2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Towards Safe Remote Manipulation: User Command Adjustment based on Risk Prediction for Dynamic Obstacles

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

***Abstract*— Real-time remote manipulation requires careful operations by a user to ensure the safety of a robot, which is designed to follow user's commands, against dynamic obstacles. However, a user may give commands to a robot at the risk of collision with dynamic obstacles due to a user's unfamiliar control ability or unexpected situations. In this paper, we propose a risk-aware user command adjustment method to avoid potential collision with dynamic obstacles. Our method consists of a network that predicts the risk of dynamic obstacles and another network that synthesizes commands to avoid obstacles. Based on the predicted risk, our method decides an adjusted command between a user command and a command to avoid collisions. We evaluate our method in problems that face collisions with dynamic obstacles when following given commands and in problems with static obstacles. We show that our method improves safety against the risk of dynamic obstacles or follows user commands when there is no risk. We also demonstrate the feasibility of our method using the real fetch manipulator with seven-degrees-of-freedom.**

I. Introduction

Real-time remote manipulation has been primarily used in special sites (e.g., medical facilities or nuclear power plants) that require sophisticated or hazardous work [1], [2]. Recent studies [3], [4], [5] have expanded the scope of remote manipulation to environments around us, such as convenience stores (e.g., Telexistence[1]) and homes. In environments around us, there are various obstacles, both static and dynamic, and avoiding such obstacles is a critical issue for safe remote manipulation.

Recent proposed inverse kinematics (IK) methods for real-time remote manipulation, e.g., CollisionIK [6] and RCIK [7], handle static and dynamic obstacles to find collision-free joint configurations from consecutively given user's commands. These IK approaches have the characteristic of following a user's command, and thus the user's judgment on giving proper commands also plays a big role in obstacle avoidance. However, a user makes judgments by observing a restricted environment through a camera, and thus there is a possibility that the user may encounter unexpected situations for giving commands in remote manipulation (Fig. 1).

In the case of static obstacles, we can keep the safety by naïvely stopping a robot, even if the user's command has a possibility of causing a collision. However, it is difficult to guarantee the safety against dynamic obstacles with uncertain motion [8], [9]. To increase the probability of avoiding dynamic obstacles, it is necessary to identify the danger of dynamic obstacles in advance and take action to avoid them. For safe remote manipulation, we aim to adjust user's commands to avoid obstacles according to the risk of dynamic obstacles. Conversely, when there is no risk of dynamic obstacles, a robot follows the user's command so as not to interfere with the user's desired tasks, e.g., picking an object on a table.

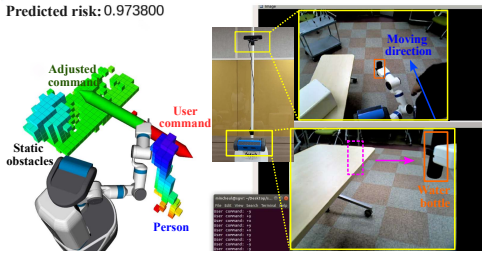


Fig. 1. This shows the visualization of the user monitor screen for remote manipulation. In the visualized situation, the user gives commands (red arrow) to the robot to move a water bottle (orange boxes) from the dotted magenta box to the direction of the magenta arrow, and a person suddenly appears in the camera's field of view (blue arrow). Our method predicts high risk of collision, which is expressed between 0 and 1, and for safe remote manipulation, it adjusts the user's command accordingly to move the robot away from the obstacle (green arrow). The cells of the occupancy grid are colored according to the z-axis value.

Main contributions. In this paper, we present a risk-aware user command adjustment method to adjust risky commands that can cause collisions with dynamic obstacles due to a user's unpredictability or carelessness (Fig. 1). We suggest a risk prediction network (RPN) and an obstacle avoidance command network (OACN) to minimize the delay of remote operation and to handle sensor noise, which is the difficulty of a real environment. Using two networks, our method predicts the risk of dynamic obstacles and then adjusts a user command to avoid obstacles depending on the predicted risk.

We model the risk of dynamic obstacles utilizing the proximity between robot links and dynamic obstacles, and the RPN learns the modeled risk values with consecutive robot state and obstacle information (Sec. IV-C); we use occupancy grids updated from sensor data in real-time to grasp obstacle information. Based on the predicted risk from the RPN, our method decides an adjusted command between a user command and the output of the OACN. We train the OACN via reinforcement learning so that the final adjusted command avoids obstacles when the predicted risk is high, otherwise, we follow the user's commands (Sec. IV-B).

[1]Mincheul Kang (mincheul.kang@kaist.ac.kr), [1]Minsung Yoon (minsung.yoon@kaist.ac.kr) are with the School of Computing, and [2]Sung-Eui Yoon (Corresponding author, sungeui@kaist.edu) is with the Faculty of School of Computing, KAIST at Daejeon, Korea 34141

[1]https://tx-inc.com

We evaluate our proposed method in environments including dynamic obstacles varying velocities and environments including only static obstacles (Sec. V-A). We show that our method improves the safeness in dynamic environments where a collision occurs when naïvely following given commands. We also show that our RPN robustly handles sensor noise by predicting the risk of dynamic obstacles close to zero in static environments. Lastly, our method shows a fast computation time of 3*ms*, and we verify the feasibility of our method in real environments (Sec. V-B).

II. Related Work

In this section, we introduce inverse kinematics methods related to remote manipulation and motion planning approaches for dynamic obstacle avoidance.

A. Inverse kinematics for remote manipulation

In remote manipulation, a user can easily control a manipulator by giving a command to the manipulator's end-effector in the Cartesian space. Accordingly, inverse kinematics (IK) techniques have been widely used in remote manipulation to compute a joint configuration for the user command [10], [11], [12]. Conventional IK approaches (e.g., IKfast [13] and Trac-IK [14]) focus on finding a joint configuration that matches the given end-effector pose. However, to synthesize feasible joint configurations for consecutively given user commands, it is necessary to consider several constraints, such as collision, continuity of joints, and kinematic singularity [7].

RelaxedIK [15] handles several constraints using an optimization-based technique and uses a neural network for real-time remote manipulation to quickly avoid self-collision. Extending RelaxedIK, CollisionIK [6] avoids static and dynamic obstacles by adding a collision avoidance term that quickly computes the shortest distance between convexified robot links and obstacles using the QuickHull algorithm. However, this method has difficulty coping with sensor noise in a real environment.

RCIK [7] also handles static and dynamic obstacles, while overcoming the difficulty of a real environment by utilizing deep learning with an occupancy grid via real-time updates based on probability [16]. In addition, RCIK accurately follows given commands through a sampling-based IK approach. Although the high accuracy is one of the important factors in remote manipulation, it requires a user's proper and fast judgment to avoid collisions of robot links, especially related to the end-effector. Since users cannot always make perfect decisions in a timely manner due to carelessness or unforeseen circumstances, we aim to adjust user commands to reduce the risk of collision for safe remote manipulation.

B. Dynamic obstacle avoidance

Motion uncertainty of dynamic obstacles threatens safe robot movement and makes it difficult to plan a collision-free robot motion for reaching a target position [17]. Classical motion planning approaches [8], [18], [19], [20] handling dynamic obstacles perform iterative replanning in a short time considering the sensing cycle. To overcome the short replanning time, Vannoy et al. [8] suggest a method that generates multiple initial trajectories to a target configuration and then iteratively updates the trajectories. Hauser et al. [18] present an adaptive time-stepping approach for replanning, considering responsiveness, safety, and completeness of planning results. In addition, Park et al. [19] present GPU-based parallel processing to accelerate optimization-based replanning.

Recently, learning-based approaches [21], [22], [23] have been studied to quickly replan a trajectory for reaching a target position using a fast inference time of a neural network. Furthermore, several works [24], [9] construct a learning-based safety layer that iteratively checks collisions on a planned trajectory and modifies the collision part of the trajectory to be safe.

While these methods consider avoiding obstacles without distinction between static and dynamic, we mainly consider the risk of dynamic obstacles that are difficult to ensure safety in remote manipulation. Since we can ensure safety from static obstacles by stopping a robot, we aim to follow user's commands when there is no danger of dynamic obstacles to avoid disturbing the user's work progress, e.g., placing an object on the bookshelf. Hence, our method predicts the risk of dynamic obstacles and determines the final command between the user's command and a command to avoid obstacles according to the predicted risk.

III. Background

A. Problem definition

Our goal is to perform safe remote manipulation in environments including static and dynamic obstacles. In this paper, we deal with real-time remote manipulation in which a user consecutively gives a command, $\Delta x_u \in \mathbb{R}^6$, to the end-effector in the Cartesian space. Δx indicates the amount of movement for the end-effector, and a target end-effector pose, x_u, from a user command is calculated by adding a current end-effector pose, x_{cur}, and Δx_u; $x_u = x_{cur} + \Delta x_u$. For x_u, a real-time IK solver, such as RCIK [7], synthesizes a joint configuration, q_u, considering various constraints, e.g., continuity of joints, collision avoidance, and kinematic singularity. In this paper, we mainly consider a redundant manipulator with multiple joint configurations for one end-effector pose; a redundant manipulator has greater than six degrees of freedom (DoF).

In real-time remote manipulation, a user's decision-making greatly affects the process of avoiding obstacles since a robot is designed to follow the user command Δx_u. In the case of static obstacles, we can maintain the safety from collisions by naïvely stopping a robot, since we can check whether joint configuration for x_u is collision or not [7]. On the other hand, it is more difficult to avoid collision with dynamic obstacles that have even uncertain movement. Unfortunately, a user may give commands that lead to collisions due to the user's unfamiliar control skills or unexpected situations. Therefore,

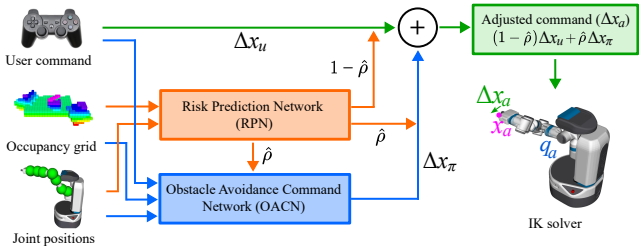


Fig. 2. This shows the overall system flow of our approach.

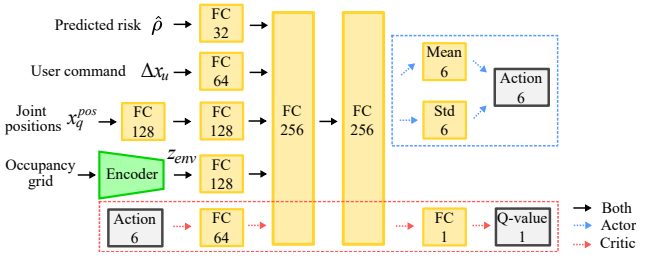


Fig. 3. This shows the structure of obstacle avoidance command network (OACN); note that both networks for actor and critic do not share features. The encoder is a 3-D convolutional network following the obstacle feature extractor of RCIK [7].

there is a strong demand to protect a robot from risky commands that can cause collision with dynamic obstacles.

In this paper, our problem is to cope with the danger of dynamic obstacles where adjusting the user command is essential. To solve our problem, we have to figure out the risk of dynamic obstacles in advance, and adjust the user's commands to avoid obstacles considering the risk of dynamic obstacles. When no risk of dynamic obstacles is expected, we simply follow user command.

In addition, we handle the noise problem of real sensor data to apply our method on a real environment and aim to solve our problem as quickly as possible to reduce the delay of real-time remote manipulation; in real-time IK solvers [15], [6], [7], the time interval, Δt, between commands is within $30ms$ according to the sensing cycle.

B. Reinforcement Learning

We use reinforcement learning to solve our problem by formulating our problem as Markov decision process (MDP) defined by a tuple: $(\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R},\gamma)$, where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, $\mathcal{R}$, and $\gamma \in [0,1)$ are the state space, the action space, a transition function, a reward function, and a discount factor, respectively [25]; more details can be found in Sec. IV-B. We find an optimal policy, $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$, that maximizes a discounted cumulative reward, $\mathcal{G}$, via the soft actor-critic (SAC) [26] framework. $\mathcal{G}$ is denoted as $\sum_{t=0}^{T} \gamma^t \mathbb{E}[\mathcal{R}(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))]$, where s_t and a_t are the subset of $\mathcal{S}$ and $\mathcal{A}$, respectively, at each time step $t \in [0,T]$, and α and $\mathcal{H}$ are regulation coefficients for controlling the stochasticity of π and the entropy of π, respectively.

IV. Approach

In this section, we give an overview of our method and then describe our approach in detail.

A. Overview

To solve our problem, we present a risk-aware user command adjustment method using deep learning. We adopt a learning-based approach to quickly adjust a risky user command for dynamic obstacles and cope with sensor noise. Fig. 2 shows our system flow. Our method consists of two kinds of networks: risk prediction network (RPN) and obstacle avoidance command network (OACN).

The RPN predicts the risk of dynamic obstacles, $\hat{\rho} \in [0,1]$, in environments including static obstacles (Sec. IV-C). To identify the risk of dynamic obstacles, the RPN uses consecutive robot state information and occupancy grids updated in real-time from sensor data. According to the predicted risk $\hat{\rho}$, our method decides an adjusted command, Δx_a, between the user command Δx_u and a command, Δx_π, generated by OACN:

$$\Delta x_a = (1-\hat{\rho})\Delta x_u + \hat{\rho}\Delta x_\pi. \quad (1)$$

The OACN is learned via reinforcement learning to find an optimal Δx_π considering the predicted risk, the user command Δx_u, current robot state, and obstacle information (Sec. IV-B); note that an optimal Δx_π is computed in a way that Δx_a gets the maximum reward. Finally, to find a joint configuration, we give the adjusted command Δx_a to a real-time IK solver considering collision avoidance.

B. User command adjustment

We decide an adjusted command Δx_a by combining a user command Δx_u and a command Δx_π generated by the OACN according to the predicted risk $\hat{\rho}$ (Eq. 1). We learn the OACN through reinforcement learning so that Δx_a avoids obstacles at high $\hat{\rho}$ and follows a user's command at low $\hat{\rho}$. To achieve the desired Δx_a, we introduce our MDP setup based on $\hat{\rho}$.

We learn our policy π, as the OACN, to maximize a discounted cumulative reward $\mathcal{G}$ for Δx_a. Our policy π generates $a_t \in \mathbb{R}^6$ from the current state s_t (Fig. 3), and we synthesize Δx_π from a_t:

$$\Delta x_\pi = diag(\boldsymbol{\lambda}_{pos}, \boldsymbol{\lambda}_{ori}) a_t, \quad (2)$$

where each element of a_t has a value between -1 to 1, and $diag(\boldsymbol{\lambda}_{pos}, \boldsymbol{\lambda}_{ori})$ is a diagonal matrix to control the amount of movement of the end-effector during the short Δt; we set $\boldsymbol{\lambda}_{pos}$ and $\boldsymbol{\lambda}_{ori}$ to $[0.01, 0.01, 0.01]$ and $[0.05, 0.05, 0.05]$, respectively.

We define s_t as a tuple: $(\hat{\rho}, \Delta x_u, x_q^{pos}, z_{env})$, where $x_q^{pos} \in \mathbb{R}^{d\times 3}$ is the 3-dimensional (D) position for d joints, and z_{env} is an obstacle feature compressed from an occupancy grid, O. We extract z_{env} from a pre-trained encoder, which is part of the variational autoencoder (VAE) [27].

Risk-based reward function. We introduce our reward function based on the predicted risk $\hat{\rho}$ to synthesize the desired Δx_a. We define the reward function so that when $\hat{\rho}$ is high, Δx_a moves away from obstacles, and when $\hat{\rho}$ is low, Δx_a gets closer to Δx_u:

$$\mathcal{R} = \begin{cases} (1-\hat{\rho})\mathcal{R}_{match} + \hat{\rho}\mathcal{R}_{obs}, & \text{if IK solution exists,} \\ \mathcal{R}_{fail}, & \text{otherwise,} \end{cases} \quad (3)$$

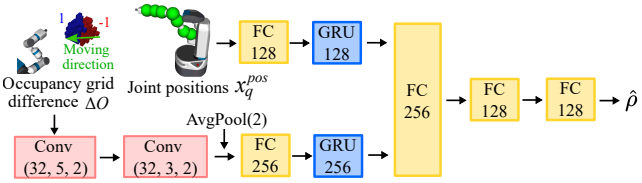
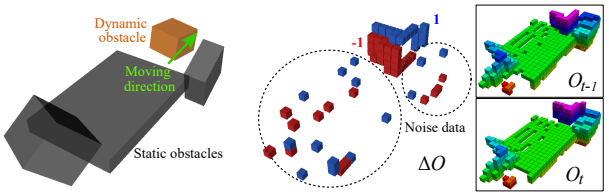


Fig. 4. This shows our risk prediction network (RPN) for dynamic obstacles. The RPN recurrently takes joint positions x_q^{pos} and an occupancy grid difference ΔO. We apply VoxNet [28] to extract the feature of an occupancy grid difference ΔO. Furthermore, we utilize the gate recurrent unit (GRU) [29] to preserve previous robot state and obstacle information.



(a) Obstacle environment (b) Occupancy grid difference

Fig. 5. (a) shows an example of a randomly generated obstacle environment. Gray boxes are static obstacles, and the orange box is a dynamic obstacle. The green arrow indicates the moving direction of the dynamic obstacle. (b) shows the occupancy grid difference ΔO between O_t and O_{t-1} constructed from sensor data for (a). Each cell in O is represented as 1 if there is an obstacle or 0 otherwise. Therefore, in the occupancy grid difference, 1 (blue boxes) and -1 (red boxes) indicates newly detected and disappeared obstacles, respectively. From the occupancy grid difference, we can see the moving direction of the dynamic obstacle, but the noise data due to sensor noise occurs near static obstacles (dotted black circles).

where $\mathcal{R}_{match}$ is to match Δx_u, and $\mathcal{R}_{obs}$ is to avoid obstacles. When there is no IK solution for Δx_a due to collision or inaccessibility, we set $\mathcal{R}$ to $\mathcal{R}_{fail}$; we set $\mathcal{R}_{fail}$ to -10.

We compute $\mathcal{R}_{match}$ and $\mathcal{R}_{obs}$ by comparing Δx_u and Δx_a. Considering Δt, the difference between Δx_u and Δx_a is very small. To amplify the difference, we apply a normalization function to $\mathcal{R}_{match}$ and $\mathcal{R}_{obs}$. Also, it can control the relative importance of the two terms. We design a normalization function in the form of a cubic function: $F(v, \boldsymbol{w}) = w_0(w_1 v + w_2)^3$. Although the value of a cubic function can increase infinitely, we can amplify the value to a limited range, since the difference between Δx_u and Δx_a is bounded.

$\mathcal{R}_{obs}$ indicates whether the end-effector pose x_a from Δx_a is further away from obstacles than x_u from Δx_u:

$$\mathcal{R}_{obs} = F(dist_{x_a} - dist_{x_u}, \boldsymbol{w}_{obs}), \tag{4}$$

where $dist$ is a distance between obstacles and the end-effector; we set $\boldsymbol{w}_{obs}$ to $[1/3, 200, 0]$. We simply use the distance with the end-effector instead of the whole arm. This is because a collision-free IK solver [6], [7] has already taken into account a distance between the whole arm and obstacles. Furthermore, the distance for the whole arm makes it difficult to converge the policy π since it is not constant due to the various solutions for a given command.

$\mathcal{R}_{match}$ represents the difference between Δx_u and Δx_a:

$$\mathcal{R}_{match} = F(e_{pos} + \lambda_{match} e_{ori}, \boldsymbol{w}_{match}), \tag{5}$$

where e_{pos} and e_{ori} are the position and orientation error [14] between Δx_a and Δx_u, and λ_{match} is a constant to calibrate different units of position (m) and orientation (rad); we set $\boldsymbol{w}_{match}$ to $[-1, 100, -2]$ and λ_{match} to 0.17, as used in [12];

C. Risk prediction for dynamic obstacles

We aim to quickly predict the risk of dynamic obstacles in environments that contain both static and dynamic obstacles. To this end, we model the risk of dynamic obstacles, ρ, and present a risk prediction network (RPN) that learns ρ from consecutive robot state and obstacle information (Fig. 4). To figure out dynamic obstacles, the RPN recurrently takes an occupancy grid difference, ΔO, between O_t and O_{t-1}, motivated by the work of Villegas et al. [30] as the optical flow estimation method using an image difference.

Ideally, using the occupancy grid difference would remove the information on static obstacles, but in reality, it is difficult to obtain information on dynamic obstacles due to sensor noise. Even though an occupancy grid O is updated by the probabilistic rule to reduce sensor noise, it cannot completely eliminate sensor noise near static obstacles (Fig. 5). Therefore, we learn the RPN from a dataset including sensor noises to cope with the sensor noises; we give the Gaussian noise to sensor data and set the standard deviation to 0.008.

Modeling of the risk for dynamic obstacles. Most of the approaches [31], [32], [6] dealing with dynamic obstacles have used a distance between a robot and the obstacles and considered their moving direction and velocity. Inspired by these methods, we model the risk of dynamic obstacles ρ as the proximity, $prox_q^{dyn}$, between robot links and dynamic obstacles, and its amount of change. $prox_q^{dyn}$ is computed from the distance between spheres surrounding the robot links and dynamic obstacles [33], [12]. When a robot is close to dynamic obstacles, $prox_q^{dyn}$ has a large value. The risk of dynamic obstacles ρ is represented as:

$$\rho = \sigma\left(max\left(prox_{q_t}^{dyn} + \lambda_\rho \sum_{i=1}^{N} (prox_{q_{t-i+1}}^{dyn} - prox_{q_{t-i}}^{dyn})/i, 0\right)\right), \tag{6}$$

where λ_ρ is a weight for the change of $prox_q^{dyn}$, and N is the number of commands to consider changes in $prox_q^{dyn}$; we set λ_ρ to 2.0 and N to 3. Also, $\sigma(v) = tanh(v)$ is to represent ρ between 0 to 1 for easily computing Δx_a between Δx_u and Δx_π (Eq. 1).

D. Training details

We train the OACN via reinforcement learning using the SAC [26] algorithm and the RPN via supervised learning by gathering learning data during the training of the OACN. We prepare three kinds of obstacle scenes, including 1) static obstacles, 2) dynamic obstacles, and 3) both static and dynamic obstacles.

Each scene has box-shaped obstacles with random sizes and positions and has no more than five fixed obstacles and no more than two dynamic obstacles (Fig. 5(a)). Dynamic obstacles have velocity between $0.4m/s$ and $0.9m/s$. The velocity of the end-effector is less than $0.57m/s$ considering λ_{pos} and the short Δt; we set Δt to $30ms$ in simulation environments, considering the sensing cycle.

To recognize obstacles, we construct an occupancy grid O in real-time using the SuperRay [16]; the resolution of O is $0.05m$ and the size of O is $40 \times 40 \times 40$. Also, we used the extended sensing range to recognize the entire obstacle in the Gazebo simulator [34].

We set user commands to move a robot from a random start configuration, q_s, to a random goal pose, x_g, since it is difficult for a user to give commands by intervening the learning process. We decide a user command by considering simple linear movement; note that the case where a user command is not feasible, we do not use it to learn our networks. Each scene is terminated when a robot reaches x_g, or there is no IK solution for Δx_a due to collision or inaccessibility.

As one command means one step, we train the OACN during one million steps with 1×10^{-5} learning rate, 4096 batch size, 10^6 replay buffer size, $\gamma = 0.99$, and 0.995 polyak for target network update. We update the OACN 100 times every $1,000$ steps using the Adam optimizer [35].

We construct the RPN dataset with $60,000$ groups, and each group consists of 16 consecutive joint positions, occupancy grids, and ρs. We train the RPN during 100 epochs using the adam optimizer with 1×10^{-5} learning rate and the mean square error (MSE) loss. In addition, we use the ten thousand occupancy grids to train the encoder of the OACN. We train the VAE [27] for the pre-trained encoder with the same condition as the RPN.

V. Experiments

In this section, we describe our experimental setting and discuss our experimental results. Our experiments are tested on a machine equipped with a 3.60 GHz Intel i7-9700 K CPU and an RTX 2080 Ti graphics card. In these experiments, we use the Fetch manipulator with 7-DoF.

A. Evaluation

To evaluate our method, we prepare two kinds of environments one consisting of both static and dynamic obstacles and the other consisting of only static obstacles. In each environment, we construct $1,000$ problems with different configuration of obstacles and commands (Fig. 5(a)); the generation method of obstacles and commands is the same as making the training scenes (Sec. IV-D).

In the environments including both static and dynamic obstacles, we measure the success rate of avoiding dynamic obstacles at different velocities of dynamic obstacles. For reliable evaluation, we extract $1,000$ problems that cause collisions with dynamic obstacles when following given commands. In contrast, in the environments including only static obstacles, we extract $1,000$ problems where there is no collision with obstacles when following given command. In these problems, we measure whether the RPN predicts the risk of dynamic obstacles close to zero handling sensor noise that occurs near static obstacles; note that sensor data for obstacles in all problems include sensor noise.

Table I shows the results of RCIK [7] and our method on the various problems. RCIK is our baseline approach and is used in our method to find a joint configuration for a given command. Since RCIK is a method of finding a joint configuration for a given command with high accuracy, we can verify that our method appropriately adjusts user commands according to the risk of dynamic obstacles. Accordingly, RCIK fails on all problems involving dynamic obstacles that are set up to collide with dynamic obstacles when following given commands.

TABLE I

Results in two kinds of obstacle environments: both static and dynamic obstacles, and only static obstacles.

	Static and dynamic obs.					Static obs.
Velocity range of dynamic obs. (m/s)	[0.4, 0.5)	[0.5, 0.6)	[0.6, 0.7)	[0.7, 0.8)	[0.8, 0.9]	-
	Success rate (%)					Command error (mean, std)
RCIK [7]	0					-
Ours	93	86	83	78	74	6.6×10^{-5}, 8.1×10^{-5}

Command error: difference between Δx_u and Δx_a ($e_{pos} + \lambda_{match} e_{ori}$).

TABLE II

The mean and standard deviation of difference between ρ computed from Eq. 6 and $\hat{\rho}$ predicted by the RPN.

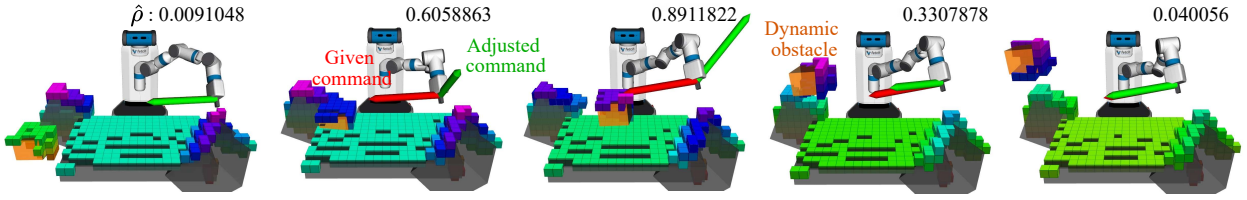
$\lvert\rho$ - $\hat{\rho}\rvert$	No obs.	Static obs.	Dynamic obs.	Both
Mean	3.85×10^{-3}	8.44×10^{-3}	5.89×10^{-2}	6.69×10^{-2}
Std	3.57×10^{-3}	3.21×10^{-2}	8.65×10^{-2}	1.03×10^{-1}

On the other hand, our method achieves a maximum success rate of 93% and a minimum of 74% throughout different velocity ranges of dynamic obstacles; the tested dynamic obstacles are all faster than the maximum velocity of the end-effector in the uniaxial direction, $0.33m/s$. These results indicate that our method can improve the collision avoidance rate by adjusting the given commands at risk of the collision to avoid obstacles in advance based on $\hat{\rho}$ (Fig. 6(a)), even though the success rate decreases as the velocity of dynamic obstacles increases.

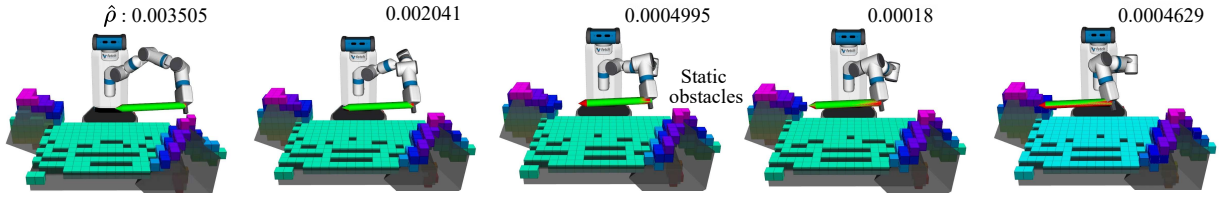
As shown in the Fig. 6, when the dynamic obstacle is moving away from the manipulator or absent, our method shows accurately following given commands by predicting a low $\hat{\rho}$. Consequently, the difference between Δx_u and Δx_a is less than 0.0001 on average in $1,000$ different static environments (Table I). This result is thanks to the RPN predicting a low $\hat{\rho}$ by robustly handling sensor noise (Table II).

For a more detailed analysis of the RPN, we prepare a validation set of $2,000$ groups each for different types of obstacles: no obstacles, only static obstacles, only dynamic obstacles, and both static and dynamic obstacles. Each group contains 16 consecutive joint positions, occupancy grids, and ρs calculated using Eq. 6; when there are no obstacles and only static obstacles, ρs are all zero. We measure the difference between ρ and $\hat{\rho}$ predicted by the RPN.

Table II shows the mean and standard deviation (std) of the difference in the validation set. It shows the largest mean difference of 0.067 and standard deviation of 0.1 in the environment including both static and dynamic obstacles. These are quite small numbers and indicate that the RPN

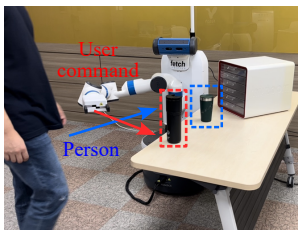


(a) Result in an environment with static and dynamic obstacles

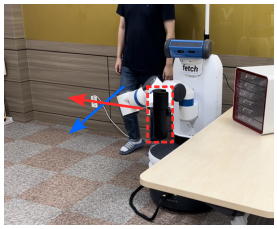


(b) Result in an environment with only static obstacles

Fig. 6. These figures show examples of our results in (a) an environment with static and dynamic obstacles and (b) an environment with only static obstacles. Orange and gray boxes are dynamic and static obstacles, respectively, and red and green arrows indicate the magnitude and direction of given command and adjusted command, respectively. The numbers in the upper right of figures are predicted risks for dynamic obstacles $\hat{\rho}$s. (a) shows that as the dynamic obstacle approaches to the manipulator, our method predicts a high $\hat{\rho}$ and adjusts the given command to avoid the obstacle. In contrast, as the dynamic obstacle moves away from the manipulator, our method follows the given commands, predicting a low $\hat{\rho}$. (b) shows that our method robustly copes with the noisy sensor data by predicting a near-zero $\hat{\rho}$ in a static environment.



(a) Moving to pick the water bottle



(b) Moving to transfer the water bottle

Fig. 7. These figures show tested problems in a real environment. Red arrow and blue arrow indicate the moving direction of the user commands and a person, respectively. In both problems, there is a risk of collision since the moving direction of the user commands and the person intersect. (a) is that the user gives commands to the robot to pick the black water bottle (red dotted box) and the person moves to pick the green water bottle (blue dotted box). (b) is that the user gives commands to the robot to transfer the water bottle and the person suddenly appears from behind the robot.

can predict a value close to ρ we model. Moreover, the low difference in the environment containing only static obstacles (0.008) supports that the RPN can robustly cope with noisy sensor data.

In conclusion, we demonstrate through various experiments that our method can increase safety from dynamic obstacles by adjusting a user command to avoid obstacles based on the risk prediction for dynamic obstacles.

B. Real robot test

We tested our method using the real fetch manipulator to verify the feasibility of our method in a real world. In our test environment, mimicking a remote manipulation task environment, the user observes the robot state and obstacles, and gives commands to the robot using a keyboard. For the observation, we installed a camera over the head to check a wide range and additionally used a camera mounted on the robot head (Fig. 1).

As shown in the Fig. 7, we tested our method in situations where the user transmits the commands with a risk of collision to the robot without recognizing the movement of a person, since the person suddenly appears in the camera's field of view. In these experiments, we showed that our method can predict the risk of dynamic obstacles from real sensing data and adjust the user commands at risk of collision in a safe direction. Our detailed experimental results can be seen in the attached video.

In addition, we measure the computation time of our method. Our OACN and RPN takes about 0.6*ms* and 1.0*ms*, respectively, and RCIK takes about 28.5*ms* on average. As a result, the computation time of our method is about 32*ms* on average including other computational processes, e.g., GPU allocation (1.5*ms*). Accordingly, our method can improve safety without much delay by adjusting the user's command with approximately 3*ms* additional time.

VI. Conclusion

In this paper, we proposed a risk-aware user command adjustment method to avoid the risk of dynamic obstacles for safe remote manipulation. Our method predicts the risk of dynamic obstacles and adjusts a user command to avoid collisions with obstacles according to the predicted risk. We showed the feasibility of our method towards safe remote manipulation through various experiments in simulation and real robot experiments. In future work, we would like to enhance the basis of risk judgment by adding semantic information about dynamic obstacles beyond considering the accessibility of dynamic obstacles using simple sensor data.

Acknowledgment

We appreciate anonymous reviewers' constructive comments. This work was supported in part by two projects (No. 2019R1A2C3002833 and No. 2021R1A4A1032582) from the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT).

REFERENCES

[1] Cai Meng, Tianmiao Wang, Wusheng Chou, Sheng Luan, Yuru Zhang, and Zengmin Tian, "Remote surgery case: robot-assisted teleneurosurgery", in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2004, vol. 1, pp. 819–823.

[2] Shinji Kawatsuma, Mineo Fukushima, and Takashi Okada, "Emergency response by robots to fukushima-daiichi accident: summary and lessons learned", *Industrial Robot: An International Journal*, 2012.

[3] Marcus Mast, Zdeněk Materna, Michal Španěl, Florian Weisshardt, Georg Arbeiter, Michael Burmester, Pavel Smrž, and Birgit Graf, "Semi-autonomous domestic service robots: Evaluation of a user interface for remote manipulation and navigation with focus on effects of stereoscopic display", *International Journal of Social Robotics*, vol. 7, no. 2, pp. 183–202, 2015.

[4] Zdeněk Materna, Michal Španěl, Marcus Mast, Vítězslav Beran, Florian Weisshardt, Michael Burmester, and Pavel Smrž, "Teleoperating assistive robots: a novel user interface relying on semi-autonomy and 3d environment mapping", *Journal of Robotics and Mechatronics*, vol. 29, no. 2, pp. 381–394, 2017.

[5] Eimei Oyama, Kohei Tokoi, Ryo Suzuki, Sousuke Nakamura, Naoji Shiroma, Norifumi Watanabe, Arvin Agah, Hiroyuki Okada, and Takashi Omori, "Augmented reality and mixed reality behavior navigation system for telexistence remote assistance", *Advanced Robotics*, vol. 35, no. 20, pp. 1223–1241, 2021.

[6] Daniel Rakita, Haochen Shi, Bilge Mutlu, and Michael Gleicher, "Collisionik: A per-instant pose optimization method for generating robot motions with environment collision avoidance", 2021.

[7] Mincheul Kang, Yoonki Cho, and Sung-Eui Yoon, "RCIK: Real-time collision-free inverse kinematics using a collision-cost prediction network", *IEEE Journal of Robotics and Automation (RA-L)*, vol. 7, no. 1, pp. 610–617, 2021.

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

[9] Jakob Thumm and Matthias Althoff, "Provably safe deep reinforcement learning for robotic manipulation in human environments", in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2022.

[10] Daniel Rakita, Bilge Mutlu, and Michael Gleicher, "STAMPEDE: A discrete-optimization method for solving pathwise-inverse kinematics", in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3507–3513.

[11] Rachel Holladay, Oren Salzman, and Siddhartha Srinivasa, "Minimizing task-space fréchet error via efficient incremental graph search", *IEEE Journal of Robotics and Automation (RA-L)*, vol. 4, no. 2, pp. 1999–2006, 2019.

[12] Mincheul Kang and Sung-Eui Yoon, "Analysis and acceleration of torm: optimization-based planning for path-wise inverse kinematics", *Autonomous Robots*, pp. 599–615, 2022.

[13] Rosen Diankov, *Automated construction of robotic manipulation program*, Ph.D. dissertation, Carnegie Mellon University, 2010.

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

[15] Daniel Rakita, Bilge Mutlu, and Michael Gleicher, "RelaxedIK: Real-time synthesis of accurate and feasible robot arm motion.", in *Robotics: Science and Systems*, 2018.

[16] Youngsun Kwon, Donghyuk Kim, Inkyu An, and Sung-eui Yoon, "Super rays and culling region for real-time updates on grid-based occupancy maps", *IEEE Transactions on Robotics (T-RO)*, vol. 35, no. 2, pp. 482–497, 2019.

[17] Baisravan HomChaudhuri, Lee Smith, Lydia Tapia, et al., "Safety, challenges, and performance of motion planners in dynamic environments", in *Robotics research*, pp. 793–808. Springer, 2020.

[18] Kris Hauser, "On responsiveness, safety, and completeness in real-time motion planning", *Autonomous Robots*, vol. 32, no. 1, pp. 35–48, 2012.

[19] Chonhyon Park, Jia Pan, and Dinesh Manocha, "Real-time optimization-based planning in dynamic environments using gpus", in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 4090–4097.

[20] Daniel Kappler, Franziska Meier, Jan Issac, Jim Mainprice, Cristina Garcia Cifuentes, Manuel Wüthrich, Vincent Berenz, Stefan Schaal, Nathan Ratliff, and Jeannette Bohg, "Real-time perception meets reactive motion generation", *IEEE Journal of Robotics and Automation (RA-L)*, vol. 3, no. 3, pp. 1864–1871, 2018.

[21] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn, "Universal planning networks: Learning generalizable representations for visuomotor control", in *International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 4732–4741.

[22] Mohamed El-Shamouty, Xinyang Wu, Shanqi Yang, Marcel Albus, and Marco F Huber, "Towards safe human-robot collaboration using deep reinforcement learning", in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4899–4905.

[23] Kei Ota, Devesh Jha, Tadashi Onishi, Asako Kanezaki, Yusuke Yoshiyasu, Yoko Sasaki, Toshisada Mariyama, and Daniel Nikovski, "Deep reactive planning in dynamic environments", in *Conference on Robot Learning (CoRL)*. PMLR, 2021, pp. 1943–1957.

[24] Tu-Hoa Pham, Giovanni De Magistris, and Ryuki Tachibana, "Optlayer-practical constrained optimization for deep reinforcement learning in the real world", in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6236–6243.

[25] Richard S Sutton and Andrew G Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[26] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al., "Soft actor-critic algorithms and applications", *International Conference on Machine Learning (ICML)*, 2018.

[27] Diederik P. Kingma and Max Welling, "Auto-encoding variational bayes", in *International Conference on Learning Representations (ICLR)*, 2014.

[28] Daniel Maturana and Sebastian Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition", in *IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2015, pp. 922–928.

[29] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling", in *NIPS 2014 Workshop on Deep Learning*, 2014.

[30] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee, "Decomposing motion and content for natural video sequence prediction", in *International Conference on Learning Representations (ICLR)*. ICLR, 2017.

[31] Dong Han, Hong Nie, Jinbao Chen, and Meng Chen, "Dynamic obstacle avoidance for manipulators using distance calculation and discrete detection", *Robotics and computer-integrated manufacturing*, vol. 49, pp. 98–104, 2018.

[32] Liangliang Zhao, Jingdong Zhao, and Hong Liu, "Solving the inverse kinematics problem of multiple redundant manipulators with collision avoidance in dynamic environments", *Journal of Intelligent & Robotic Systems*, vol. 101, no. 2, pp. 1–18, 2021.

[33] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning", in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2009, pp. 489–494.

[34] Nathan Koenig and Andrew Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator", in *IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2004, vol. 3, pp. 2149–2154.

[35] Diederick P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization", in *International Conference on Learning Representations (ICLR)*, 2015.