RCIK: Real-Time Collision-Free Inverse Kinematics using a Collision-Cost Prediction Network

Mincheul Kang[1], Yoonki Cho[1], and Sung-Eui Yoon[2]

***Abstract*—In this paper, we present real-time collision-free inverse kinematics (RCIK) that accurately performs consecutively provided six-degrees-of-freedom commands in environments containing static and dynamic obstacles. Our method is based on an optimization-based IK approach to generate IK candidates with high feasibility for the command. While checking various constraints (e.g., collision and joint velocity limits), we select the best configuration among generated IK candidates through our objective function, considering the continuity of joints and collision avoidance with obstacles. To avoid dynamic obstacles efficiently, we propose a novel, collision-cost prediction network (CCPN) that estimates collision costs using an occupancy grid updated from sensor data in real-time. We evaluate our method in three dynamic problems using a real robot, the Fetch manipulator, and four static problems using three different configurations of robots. We show that the proposed method successfully performs the consecutively given commands in real-time, mainly thanks to the collision-cost prediction network, while avoiding dynamic and static obstacles. The results of tested problems are available in the accompanying video.**

***Index Terms*—Manipulation Planning, Kinematics**

I. INTRODUCTION

REMOTE control of a robotic manipulator has been used to perform tasks on behalf of humans at special-purpose sites such as nuclear power plants [1] and hospitals during telesurgery [2]. Recent research on the remote control of robots has been expanding to a diverse set of environments, including the home environment [3]. Due to this broadening diversity, robots are now expected to handle more difficult challenges when avoiding various and dynamic obstacles, while also accurately following human commands.

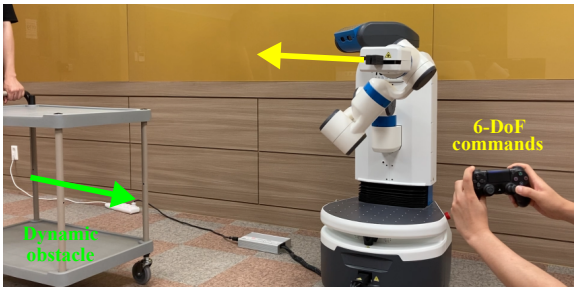
A robotic manipulator is capable of human-like movement given its many joints, but it has various constraints, e.g., avoiding collisions with obstacles and kinematic singularities, and maintaining the continuity of joint configurations. Among the various constraints, collision avoidance from obstacles incurs significant computational overhead and is one of the most important issues in many prior approaches [4], [5] to achieve real-time performance when executing human commands.

Manuscript received: September 2, 2021; accepted November 1, 2021.

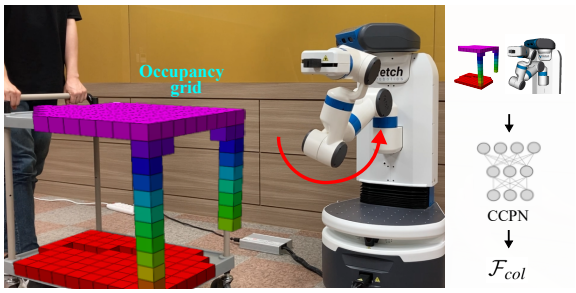
This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korea government (MSIT) under Grants 2021R1A4A1032582 and 2019R1A2C3002833. This letter was recommended for publication by Associate Editor R. Carloni and Editor L. Pallottino upon evaluation of the reviewers' comments. *(Corresponding author: Sung-Eui Yoon.)*

[1]Mincheul Kang and Yoonki Cho are with School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, South Korea mincheul.kang@kaist.ac.kr, yoonki@kaist.ac.kr

[2]Sung-Eui Yoon is with Faculty of School of Computing, Korea Advanced Institute of Science and Technology, Deajeon, South Korea sungeui@kaist.edu



(a) We send consecutive 6-DoF commands to a robot, and a moving cart obstructs the robot.



(b) The robot avoids the cart by computing F_{col} using our collision-cost prediction network (CCPN), while following consequtively given commmands.

Fig. 1. These figures show the progress of our method in a dynamic environment. (a) Our method performs consecutively given 6-DoF commands while avoiding a dynamic obstacle, a cart in this case. (b) To avoid the cart, we compute a collision cost, F_{col}, using our collision-cost prediction network (CCPN). The CCPN uses an occupancy grid updated through sensor data to reflect changing environments immediately. Finally, our method finds a desired joint configuration that allows the robot to follow the given command accurately while avoiding obstacles. Our algorithm conducts all of these operations within 30ms for each command, mainly due to CCPN.

Recently, TORM [6] reduces the computational overhead required for external static obstacles by applying the collision avoidance method using the signed distance field (SDF) used in CHOMP [7]. However, the SDF is hard to deal with dynamic obstacles in real-time due to its long construction time.

Main Contributions. In this work, we present real-time collision-free inverse kinematics (RCIK) using a novel, collision-cost prediction network (CCPN) to reduce the massive computational overhead associated with collision avoidance. Our method performs consecutively given six-Degrees-of-Freedom (DoF) commands in environments, including dynamic obstacles (Fig. 1). We generate IK candidates with high feasibility based on an optimization-based IK approach (Sec. IV-A). We then check whether generated IK candidates are satisfied with constraints, collision, joint velocity limits, and kinematic singularities. Among the IK candidates

satisfying the constraints, we select the best configuration through our objective function, consisting of collision avoidance and the smoothness of joints to support consecutively given commands (Sec. IV-B). For handling dynamic obstacles, our network estimates a collision cost by grasping environment information from an occupancy grid updated from sensor data in real-time (Sec. IV-C).

To evaluate our method, we test three dynamic problems using a real robot, the Fetch manipulator, and four static problems for three different configurations of robots (Sec. V-A). We observe that our method finds collision-free joint configurations in real-time in both static and dynamic environments, thanks to our deep neural network estimating collision costs quickly and accurately. We also analyze our method by comparing over the SDF (Sec. V-B).

II. RELATED WORK

In this section, we discuss prior studies in the areas of inverse kinematics and collision avoidance for manipulation planning.

A. Inverse Kinematics

Traditional inverse kinematics (IK) has been widely studied to determine a joint configuration for a target end-effector pose quickly and successfully [8]. For example, IKFast [9] and the work of Sinha et al. [10] analytically solve equations of a complex kinematics chain to reduce the computation time. Additionally, Trac-IK [11] improves the success rate of the Jacobian-based IK method by restarting from a random joint configuration and applying sequential quadratic programming instead of the Jacobian. Nevertheless, these methods encounter difficulty when following a given sequence of end-effector poses, called *path-wise IK*, because there is no consideration of the continuity on generated joint configurations and constraints (e.g., collisions, joint velocity limits, and kinematic singularities).

Considering the aforementioned constraints, RelaxedIK [4] solves the path-wise IK by means of weighted-sum non-linear optimization. Based on RelaxedIK, Stampede [12] synthesizes an optimal trajectory from a discrete graph built in consideration of several constraints. These methods use a neural network for self-collision avoidance to reduce the computational overhead, but do not consider external obstacles.

To handle external obstacles, Holladay et al. [13], [14] suggest two methods based on optimization-based and sampling-based motion planning. These works apply the discrete Fréchet distance to approximate the similarity with a given end-effector path in the Cartesian space. Moreover, TORM [6] improves speed and accuracy by integrating the Jacobian-based IK method and an optimization-based motion planning approach.

Recently, CollisionIK [5] deals with static and dynamic obstacles in real-time by computing the distance between robot links and obstacles using the convex hull approach. On the other hand, it does not take into account difficulties of a real environment, such as sensor noise.

We also propose a real-time approach to solve path-wise IK problems in an environment surrounding static and dynamic obstacles, but we overcome the difficulties of a real environment using an occupancy grid and a deep neural network. An occupancy grid is updated in real-time while eliminating sensor noise based on a probabilistic update rule [15], and our deep neural network quickly predicts collision costs by grasping the environment information from the occupancy grid.

B. Collision Avoidance for Manipulation Planning

A SDF has been used in many manipulation planning methods, especially in relation to optimization-based approaches [6], [7], because the SDF helps to avoid collision efficiently by computing a quantitative collision cost using the distance information from obstacles. On the other hand, it is difficult to use the SDF in dynamic environments due to its construction time. Although selective dynamic updating [16], [17] and parallel processing [18], [19] methods can reduce the construction time, constructing the SDF still incurs high computational overhead, especially when used to handle dynamic environments in real-time.

For a dynamic environment, several works have used simplified methods, such as approximating obstacles to spheres [20], using the closest distance from the obstacles [21]. Recently, Kew et al. [22] suggest a neural estimator for predicting the shortest distance. This estimator serves as a collision detector, improving the performance of a motion planner. On the other hand, this estimator can be used only within the learned objects because it is learned by changing only the positions of the objects.

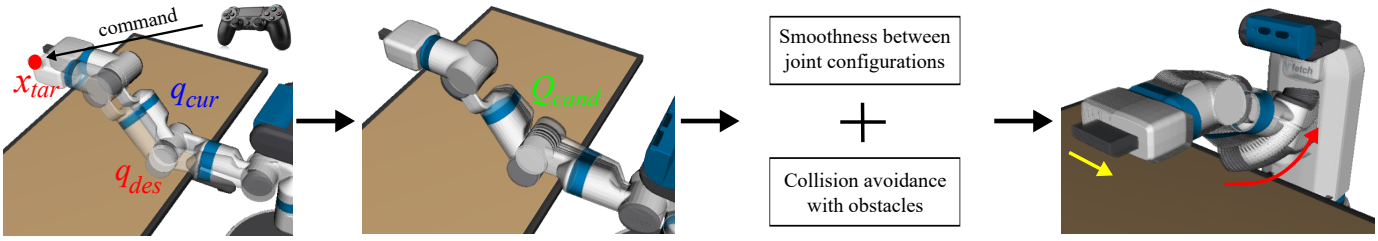
Unlike the work of Kew et al. [22], our deep neural network reflects changing environments immediately without any restrictions on objects by grasping environment information from an occupancy grid updated through sensor data in real-time [15]. Moreover, our network is not used as a collision detector and is used to select a desired joint configuration away from obstacles among generated joint configurations for a target end-effector pose in our sampling-based approach.

III. OVERVIEW

In this section, we introduce the problem we aim to solve, followed by giving motivations of our work.

A. Problem Definition

Our goal is to develop a collision-free IK solver in real-time for accurately performing consecutively given 6-DoF commands in environments with static and dynamic obstacles. The command is provided as a target end-effector pose, $x_{tar} \subset \mathbb{R}^6$, and we deal with a redundant manipulator that can have many joint configurations for one end-effector pose x_{tar}; a redundant manipulator generally has a DoF value that exceeds 6. Among many solutions, we find a desired joint configuration, $q_{des} \subset \mathbb{R}^D$, where D is the DoF of the robot, considering consecutively given 6-DoF commands and various constraints.



(a) 6-DoF command (b) IK candidates (c) Cost function (d) Execution results

Fig. 2. This shows an overview of our approach. (a) Our method synthesizes a desired joint configuration q_{des} that matches the target end-effector pose x_{tar} as a 6-DoF command and has the motion feasibility considering the current configuration q_{cur}. (b) First, we generate IK candidates Q_{cand} with high feasibility based on an optimization-based IK approach. (c) With checking constraints, such as collision and joint velocity limits, for Q_{cand}, we select the best one using our objective function for collision avoidance with obstacles and smoothness with previous joint configurations. (d) For 6-DoF commands given consecutively, denoted with the yellow arrow, our method sequentially synthesizes joint configurations that accurately realize the commands while avoiding collision and achieving smoothness, as indicated by the red arrow.

We now describe our assumption for a given command and the criteria of finding q_{des}. First, we assume that a command given to a robot from a user is considering the robot specifications, a given time, and an environment. To perform the command accurately, our method finds q_{des} that accurately matches x_{tar}. When receiving an invalid command, e.g., collision due to moving obstacles and out of possible moving range, we stop the robot's movement for safety and wait for a valid command.

Next, our problem undertakes collision avoidance in static and dynamic environments and supports the continuity of synthesized joint configurations for consecutively given commands. To avoid a collision, we find q_{des} far away from obstacles among many joint configurations for x_{tar}, since we aim to follow the command exactly.

Lastly, q_{des} must satisfy various constraints, and we call it having *motion feasibility* in this paper. The requirement of the *motion feasibility* include the following: the manipulator 1) must not collide with obstacles, 2) it does not violate the joint velocity limit in order to be able to move from the current configuration, q_{cur}, within the given time, t, and 3) it does not enter kinematic singularities, which occur when the Jacobian matrix loses the full rank.

In summary, we target to solve real-time inverse kinematics for consecutively given 6-DoF commands, while satisfying the requirement of motion feasibility in both static and dynamic environments.

B. Motivation

A real-time approach has to compute the desired solution during a given short time t, i.e., 30ms in our problem. Therefore, reducing the computational overhead is a critical issue for our problem. As explained in Sec. II-A, many works have proposed methods to reduce the computational overhead of obstacle avoidance, which takes up a lot of computation. Among them, TORM [6] uses the signed distance field (SDF) to reduce the computational overhead for collision avoidance. TORM improves the trajectory optimization performance by quickly synthesizing an initial trajectory using IK samples generated by the traditional IK method.

Inspired by TORM, our method generates IK candidates, $Q_{cand} = \{q^1_{cand}, q^2_{cand}, ...\}$, for x_{tar} based on an optimization-based IK, one of the traditional-IK methods, and then selects the best q_{des} that has the motion feasibility and minimizes our objective function (Fig. 2). To follow consecutive commands, our objective function considers collision avoidance of obstacles and the continuity of joint configurations.

Due to selecting one of generated IK candidates, increasing the possibility of having the motion feasibility of IK candidates is an important factor for improving the success rate of our approach. To obtain such IK candidates, especially satisfying the joint velocity limits, we utilize an optimization-based IK approach that has the property of obtaining a solution depending on the initial configuration [23], [24]. By maneuvering the initial configuration close to q_{cur}, we can generate IK candidates that have the potential to satisfy the joint velocity limits given the short time t.

The SDF can reduce the computational overhead of collision avoidance by calculating the distance information between obstacles in advance. The SDF also makes it possible to compute a quantitative cost, F_{col}, for collisions with obstacles. Despite these advantages, it is difficult to apply the SDF to dynamic environments due to its long construction time. To overcome this problem, we predict F_{col} using a deep neural network without constructing the SDF. Our network uses an occupancy grid updated from sensor data as the input to reflect changing environments in real-time. F_{col} is used to select an IK candidate far away from obstacles among generated IK candidates.

IV. Real-time Collision-free Inverse Kinematics

Fig. 2 shows the system flow. In this section, we explain how the proposed method generates IK candidates based on an optimization-based IK and how it assesses the motion feasibility. We then introduce the method that selects the best candidate from among all IK candidates using our objective functions, considering collision avoidance with obstacles and the continuity of joint configurations. We also propose a collision-cost prediction network (CCPN) for avoiding dynamic obstacles.

A. IK candidates

While a redundant manipulator has various joint configurations for a target end-effector pose x_{tar}, joint configurations

with the motion feasibility are limited. In particular, considering the given short time and velocity limits of joints, the possible moving range of joints from the current joint configuration q_{cur} is very narrow. Finding many IK candidates within possible moving range in a short given amount of time can increase the performance of our approach. To generate IK candidates Q_{cand} within the most likely joint bounds, we use an optimization-based IK approach and set its initial configuration, q_{init}^{opt-IK}, to randomly generated configurations using a Gaussian distribution (Fig. 2(b)); we use the Jacobian-based IK solver.

An optimization-based IK method starts with the initial configuration q_{init}^{opt-IK} and finds a joint configuration q for x_{tar} by iteratively minimizing the error with x_{tar} in a Cartesian space. By repeatedly minimizing the error, it finds q that is close to the initial configuration q_{init}^{opt-IK} [23]–[25]. Also, different initial configurations can generate diverse solutions for a redundant manipulator [24], [25].

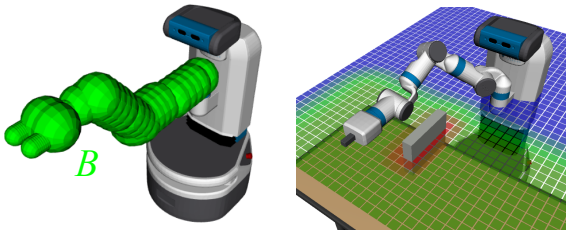
Inspired these properties, we create an initial configuration of optimization-based IK q_{init}^{opt-IK} by modifying q_{cur} using the multivariate Gaussian distribution, N:

$$q_{init}^{opt-IK} \sim N(q_{cur}, diag(\sigma_1^2, \sigma_2^2, ..., \sigma_D^2)), \tag{1}$$

where $diag(\cdot)$ is a diagonal, covariance matrix of size D and σ_i is the standard deviation for considering the properties of the i-th joint. $diag(\cdot)$ controls how closely IK candidates are generated from the current configuration q_{cur}. Specifically, σ_i of $diag(\cdot)$ is determined by $\mu \times v_i$, where v_i is the maximum velocity of the i-th joint and μ is a constant that serves to adjust the closeness of IK candidates from q_{cur}. When μ is too small, it is difficult to obtain diverse Q_{cand}, whereas it is difficult to obtain Q_{cand} satisfying the joint velocity limits when μ is too large; thus, we experimentally found $\mu = 0.1$ to strike the balance between diversity and generating many feasible Q_{cand}. In addition, we consider v because the maximum velocity of each joint $v = \{v_1, v_2, ..., v_D\}$ can differ.

There is no guarantee that Q_{cand} generated by the aforementioned process have the motion feasibility. We thus check the constraints for collision, joint velocity limits, and kinematic singularities. Note that we delay collision checking until after we calculate our cost function to reduce the computational overhead of collision detection; we use FCL [26], a commonly used collision checker. The joint velocity limit checks whether the manipulator can move from q_{cur} to the generated Q_{cand} during the given time t. In the case of kinematic singularities, we use the lower bound of the manipulability [27] obtained from random configurations, as is used by RelaxedIK [4]. In short, we check Q_{cand} where the manipulability value is greater than the lower bound.

When checking the constraints for Q_{cand}, there can be no feasible solution due to invalid command or insufficient Q_{cand} caused by the short time constraints and the randomness of our generation method. In those cases, we stop the robot's movement by following prior methods of handling dynamic obstacles [28], [29]. Since no algorithm guarantees success in the problem of dealing with unknown dynamic obstacles in real-time, the prior methods have decided to stop the robot's movement for robot safety [28], [29]. Furthermore, our problem deals with an unknown future command from a user. To increase the likelihood of success against unknown commands and obstacles in the future, we select a joint configuration away from obstacles among Q_{cand} through our objective function.



(a) Simplification of robot links. (b) Signed distance field for external obstacles.

Fig. 3. These figures show (a) the simplified robot links, spheres B, used for manipulation and (b) signed distance field for external obstacles, the table and gray box. The color of cells represents the distance from the closest obstacles; as the distance from a cell to the obstacle becomes smaller, its color gets closer to the red, and vice-versa to blue.

B. Objective Function

For one target end-effector pose x_{tar}, we can use all of Q_{cand} with the motion feasibility. However, to execute consecutive commands, it is necessary to consider the continuity between joint configurations and collision avoidance with obstacles. Because a collision can occur from an unknown future command and obstacles, we focus on selecting a joint configuration that is far away from obstacles among generated Q_{cand}. In this work, we select the configuration with the smallest value among Q_{cand} with the motion feasibility using the objective function, U, which numerically expresses the aforementioned properties.

We model the objective function U to have two cost terms for collision avoidance with obstacles, F_{col}, and smoothness between joint configurations, F_{smooth}:

$$U = F_{smooth} + w_{col} F_{col}, \tag{2}$$

where w_{col} is the weight for F_{col} in our objective function; we empirically set w_{col} to 2.0.

F_{col} represents a quantitative collision cost determined by calculating the approximate distance between robot links and obstacles. We first introduce a method of computing F_{col} using the SDF and then explain our deep neural network for efficiently avoiding dynamic obstacles (Sec. IV-C).

The SDF can significantly reduce the computational overhead during the planning stage by calculating the distance information for static obstacles in advance. Thus, many works [6], [7] use the SDF to improve the performance of their optimization-based planners. For the effective use of the SDF, we simplify the robot links used for manipulation into spheres, $B = \{b_1, b_2, ..., b_n\}$ (Fig. 3). Through this method, we can calculate an approximate distance, $\tilde{d}_i$, from the nearest obstacles to the i-th sphere b_i, simply as $SDF(p_{b_i}) - r_{b_i}$, where $p_{b_i} \subset \mathbb{R}^3$ and r_{b_i} are correspondingly the center position and radius of b_i, and $SDF(p_{b_i})$ returns the SDF value at p_{b_i}.

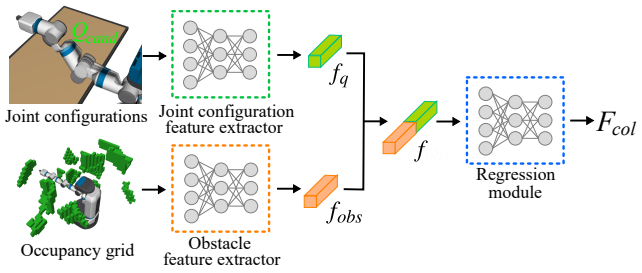
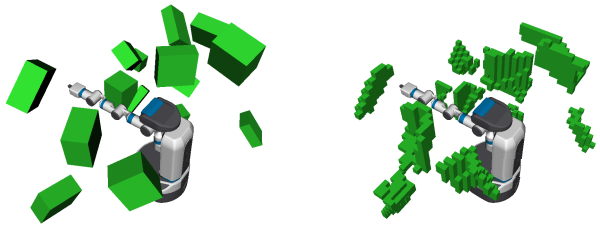


Fig. 4. This figure shows the collision-cost prediction network (CCPN), which consists of two feature extractors and one regression module.



(a) Randomly distributed obstacles. (b) Occupancy grid.

Fig. 5. These figures show one of the generated environments to obtain training data. From (*a*) randomly distributed obstacles, we construct (*b*) an occupancy grid with updates from sensor data.

Unlike most existing optimization-based methods that calculate a cost considering continuity between joint configurations [6], [7], we compute F_{col} only for a joint configuration q because the desired joint configuration q_{des} is very close to the current configuration q_{cur} to satisfy the joint velocity limits given the short time t. Therefore, F_{col} using the SDF is represented as:

$$F_{col} = \sum_{b}^{|B|} \max\left(\varepsilon - (SDF(P(q,b)) - r_b), 0\right), \tag{3}$$

where $P(q,b)$ is the partial forward kinematics that computes the position of a sphere $b \in B$ at a joint configuration q, and ε is the clearance value to make zero when the distance from the obstacle is far enough; ($SDF(\cdot) < \varepsilon$), and we set ε to $1.0m$. In short, F_{col} increases as the distance between obstacles and robot links decreases.

F_{smooth} in Eq. 2 serves to synthesize smooth joint configurations for consecutive commands. We consider the joint velocity and acceleration from the past three configurations. F_{smooth} is formulated via:

$$F_{smooth} = \|q_i - q_{i-1}\|^2 + \|\dot{q}_i - \dot{q}_{i-1}\|^2. \tag{4}$$

C. Collision-Cost Prediction Network

Although the SDF helps to accelerate the calculation of a collision cost F_{col}, it is difficult to use in dynamic environments due to its long construction time, e.g., 61ms for a $0.025m$ resolution. Furthermore, as the resolution of the SDF increases, longer construction times become necessary. Unfortunately, our approach needs a high-resolution of the SDF to accurately differentiate F_{col} of the IK candidates Q_{cand}, which are located very close to each other to satisfy the joint velocity limits given the short time t.

For a real-time approach in dynamic environments, we present a collision-cost prediction network (CCPN) that estimates F_{col} of Q_{cand} directly without constructing a SDF. We set the ground-truth data to collision costs computed from Eq. 3 using the SDF with a high-resolution of $0.005m$. Although the high-resolution SDF requires a long construction time and large amount of memory, it can provide very accurate distance information from obstacles. In addition, our CCPN serves to select a joint configuration far away from obstacles among Q_{cand}, and thus it learns to predict not only the value of F_{col} but also the ranking order for F_{col} of Q_{cand}.

Network architecture The CCPN aims to estimate collision costs close to computation by Eq. 3. As explained in Sec. IV-B, Eq. 3 is computed from the distance information of the obstacles in the SDF and the positions of the target robot links, as simplified links B according to a joint configuration q. We design the architecture of the CCPN by reflecting such information of Eq. 3.

As shown in Fig. 4, the CCPN extracts deep features from a joint configuration and an occupancy grid, and the regression module predicts F_{col} by concatenating the extracted two features. Although the CCPN has a simple architecture consisting of two feature extractors and one regression module, it is suitable for our real-time approach; it takes 2.3 ms to compute collision costs of 50 IK candidates.

At a high level, the regression module is designed to behave using an approach similar to that in Eq. 3 by predicting collision costs from the concatenated feature out of two features; the joint configuration feature, f_q, and the obstacle feature, f_{obs}. f_q has position information pertaining to the robot links, and f_{obs} contains obstacle information from an occupancy grid in the role of the SDF.

An occupancy grid includes information about obstacles and is updated from sensor data, while removing sensor noise based on the probabilistic update rule [15]. For real-time update, we set the resolution of the occupancy grid to $0.05m$ and the size of the occupancy grid is $40 \times 40 \times 40$. Our network extracts f_{obs} from the occupancy grid of $0.05m$, and f_{obs} serves as the high-resolution SDF. Similarly, some super-resolution and shape representation methods using a deep neural network have produced high-resolution results even with low-resolution inputs [30], [31].

In the detailed structure of the CCPN, the feature extractor of the joint configuration has six one-dimensional (D) convolution layers and two linear layers. The feature extractor of the environment has a structure identical to that of the feature extractor of the joint configuration, except it has 3-D convolution layers instead of 1-D convolution layers. In addition, the regression module is a multi-layer perceptron with six hidden layers. We leverage batch normalization and a ReLU activation function between all layers in the CCPN.

Loss functions Prediction results from the CCPN are used to select one of Q_{cand} in conjunction with F_{smooth}. Therefore, it is important for the predicted values and, more importantly, the predicted ranking among Q_{cand}, to close to those of the

ground-truth data. We use the mean squared error (MSE) loss, L_{MSE}, to match the ground-truth data, and the ranking loss, L_{rank}, to select one of Q_{cand} far away from obstacles:

$$loss = L_{MSE} + w_{rank} L_{rank}, \tag{5}$$

where w_{rank} is the weight for L_{rank} in our loss function.

Some deep learning approaches [32], [33] have shown that considering the ranking order of the data is effective when selecting the desired data. Inspired by these approaches, we also aim to preserve the ranking order of two different estimations for our method to select the best joint configuration among the IK candidates Q_{cand}.

L_{rank} is measured by comparing a data pair, and we make pairs by dividing a subset of the training data used in one training iteration, as mini-batch data, in half. For this type of data pair, L_{rank} computes the difference between two prediction values, $\hat{y_1}$ and $\hat{y_2}$, as to whether their ranking order is incorrect. Otherwise, L_{rank} is 0. Specifically, L_{rank} is formulated as:

$$\begin{aligned} L_{rank} &= \max(\Gamma(y_1, y_2) \times (\hat{y_1} - \hat{y_2}), 0), \\ s.t. \quad \Gamma(y_1, y_2) &= \begin{cases} 1, & y_1 < y_2, \\ -1, & \text{otherwise}, \end{cases} \end{aligned} \tag{6}$$

where y_1 and y_2 are the ground-truth values for $\hat{y_1}$ and $\hat{y_2}$, respectively.

Training dataset To train the CCPN, we construct datasets that consist of 500 random joint configurations in 2000 different environments for one million data instances in total. Some prior learning-based motion planning methods [34], [35] constructed datasets from randomly generated obstacles to cover a diverse set of environments. By adopting this protocol, we construct the environments by randomly positioning obstacles with arbitrary sizes (Fig. 5). From each environment, we extract occupancy grids and F_{col} of the joint configurations as the ground-truth data through using Eq. 3 with the high resolution of SDF; the SDF is built from the extracted occupancy grid, same as the network input information.

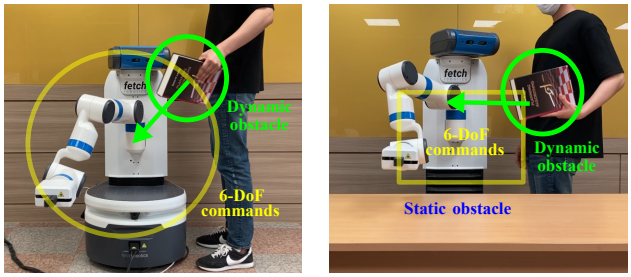
The CCPN is trained using the Adam optimizer with an initial learning rate of 5×10^{-5} and set the mini-batch size to 128. The training process takes 100 epochs, and the learning rate is decayed to 5×10^{-6} at 50 epoch.

V. Experiments And Analysis

In this section, we introduce our experiment setting and discuss the experimental results. Various experiments are tested on a machine that has a 3.60GHz Intel i7-9700K CPU, 32GB of RAM, and a RTX 2080 Ti graphics card.

A. Experiments in dynamic environments

In dynamic environments, we tested our method using a real robot, the Fetch manipulator; for sensing dynamic obstacles, we used the 3-D RGB-D camera mounted on the robot head. To test the usability of the proposed method, CCPN, we also compare its results against using a SDF. A SDF cannot be used directly in a dynamic environment, especially when the construction time is longer than the given time, 30ms. We therefore test the SDF with a resolution of $0.05m$, which has



(a) Circle tracing (b) Square tracing

Fig. 6. These figures show our experiment in the dynamic environments using a real robot, the Fetch manipulator. The robot receives 6-DoF commands consecutively to trace (a) circle and (b) square, while avoiding an incoming book in the direction of the green arrow. Additionally, (b) includes a static obstacle, a table. Note that we give 6-DoF commands consecutively to follow the yellow line.

TABLE I
Construction times of the SDF with varying resolutions and computation time of collision costs for 50 IK candidates using the SDF and the CCPN.

Fetch 7-DoF	SDF			CCPN
Resolution (m)	0.005	0.025	0.05	-
Construction time (ms)	8400	61	7	**0**
Computation time (ms)	17.5	6.0	4.5	**2.3**

a low construction time, $7ms$ (Table I). In addition, we test two types of CCPNs learned with and without L_{rank}.

We prepare three dynamic problems with different obstacles and 6-DoF commands. The first problem is to carry out the command in the form of a straight line transmitted using the joystick, while avoiding an incoming cart (Fig. 1). Next, the second and third problems consecutively give 6-DoF commands to trace a circle and a square, while avoiding an approaching book (Fig. 6). The third problem also involves a table as a static obstacle. The 6-DoF commands for three problems are provided consecutively, and our method progressively computes a joint configuration for each single command; our method does not know the future commands. In addition, since these dynamic problems are designed to evaluate the proposed method, the dynamic obstacle deliberately approaches the robot.

To determine the obstacles, we update the occupancy grid from sensor data in real-time [15]. As implementation details, we filter out sensor data corresponding to the manipulator, the spheres B shown in Fig. 3(a), as those data are not part of the environment. For the real-time update of the occupancy grid, including the filtering process, we used the occupancy grid with a resolution of $0.05m$; its update time is $30ms$ on average. The updated occupancy grid is used to check the collision with obstacles and as input to our CCPN for grasping the environment information; in the case of static obstacles, we can use prior obstacle information when precise collision checking is required, e.g., narrow passage. It should be noted self-collision detection is performed with the robot geometry information.

In addition, conducting experiments in real dynamic environments always involves different speeds of dynamic obstacles and timings of the human commands, which is inappropriate for fairly testing different methods. For a fair evaluation

TABLE II
RESULTS WITH 20 EXPERIMENTS WITH DIFFERENT METHODS IN DYNAMIC ENVIRONMENTS.

Fetch 7-DoF	Straight (Fig. 1)			Circle (Fig. 6(a))			Square (Fig. 6(b))		
	PE	OE	NF	PE	OE	NF	PE	OE	NF
SDF (0.05m res.)	5.5e-5	1.0e-4	12	-	-	20	4.3e-5	7.1e-5	15
CCPN ($w_{rank} = 0$)	5.6e-5	9.5e-5	7	5.0e-5	7.8e-5	9	1.8e-4	1.1e-4	6
CCPN ($w_{rank} = 10$)	6.3e-5	1.1e-4	**3**	5.3e-5	8.4e-5	**1**	6.6e-5	1.0e-4	**2**

PE: Position error (m). OE: Orientation error (rad). NF: Number of failures.

of different methods, we use a recorded data consisting of sensor data and consecutive commands obtained from a real experiment. Concretely, we prepare 20 different recorded data for each problem through real experiments; the 20 recorded data have slightly different speeds of dynamic obstacles and timings of the commands.

From the recorded data, we evaluate whether the computed joint configurations accurately match the given end-effector poses, while achieving the motion feasibility in dynamic problems. We measure the position and orientation error by adding one more between a current and a given end-effector pose for precise computation, since our target robot, a redundant manipulator, has multiple joint configurations for a target end-effector pose. We also regard failure when we do not find a joint configuration with the motion feasibility for each problem; in this case, we stop the movement of the robot.

Table II shows the results in dynamic environments with different methods using a 0.05m resolution of SDF and two types of CCPNs trained by only L_{MSE} and by L_{MSE} and L_{rank} together. Using a 0.05m resolution of the SDF, which satisfies the real-time performance requirement, shows the highest number of failures among the tested methods, as its resolution cannot accurately distinguish F_{col} of nearly located Q_{cand}. Moreover, the CCPN learned w/ L_{rank} shows fewer failures than the CCPN w/o L_{rank} due to being learned to distinguish F_{col} of Q_{cand}.

When measuring the position and orientation errors for the success cases, all methods show reasonably low errors. This is because all methods generate Q_{cand} through our generation method, and generated Q_{cand} have fairly low errors for x_{tar} to follow a given command.

In conclusion, our proposed method performs consecutively given commands in real-time while avoiding dynamic and static obstacles effectively. A quantitative analysis of the proposed method will be discussed in the next sub-section.

B. Analysis

To analyze our proposed method, we construct validation sets using three different configurations of robots, Kuka iiwa with 7-DoF, and the Fetch manipulator with 7 and 8 DoF. We also evaluate the performance of the CCPN with varying values of w_{rank} and the SDF with different resolutions.

To evaluate our proposed method in various environments, we prepare 200 environments with randomly distributed obstacles as shown in Fig. 5(a). Each environment has ten random end-effector poses, and we obtain 50 IK candidates for one end-effector pose. The reason we use 50 IK candidates as a standard is that our approach generates about 125 IK candidates on average for a given 30ms, and about 50 of them satisfy the constraints except for collision. We ensure that the 50 candidates have different collision costs so as to evaluate whether the predicted collision costs have the correct ranking; we compute the collision costs at a resolution of 0.005m of SDF to achieve high accuracy. Furthermore, the 50 IK candidates are located within a very narrow joint boundaries considering the joint velocity limit and given short time.

TABLE III
RESULTS OF USING THE SDF WITH VARYING RESOLUTIONS AND THE CCPN TRAINED BY VARYING w_{rank} IN THE VALIDATION DATASET. THE NUMBER OF UNIQUE DATA INDICATES THE NUMBER OF DIFFERENT COLLISION COSTS AMONG 50 IK CANDIDATES.

		SDF w/ varying resolutions		CCPN w/ varying w_{rank}		
		0.025m	0.05m	0	10	20
Kuka	Ranking error	7.9	12.6	5.9	**5.8**	**5.8**
7-DoF	# of unique data	13	6	**50**		
Fetch	Ranking error	7.5	11.1	7.0	**6.6**	6.9
7-DoF	# of unique data	20	16	**50**		
Fetch	Ranking error	8.0	9.1	8.0	**7.6**	7.7
8-DoF	# of unique data	44	43	**50**		

From the validation sets, we compute F_{col} for the different tested methods. We evaluate how well estimated F_{col} of 50 IK candidates have similar ranking orders among them compared to the corresponding ground-truth values. To measure the similarity, we measure the ranking orders of the predicted outcomes and do the same procedure for the ground-truth values, after which we compute the L1 distance between them. In this work, we refer to this metric as the "ranking error". Intuitively, the low ranking error implies that it is highly possible to select an IK candidate far away from obstacles among various IK candidates.

Table III shows the ranking error of different SDFs and CCPNs with varying configurations of the robots. Overall, the 0.05m resolution of the SDF has the highest ranking error. We also measure how unique the values for the SDF and CCPN are. The tested SDF has many more duplicate values over the tested CCPN. Although the accuracy of the SDF can be improved as we increase the resolution, its construction time increases exponentially (Table I), becoming inappropriate for our real-time purpose.

On the other hand, the CCPN does not require construction time at runtime and has fewer ranking errors than the SDF with resolutions of 0.05m and even 0.025m. Furthermore, the CCPN computes collision costs of 50 IK candidates nearly twice as fast compared to the use of tested SDF with the 0.05m resolution (Table I). Owing to such performances, our method achieved an average of 90% success rate for three dynamic problems. These results indicate that the CCPN has computational advantages for our real-time approach in dynamic environments and selects a joint configuration away from obstacles among Q_{cand}.

In addition, the CCPN which underwent learning through L_{MSE} and L_{rank} shows smaller ranking errors (Table III) and higher success rates in dynamic environments than the CCPN trained by only L_{MSE} (Table II). In particular, the greatest difference was found in the circle problem, as shown in the middle column of Table II. This improvement stems from the

fact that L_{rank} reduces the ranking error of the CCPN, allowing a better IK candidate to be selected.

Lastly, we test our method for three different configurations of robots in four static problems, which were used in earlier work [6]. Our method successfully solves the given problems, and the results can be available in the accompanying video.

VI. Conclusion

We presented real-time collision-free inverse kinematics (RCIK) for performing consecutive 6-DoF commands accurately in environments including static and dynamic obstacles. For real-time solving, we proposed a simple method that generates IK candidates with a high probability of achieving the motion feasibility. We also suggested a collision-cost prediction network for handling dynamic obstacles. We demonstrated the benefits that our method, which can accurately perform the consecutively given commands while avoiding obstacles in real-time. Furthermore, we verified the feasibility of the proposed approach using a real robot, the Fetch manipulator.

Our method allows real-time solving in static and dynamic environments by quickly predicting collision costs using a deep neural network, while most of the time, i.e., approximately 75% of the overall time for our method, is used to generate IK candidates. Our method successfully solved three dynamic and four static problems using three different configuration robots with 7 or 8 DoF, but the number of IK candidates may not be sufficient for higher DoF robots, e.g., hyper-redundant robots. In future work, we would like to improve our generation method of IK candidates using a generative model to overcome the scalability issue of robot DoF and increase the quality and quantity of IK candidates.

Acknowledgment

The authors would like to thank the anonymous reviewers for constructive comments and insightful suggestions.

References

[1] S. Kawatsuma, M. Fukushima, and T. Okada, "Emergency response by robots to fukushima-daiichi accident: summary and lessons learned," *Industrial Robot: An International Journal*, vol. 39, no. 5, pp. 428–435, 2012.

[2] C. Meng, T. Wang, W. Chou, S. Luan, Y. Zhang, and Z. Tian, "Remote surgery case: robot-assisted teleneurosurgery," in *ICRA*, vol. 1. IEEE, 2004, pp. 819–823.

[3] G. Gorjup, A. Dwivedi, N. Elangovan, and M. Liarokapis, "An intuitive, affordances oriented telemanipulation framework for a dual robot arm hand system: On the execution of bimanual tasks," in *IROS*. IEEE, 2019, pp. 3611–3616.

[4] D. Rakita, B. Mutlu, and M. Gleicher, "RelaxedIK: Real-time synthesis of accurate and feasible robot arm motion." in *RSS*, 2018.

[5] 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 *ICRA*. IEEE, 2021, pp. 9995–10 001.

[6] M. Kang, H. Shin, D. Kim, and S.-e. Yoon, "TORM: Fast and accurate trajectory optimization of redundant manipulator given an end-effector path," in *IROS*. IEEE, 2020.

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

[8] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, no. 1-19, p. 16, 2004.

[9] R. Diankov, "Automated construction of robotic manipulation programs," 2010.

[10] A. Sinha and N. Chakraborty, "Geometric search-based inverse kinematics of 7-dof redundant manipulator with multiple joint offsets," in *ICRA*. IEEE, 2019, pp. 5592–5598.

[11] P. Beeson and B. Ames, "TRAC-IK: An open-source library for improved solving of generic inverse kinematics," in *Humanoids*. IEEE, 2015, pp. 928–935.

[12] D. Rakita, B. Mutlu, and M. Gleicher, "STAMPEDE: A discrete-optimization method for solving pathwise-inverse kinematics," in *ICRA*. IEEE, 2019, pp. 3507–3513.

[13] R. M. Holladay and S. S. Srinivasa, "Distance metrics and algorithms for task space path optimization," in *IROS*. IEEE, 2016, pp. 5533–5540.

[14] R. Holladay, O. Salzman, and S. Srinivasa, "Minimizing task-space fréchet error via efficient incremental graph search," *RA-L*, vol. 4, no. 2, pp. 1999–2006, 2019.

[15] Y. Kwon, D. Kim, I. An, and S.-e. Yoon, "Super rays and culling region for real-time updates on grid-based occupancy maps," *T-RO*, vol. 35, no. 2, pp. 482–497, 2019.

[16] C. Park, J. Pan, and D. Manocha, "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments," in *International Conference on Automated Planning and Scheduling*, 2012.

[17] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *IROS*. IEEE, 2017, pp. 1366–1373.

[18] A. Sud, M. A. Otaduy, and D. Manocha, "Difi: Fast 3d distance field computation using graphics hardware," in *Computer Graphics Forum*, vol. 23, no. 3. Wiley Online Library, 2004, pp. 557–566.

[19] H. Yu, J. Xie, K.-L. Ma, H. Kolla, and J. H. Chen, "Scalable parallel distance field construction for large-scale applications," *IEEE transactions on visualization and computer graphics*, vol. 21, no. 10, pp. 1187–1200, 2015.

[20] L. Zhao, J. Zhao, H. Liu, and D. Manocha, "Efficient inverse kinematics for redundant manipulators with collision avoidance in dynamic scenes," in *International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2018, pp. 2502–2507.

[21] D. Han, H. Nie, J. Chen, and M. Chen, "Dynamic obstacle avoidance for manipulators using distance calculation and discrete detection," *Robotics and Computer-Integrated Manufacturing*, vol. 49, pp. 98–104, 2018.

[22] J. C. Kew, B. Ichter, M. Bandari, T.-W. E. Lee, and A. Faust, "Neural collision clearance estimator for fast robot motion planning," *arXiv preprint arXiv:1910.05917*, 2019.

[23] P. Chang, "A closed-form solution for inverse kinematics of robot manipulators with redundancy," *IEEE Journal on Robotics and Automation*, vol. 3, no. 5, pp. 393–403, 1987.

[24] H. Zhang, S. Starke, T. Komura, and J. Saito, "Mode-adaptive neural networks for quadruped motion control," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–11, 2018.

[25] V. S. Chembuly and H. K. Voruganti, "An optimization based inverse kinematics of redundant robots avoiding obstacles and singularities," in *Proceedings of the Advances in Robotics*, 2017, pp. 1–6.

[26] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *ICRA*. IEEE, 2012, pp. 3859–3866.

[27] T. Yoshikawa, "Manipulability of robotic mechanisms," *IJRR*, vol. 4, no. 2, pp. 3–9, 1985.

[28] J. Vannoy and J. Xiao, "Real-time adaptive motion planning (ramp) of mobile manipulators in dynamic environments with unforeseen changes," *T-RO*, vol. 24, no. 5, pp. 1199–1212, 2008.

[29] P. Lehner, A. Sieverling, and O. Brock, "Incremental, sensor-based motion generation for mobile manipulators in unknown, dynamic environments," in *ICRA*. IEEE, 2015, pp. 4761–4767.

[30] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs," in *ICCV*. IEEE, 2017, pp. 2088–2096.

[31] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue, and Q. Liao, "Deep learning for single image super-resolution: A brief review," *IEEE Transactions on Multimedia*, vol. 21, no. 12, pp. 3106–3121, 2019.

[32] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *ICML*, 2005, pp. 89–96.

[33] D. Yoo and I. S. Kweon, "Learning loss for active learning," in *CVPR*. IEEE, 2019, pp. 93–102.

[34] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *ICRA*. IEEE, 2019, pp. 2118–2124.

[35] X. Li, Q. Cao, M. Sun, and G. Yang, "Fast motion planning via free c-space estimation based on deep neural network," in *IROS*. IEEE, 2019, pp. 3542–3548.