infinity when a solution is close to an inequality constraint boundary. However, limitations of such "strict" barrier functions have been noted in prior work [23], [24]; strict barrier functions are only defined over the feasible space and their derivatives go to infinity as one approaches the constraint boundary. Therefore, Feller and Ebenbauer [24] proposed a relaxed logarithmic barrier function that has a suitable penalty term outside of the feasible space to overcome the downsides of strict barrier functions. While this function was designed to possess desired properties for model predictive control, in this paper, we design and utilize relaxed barrier functions to restrict kinematic task values in valid ranges for real-time motion generation.

III. TECHNICAL OVERVIEW

In this section, we provide notation for our problem and an overview of the optimization structure in our method.

A. Problem Formulation

Consider an n degree of freedom robot whose configuration is denoted by $\mathbf{q} \in \mathbb{R}^n$. A *task* (kinematic requirement) is described using $\chi(\mathbf{q}) \in \mathbb{R}$. The tasks can be classified in three categories according to the type of their goals.

A *Specific-Goal Task* has a single valid goal, *e.g.*, a pose matching task that requires a robot's end-effector to match a given goal pose. For such tasks, the value of the task function $\chi(\mathbf{q})$ should match the goal value $g(t)$ at time t:

$$\chi(\mathbf{q}) = g(t) \tag{1}$$

A *Ranged-Goal Task with Equally Valid Goals* allows for multiple valid goals within an interval. For instance, all joint positions are equally valid as long as they are within the joint limits of a robot. For such tasks, the value of the task function should fall within an interval defined by the lower bound $l(t)$ and the upper bound $u(t)$:

$$l(t) \leq \chi(\mathbf{q}) \leq u(t) \tag{2}$$

A *Ranged-Goal Task With a Preferred Goal* has a range of acceptable goals while showing a preference for a specific goal. For instance, the joint acceleration of a robot should be within its motor's limits, but we would prefer the acceleration to be as small as possible to minimize energy usage. This type of task is defined by an acceptable interval $[l(t), u(t)]$ and a preferred goal $g(t)$.

$$\begin{cases} \chi(\mathbf{q}) \approx g(t) \\ l(t) \leq \chi(\mathbf{q}) \leq u(t) \end{cases} \tag{3}$$

The goal of our work is to compute a joint configuration $\mathbf{q}$ to achieve a set of tasks $\{\chi_1(\mathbf{q}), \chi_2(\mathbf{q}), ..., \chi_m(\mathbf{q})\}$ as best as possible even when competing tasks arise, such as moving a robot to a goal pose while also minimizing energy usage. A task can be defined in joint space (*e.g.*, minimizing joint acceleration), Cartesian space (*e.g.*, matching end-effector poses), or a task-relevant space (*e.g.*, keeping an object in view for a camera-in-hand robot). In real-time applications, future goals $g(t+\delta)$ or goal ranges $[l(t+\delta), u(t+\delta)]$ are unknown at time t, for any $\delta > 0$.

B. Non-linear Optimization

We formulate the problem posed above as a constrained non-linear optimization problem:

$$\mathbf{q}^* = \arg\min_{\mathbf{q}} F(\boldsymbol{\chi}(\mathbf{q})) \quad s.t.\ l_i \leq q_i \leq u_i,\ \forall i \tag{4}$$

where l_i and u_i are the lower and upper bounds of the i-th robot joint, and $\boldsymbol{\chi}$ is a set of tasks to be achieved. $F(\boldsymbol{\chi}) \in \mathbb{R}$ is the weighted sum of the loss function for each task:

$$F(\boldsymbol{\chi}) = \sum_{j=1}^{J} w_j f_j(\chi_j(\mathbf{q})) \tag{5}$$

Here, $J \in \mathbb{Z}^+$ is the total number of tasks to be achieved and $w_j \in \mathbb{R}$ is the weight value for the j-th task. $\chi(\mathbf{q}) \in \mathbb{R}$ is a task function that has a specific goal value or a goal range. $f(\chi) \in \mathbb{R}$ is a parametric loss function that calculates the error between a task value χ and the specific goal value or the goal range. We will describe the parametric loss functions and the task functions that we utilize in §IV.

IV. TECHNICAL DETAILS

In this section, we provide details on how we incorporate ranged-goal tasks in a multiple-objective optimization structure. We first introduce three basic loss functions that are used to normalize or regularize task values. Later, we use the basic loss functions as building blocks to design parametric loss functions $f(\chi)$ for specific or ranged-goal tasks. Finally, we detail task functions χ that are used to generate accurate, smooth and feasible robot motions.

A. Basic Loss Functions

1) Gaussian: To combine multiple tasks, it is important to normalize each task such that their values are over a uniform range [2]. One common normalization method is the negative Gaussian function:

$$f(\chi, g) = -e^{-(\chi-g)^2/2c^2} \tag{6}$$

where g is a goal value and c is the standard deviation, which determines the spread of the Gaussian (Figure 2-A).

2) Wall: Another normalization method involves uniformly scaling values based on the lower bound l and upper bound u: $\chi' = (2\chi - l - u)/(u - l)$. After scaling, χ' is within the interval $[-1, 1]$. We present a continuous function to impose a large penalty when χ' is close to or outside the boundaries of the interval and a nearly zero penalty when χ' is within the interval:

$$f(\chi, l, u) = a_1 \left(1 - e^{-\chi'^n/b^n}\right) \tag{7}$$

As shown in Figure 2-B, the function has "walls" at the boundaries, hence it is named the *Wall* function. In Equation 7, $a_1 \in \mathbb{R}$ determines the wall "height", $b \in \mathbb{R}$ determines the wall "locations"[2] and $n \in 2\mathbb{Z}^+$ determines the steepness of the walls.

3) Polynomial: Neither the Gaussian nor the Wall function provides sufficient derivatives when solutions are far away from their goal or goal range. The non-sufficient derivatives make optimization sensitive to the initial guess, causing it stuck before reaching the optimal point. On the other hand, polynomials provide sufficient gradients everywhere that point towards a goal g:

$$f(\chi, g) = a_2(\chi - g)^m \tag{8}$$

where $m \in 2\mathbb{Z}^+$ is the degree of the polynomial and a_2 determines the severity of the penalty when a value is away from the optimal point (Figure 2-C).

B. Parametric Loss Functions

Below, we use the basic loss functions in §IV-A as building blocks to design a loss function for each task category. All of these parametric loss functions are continuous and smooth.

1) Specific-goal tasks: To enable task values to match a specific goal g, Rakita et al. [2], [3] combine the Gaussian and polynomial functions to design the *Groove* loss function:

$$f_g(\chi, g, \Omega) = -e^{-(\chi-g)^2/2c^2} + a_2(\chi - g)^m \tag{9}$$

Here, the scalar values c, a_2, and m form the set of parameters Ω. As shown in Figure 2-D, the *Groove* function has a narrow "groove" around the goal to steer values towards the goal and a more gradual descent away from the goal for better integration with other objectives.

[2]We computed b by solving $1 - \exp(-1/b^n) = 0.95$ to impose 0.95 of the maximum penalty at boundaries.

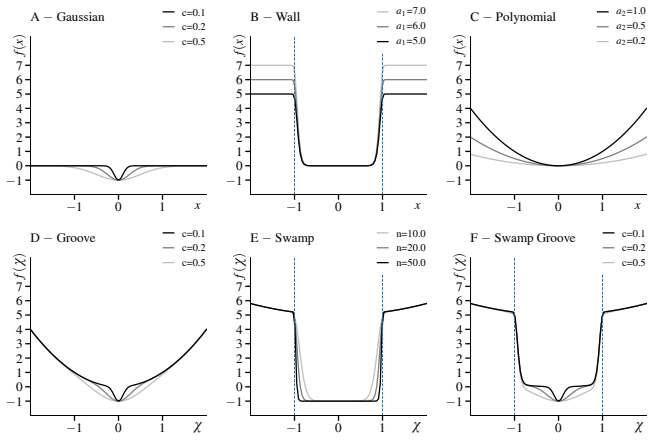

Fig. 2. We use the basic loss functions (A, B, C) as building blocks to design parametric loss functions for specific-goal tasks (D), ranged-goal tasks with equally valid goals (E), and ranged-goal tasks with a preferred goal (F).

2) Ranged-goal tasks with equally valid goals: Tasks with a continuous range $[l, u]$ of equally valid goals require a loss function that imposes nearly zero penalties if task values are within the range and large penalties if task values are close to or outside the boundaries. Moreover, the function should have sufficient gradients outside the boundaries. To satisfy these requirements, we present the *Swamp* loss function, which is a Wall function surrounded by a polynomial:

$$f_r(\chi, l, u, \Omega) = (a_1 + a_2 \chi'^m) \left(1 - e^{-\chi'^n / b^n}\right) - 1 \quad (10)$$

Here, the scalar values a_1, a_2, m, n, and b form the set of parameters Ω. χ' is the scaled task value described in §IV-A. As shown in Figure 2-E, the *Swamp* function has a flat "swamp" region to ensure that any solution within the range is equally good. There are steep "walls" near the boundaries to restrict solutions within the range. Outside of the walls, the gradual "funnel" provides gradients to drive solutions towards the swamp region.

3) Ranged-goal tasks with a preferred goal: We combine the Gaussian, Wall, and polynomial functions to design the *Swamp Groove* loss function that meets the requirements of tasks with acceptable goals within a range $[l, u]$ and a preferred goal g:

$$\begin{aligned} f_{rg}(\chi, l, u, g, \Omega) = &- e^{-(\chi-g)^2/2c^2} + a_2(\chi - g)^m \\ &+ a_1 \left(1 - e^{-\chi'^n / b^n}\right) \end{aligned} \quad (11)$$

Here, the scalar values c, a_1, a_2, m, n, and b form the set of parameters Ω. As shown in Figure 2-F, the *Swamp Groove* function has a "groove" shape near the preferred goal, steep "walls" at the boundaries, and a gradual "funnel" shape outside the walls. We note that the *Swamp Groove* function does not require the preferred goal g to be at the center of the range $[l, u]$ but g must be within the range.

C. Task Functions

In this section, we describe the mathematical details of task functions used in our work to generate accurate, smooth, and feasible motions. Besides these task functions, additional functions can be introduced for more specialized tasks. Task functions χ are composed with parametric loss functions $f(\chi)$ described in §IV-B and then combined using Equation 5 to form the objective function $F(\boldsymbol{\chi})$. The parameters Ω of the parametric loss function can be found in our open-source implementation. We estimated the parameters based on visualizations (*e.g.*, Figure 2) and fine-tuned them according to the robot behaviors. The following task functions were adapted from the objective terms in prior work [2].

1) End-effector Pose Matching: $\chi_P(\mathbf{q}, t, i)$ and $\chi_R(\mathbf{q}, t, i)$ indicate the end-effector's position and rotation error in the i-th DoF. We describe χ_P and χ_R with respect to the goal end-effector pose $[\mathbf{p}(t), \mathbf{R}(t)]$.

$$\chi_P(\mathbf{q}, t, i) = \left\{\mathbf{R}^{-1}(t)\left[\boldsymbol{\Psi}_p(\mathbf{q}) - \mathbf{p}(t)\right]\right\}_i \quad (12)$$

$$\chi_R(\mathbf{q}, t, i) = \left\{\mathbf{S}\left[\mathbf{R}^{-1}(t)\boldsymbol{\Psi}_R(\mathbf{q})\right]\right\}_i \quad (13)$$

where $\boldsymbol{\Psi}(\mathbf{q})$ is the robot's forward kinematics function. $\boldsymbol{\Psi}_p$ and $\boldsymbol{\Psi}_R$ denote the position and orientation of the robot's end-effector. $\mathbf{S}$ denotes a conversion from a rotation matrix to a vector (scaled-axis) with the direction of the rotation axis and whose norm is the rotation angle. $\{\cdot\}_i$ denotes the i-th element of a vector. χ_P and χ_R can be specific-goal tasks if exact pose matching is desired, but for applications with Cartesian tolerances, *e.g.*, the benchmark applications we evaluate in §V, χ_P or χ_R can be ranged-goal tasks.

2) Smooth Motion Generation: We consider the velocity v, acceleration a, and jerk j of each robot joint separately for the smooth motion generation task. We set limits for each joint as follows (i refers to the i-th joint of the robot):

$$\chi_v(\mathbf{q}, t, i) = \dot{\mathbf{q}}_i;\ \chi_a(\mathbf{q}, t, i) = \ddot{\mathbf{q}}_i;\ \chi_j(\mathbf{q}, t, i) = \dddot{\mathbf{q}}_i; \quad (14)$$

For joint velocities and accelerations, we use ranged-goal tasks with preferred goals. Joint velocities or accelerations are *acceptable* if they are within the motor limits and their *preferred* values are zeros. Joint jerks are treated as specific-goal tasks whose goal values are zeros.

3) Self-Collision Avoidance: We adapt the collision avoidance method from [27] for self-collision avoidance. Each robot link $\mathbf{l}_i$ is represented as a convex shape (*e.g.*, a capsule). A task function that represents the shortest straight-line distance between the i-th and j-th link can be written as:

$$\chi_c(\mathbf{q}, i, j) = \text{dist}(\mathbf{l}_i(\mathbf{q}), \mathbf{l}_j(\mathbf{q})) \quad (15)$$

where dist() is the shortest straight-line distance between two convex shapes and is computed using Support Mapping [28]. To consider self-collisions between all pairs of non-adjacent robot links, a group of ranged-goal tasks are added to the objective function (Equation 5): $\sum_{i=1}^{N-2}\sum_{j=i+2}^{N} f_r(\chi_c(\mathbf{q}, i, j), 0.02, \infty, \Omega)$, where N is the number of robot links. We set 0.02 m as the minimum allowable distance between two non-adjacent robot links.

4) Kinematic Singularity Avoidance: We use the Yoshikawa manipulability measure [29] to evaluate how close a configuration $\mathbf{q}$ is to a singularity. The manipulability task is defined as $\chi_m(\mathbf{q}) = \sqrt{\det\left(\mathbf{J}(\mathbf{q})\mathbf{J}^T(\mathbf{q})\right)}$, where $\mathbf{J}$ is the Jacobian matrix that maps joint speeds to end-effector

velocities. $\chi_m(\mathbf{q})$ is a specific-goal task and is injected into the loss function $f_g(\chi, 1, \Omega)$ to maximize manipulability.

V. Evaluation

In this section, we compare *RangedIK* with two alternative approaches, *RelaxedIK* and *TracIK*, to generate motions for applications that allow some tolerances in end-effector poses. These Cartesian tolerances create ranged-goal tasks that maintain end-effector pose within the tolerances.

A. Implementation Details

Our prototype implementation was based on the open-source *CollisionIK* library[3]. Our prototype uses the Proximal Averaged Newton-type Method (PANOC) [30] as the optimization solver. All evaluations were performed on an Intel Core i7-11800H 2.30 GHz CPU with 16 GB RAM.

B. Comparisons

RelaxedIK [2] is an optimization-based Inverse Kinematics solver that generates feasible motions given multiple objective terms, such as end-effector pose matching, smooth joint motion, and self-collision avoidance. However, all of the objectives have a single goal. We want to show that by considering ranged-goal tasks, *RangedIK* will generate more accurate, smooth, and feasible motions than *RelaxedIK*.

Another alternative approach is to generate motions with a widely used Inverse Kinematics solver, *Trac-IK* [4], which biases the search around the joint space of a given seed value. We provide the seed value as the configuration from the previous update. Trac-IK considers Cartesian pose tolerances by mapping pose errors to a discontinuous function:

$$p^i_{err} = \begin{cases} 0, & \text{if} \quad l \leq p^i_{err} \leq u \\ p^i_{err}, & \text{otherwise} \end{cases} \tag{16}$$

Here, p^i_{err} is the pose error in the i-th degree of freedom. The discontinuous function makes the robot stop moving when the end-effector pose error is within the interval and leads to a sudden, large movement once the pose error is outside of the range. Consequently, motions generated by Trac-IK can have large joint accelerations and jerks.

C. Experimental procedure

We compared our method to alternative approaches on the four benchmark applications described in Section V-D. The randomly generated path for each benchmark was discretized to 2,000 end-effector goal poses at 30 Hz. The goal poses, along with the Cartesian tolerances, were provided to *RangedIK* and *Trac-IK*, whereas *RelaxedIK* received only goal poses due to its inability to accommodate Cartesian tolerances. We repeated all benchmarks 10 times using two simulated robots: a 6-DoF Universal Robots UR5 and a 7-DoF Rethink Robotics Sawyer. Our experiment consisted of 480,000 discrete solutions and involved 2 robots, 3 methods, 4 benchmark applications, and 10 randomly generated configurations for each application.

[3]https://github.com/uwgraphics/relaxed_ik_core/tree/collision-ik

TABLE I
Cartesian Tolerances

Benchmark	Tolerances					
	x/m	y/m	z/m	rx/rad	ry/rad	rz/rad
Writing	0	0	0	$\pm\frac{\pi}{6}$	$\pm\frac{\pi}{6}$	$\pm\infty$
Spraying	±0.05	±0.05	0	0	0	0
Wiping	0	0	0	0	0	$\pm\infty$
Filling	±0.05	0	±0.05	0	$\pm\infty$	0

D. Benchmark

We developed four benchmark applications with Cartesian tolerances (Table I) to compare our method against alternative approaches. The first three benchmarks involve manipulation tasks on a whiteboard positioned in front of the robot. The facing angle of the whiteboard is uniformly sampled from $[0, \pi/2]$, with 0 and $\pi/2$ representing a vertical and horizontal whiteboard, respectively.

1) Writing: The robot uses a marker as the end-effector to draw curves on the whiteboard. The writing trajectory includes 5 randomly generated cubic Bezier curves. The application requires the marker tip position to be accurate but allows the marker to tilt (rotational tolerances about the marker's x and y axes) or freely rotate about the marker's principal z axis.

2) Spraying: The robot sprays cleaner on the whiteboard. The spraying positions are uniformly sampled to cover the whiteboard. Some position tolerances parallel to the whiteboard plane (the xy plane) are allowed because the cleaner will be wiped by an eraser in the next application.

3) Wiping: The robot wipes the whiteboard with a round eraser, moving along a predefined, lawnmower path to cover the entire whiteboard. The round eraser's rotational symmetry allows the robot to rotate freely about the eraser's principal axis (rotational tolerances about the z axis).

4) Filling Water: The robot fills a cup with the water from a faucet and places the cup on a tabletop. The positions of the initial empty cup, the faucet, and the final full cup are uniformly sampled from three $0.2\text{m}\times0.2\text{m}\times0.2\text{m}$ domains within the robot's workspace. A cup can be filled as long as the water flow is within the cup, allowing for some horizontal position tolerances in the xz plane. The cup's rotational symmetry allows the robot to rotate freely about the cup's principal y axis.

E. Metrics

We used 7 metrics to measure the *accuracy*, *smoothness*, *manipulability* and *validity* of robot motions. Motion *accuracy* was measured using mean position error (m) and mean rotation error (rad), which were measured only in the degrees of freedom without tolerances. We used mean joint velocity (rad/s), mean joint acceleration (rad/s^2), and mean joint jerk (rad/s^3) to assess motion *smoothness*. Motion *manipulability* was measured by mean Yoshikawa manipulability [29], where a higher value indicates better manipulability. Motion *validity* was measured by the total number of solutions that exceeded the tolerances. We also measured mean movements in joint space, where larger joint movements result in increased wear and tear on the robot.

TABLE II

RESULTS FROM THE EXPERIMENT

	Method	Mean Pos. Error (m)	Mean Rot. Error (rad)	Mean Joint Vel. (rad/s)	Mean Joint Acc. (rad/s^2)	Mean Joint Jerk (rad/s^3)	Mean Manipulability	# Exceed Tolerances	Mean Joint Movement (rad)
UR5	*RangedIK*	0.0021±0.002	0.005±0.003	**0.042±0.02**	**0.0269±0.02**	**0.224±0.2**	**0.0544±0.01**	0	**16.747±8.132**
	RelaxedIK	0.0023±0.002	0.007±0.005	0.050±0.02	0.0299±0.02	0.336±0.3	0.0533±0.01	0	20.097±8.796
	Trac-IK	**1.8e-6±1.6e-6**	**1.5e-6±1.6e-6**	0.061±0.04	1.9333±2.48	115.536±149	0.0487±0.01	0	24.195±15.918
Sawyer	*RangedIK*	0.0014±0.001	0.003±0.002	**0.034±0.02**	**0.0265±0.02**	**0.290±0.3**	**0.1539±0.04**	0	**15.897±7.934**
	RelaxedIK	0.0017±0.001	0.006±0.004	0.041±0.02	0.0280±0.03	0.328±0.4	0.1535±0.04	0	19.247±9.783
	Trac-IK	**1.3e-6±1.1e-6**	**2.0e-6±1.6e-6**	0.047±0.03	1.4882±1.79	88.930±107	0.1440±0.05	0	22.051±14.029

The range values are standard deviations. The best value among the three methods for each measure is highlighted in bold.

F. Results

As shown in Table II, both the non-redundant UR5 and the redundant Saywer robot benefitted from *RangedIK*. Compared to *RelaxedIK*, our method generated more accurate and smoother robot motions with higher manipulability. Specifically, *RangedIK* could reach better Cartesian accuracy with less joint movement compared to *RelaxedIK*, and all solutions generated by *RangedIK* were within the Cartesian tolerances. These results indicate that our method *RangedIK* could leverage the flexibility offered by ranged-goal tasks to improve the quality of generated robot motions.

Our results also show that *RangedIK* could satisfy multiple kinematics requirements simultaneously. Compared to *Trac-IK*, which focuses on generating accurate solutions within joint limits, *RangedIK* could generate smoother motions with larger manipulability. In particular, due to the discontinuity in Equation 16, *Trac-IK* generated choppy motions with large accelerations and jerks.

VI. DEMONSTRATION

The experiment in §V shows that our method can generate feasible motions for applications with Cartesian tolerances. In addition, we demonstrate the effectiveness our method on a camera-in-hand robot that tracks a user's hand to capture clear hand gestures or movements.

Prior works [11], [31] have presented a series of tasks to generate feasible camera motions (Figure 3-A). One of these tasks is the "look-at task" which steers the robot to look at the user's hand. While prior works have set a specific goal for this task (Figure 3-B left), our method allows us to set a preferred goal and a range of acceptable goals (Figure 3-B right). By utilizing the flexibility of the ranged-goal task, our method generates smoother motions to prevent shaky videos. The demonstrations are shown in the supplementary video [4].

VII. DISCUSSION

In this work, we presented *RangedIK*, a real-time motion generation method that accommodates specific and ranged-goal tasks and leverages the flexibility offered by ranged-goal tasks to generate higher-quality robot motion. Below, we discuss the limitations and implications of this work.

[4]https://github.com/uwgraphics/relaxed_ik_core/tree/ranged-ik#supplementary-video

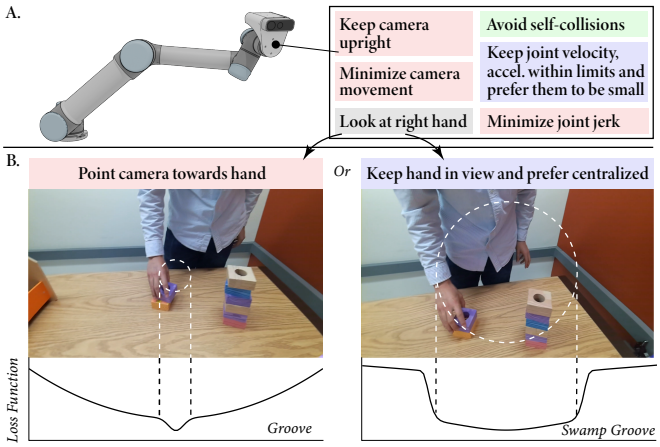

Fig. 3. **A.** We apply our method on a camera-in-hand robot, which requires a set of tasks to enable feasible camera motions. Specific-goal tasks, ranged-goal tasks with equally valid goals, and ranged-goal tasks with a prefer goal are in red, green, and blue, respectively. **B.** The task to track a user's right hand can be either a specific-goal task (left) or a ranged-goal task with a prefer goal (right). We observed that our method generated smoother videos with flexibility in ranged-goal tasks.

A. Limitations

Our work has some limitations that highlight directions for future research. While our method can generate real-time feasible motions, it can lack foresight because the tasks are defined to find a feasible solution for *now*. Extension to this work could investigate ways to incorporate *future* feasibility tasks in our optimization-based structure. Also, our method builds on a non-linear optimization formulation that may get stuck in local minima. Future work should explore methods (*e.g.*, warm-start strategies [32]) to enable our method to escape local minima. Finally, *RangedIK* treats all tasks as optimization objectives, which may require scenario-specific tuning of parameters and weights. Future work can extend our method to include automatic weight-tuning.

B. Implications

Our results demonstrate that *RangedIK* can utilize the flexibility in ranged-goal tasks to generate feasible motion on-the-fly. We believe that such capability makes our method applicable in complex real-life scenarios where many, and potentially competing, tasks need to be achieved simultaneously, such as teleoperation or active camera control. Furthermore, the parametric loss functions presented in our work could benefit other optimization-based methods, *e.g.*, offline motion planning or model predictive control.

REFERENCES

[1] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *The International Journal of Robotics Research*, vol. 6, no. 2, pp. 3–15, 1987.

[2] D. Rakita, B. Mutlu, and M. Gleicher, "Relaxedik: Real-time synthesis of accurate and feasible robot arm motion." in *Robotics: Science and Systems*. Pittsburgh, PA, 2018, pp. 26–30.

[3] ——, "An analysis of relaxedik: an optimization-based framework for generating accurate and feasible robot arm motions," *Autonomous Robots*, vol. 44, no. 7, pp. 1341–1358, 2020.

[4] P. Beeson and B. Ames, "Trac-ik: An open-source library for improved solving of generic inverse kinematics," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 928–935.

[5] ROS-I. (2015) Descartes—a ros-industrial project for performing path-planning on under-defined cartesian trajectories. [Online]. Available: http://wiki.ros.org/descartes

[6] J. De Maeyer, B. Moyaers, and E. Demeester, "Cartesian path planning for arc welding robots: Evaluation of the descartes algorithm," in *2017 22nd IEEE International conference on emerging technologies and factory automation (ETFA)*. IEEE, 2017, pp. 1–8.

[7] R. K. Malhan, S. Thakar, A. M. Kabir, P. Rajendran, P. M. Bhatt, and S. K. Gupta, "Generation of configuration space trajectories over semi-constrained cartesian paths for robotic manipulators," *IEEE Transactions on Automation Science and Engineering*, 2022.

[8] M. Cefalo, P. Ferrari, and G. Oriolo, "An opportunistic strategy for motion planning in the presence of soft task constraints," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6294–6301, 2020.

[9] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.

[10] D. Rakita, B. Mutlu, and M. Gleicher, "A motion retargeting method for effective mimicry-based teleoperation of robot arms," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, 2017, pp. 361–370.

[11] ——, "An autonomous dynamic camera method for effective remote teleoperation," in *2018 13th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2018, pp. 325–333.

[12] J. Gonçalves and P. Lima, "Grasp planning with incomplete knowledge about the object to be grasped," in *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2019, pp. 1–6.

[13] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano, "Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy," *The International Journal of Robotics Research*, vol. 10, no. 4, pp. 410–425, 1991.

[14] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 398–410, 1997.

[15] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar, "A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks," in *2009 International conference on advanced robotics*. IEEE, 2009, pp. 1–6.

[16] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 785–792, 2011.

[17] A. Escande, N. Mansard, and P.-B. Wieber, "Fast resolution of hierarchized inverse kinematics with inequality constraints," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 3733–3738.

[18] J. Salini, V. Padois, and P. Bidaud, "Synthesis of complex humanoid whole-body behavior: A focus on sequencing and tasks transitions," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1283–1290.

[19] M. Liu, Y. Tan, and V. Padois, "Generalized hierarchical control," *Autonomous Robots*, vol. 40, no. 1, pp. 17–31, 2016.

[20] A. Forsgren, P. E. Gill, and M. H. Wright, "Interior methods for nonlinear optimization," *SIAM review*, vol. 44, no. 4, pp. 525–597, 2002.

[21] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[22] Y. Chen, A. Singletary, and A. D. Ames, "Guaranteed obstacle avoidance for multi-robot operations with limited actuation: A control barrier function approach," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 127–132, 2020.

[23] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback mpc for torque-controlled legged robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4730–4737.

[24] C. Feller and C. Ebenbauer, "Relaxed logarithmic barrier function based model predictive control of linear systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 3, pp. 1223–1238, 2016.

[25] G. Yang, B. Vang, Z. Serlin, C. Belta, and R. Tron, "Sampling-based motion planning via control barrier functions," in *Proceedings of the 2019 3rd International Conference on Automation, Control and Robots*, 2019, pp. 22–29.

[26] M. Saveriano and D. Lee, "Learning barrier functions for constrained motion planning with dynamical systems," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 112–119.

[27] D. Rakita, H. Shi, B. Mutlu, and M. Gleicher, "Collisionik: A per-instant pose optimization method for generating robot motions with environment collision avoidance," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 9995–10 001.

[28] B. Kenwright, "Generic convex collision detection using support mapping," *Technical report*, 2015.

[29] T. Yoshikawa, "Manipulability of robotic mechanisms," *The international journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[30] L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos, "A simple and efficient algorithm for nonlinear model predictive control," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 1939–1944.

[31] D. Rakita, B. Mutlu, and M. Gleicher, "Remote telemanipulation with adapting viewpoints in visually complex environments," *Robotics: Science and Systems XV*, 2019.

[32] W. Merkt, V. Ivan, and S. Vijayakumar, "Leveraging precomputation with problem encoding for warm-starting trajectory optimization in complex environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 5877–5884.