Robot obstacle avoidance system using deep reinforcement learning

Xiaojun Zhu
School of Automation, Beijing University of Posts and Telecommunications, Beijing, China
Yinghao Liang
School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
Hanxu Sun
School of Automation, Beijing University of Posts and Telecommunications, Beijing, China
Xueqian Wang
Center of Intelligent Control and Telescience, Tsinghua Shenzhen International Graduate School, Shenzhen, China, and
Bin Ren
School of Electronic Engineering and Intelligence, Dongguan University of Technology, Dongguan, China

Abstract

Purpose – Most manufacturing plants choose the easy way of completely separating human operators from robots to prevent accidents, but as a result, it dramatically affects the overall quality and speed that is expected from human–robot collaboration. It is not an easy task to ensure human safety when he/she has entered a robot's workspace, and the unstructured nature of those working environments makes it even harder. The purpose of this paper is to propose a real-time robot collision avoidance method to alleviate this problem.

Design/methodology/approach – In this paper, a model is trained to learn the direct control commands from the raw depth images through self-supervised reinforcement learning algorithm. To reduce the effect of sample inefficiency and safety during initial training, a virtual reality platform is used to simulate a natural working environment and generate obstacle avoidance data for training. To ensure a smooth transfer to a real robot, the automatic domain randomization technique is used to generate randomly distributed environmental parameters through the obstacle avoidance simulation of virtual robots in the virtual environment, contributing to better performance in the natural environment.

Findings – The method has been tested in both simulations with a real UR3 robot for several practical applications. The results of this paper indicate that the proposed approach can effectively make the robot safety-aware and learn how to divert its trajectory to avoid accidents with humans within the workspace.

Research limitations/implications – The method has been tested in both simulations with a real UR3 robot in several practical applications. The results indicate that the proposed approach can effectively make the robot be aware of safety and learn how to change its trajectory to avoid accidents with persons within the workspace.

Originality/value – This paper provides a novel collision avoidance framework that allows robots to work alongside human operators in unstructured and complex environments. The method uses end-to-end policy training to directly extract the optimal path from the visual inputs for the scene.

Keywords Robotics, Reinforcement learning, Obstacle avoidance, Transfer learning, Human–robot interaction

Paper type Research paper

1. Introduction

With the development of the industry and the shift of the manufacturing chain toward a broader personalized and customized system, we have to reconsider how to use robots in the supply chain. Human–robot collaboration is the inevitable choice for future development. Robots can help us in many ways (Gao *et al.*, 2021); persons and robots share a working space in the human–robot collaboration mode, and human experience is combined with the accuracy and reliability of robots to complete complex tasks. However, safety issues quickly arise, and for a moral purpose, the overall performance of robots is often sacrificed to ensure the safety of the workers. This paper aims to make human–robot collaboration safer by providing real-time detection of possible obstacles and the mechanism to allow robots to avoid collision with operators in their operational process.

Previous studies on collision avoidance often used RGB-D cameras to acquire data and perceive environments. Therefore, with the help of RGB-D sensors, a robot can carry out path

The current issue and full text archive of this journal is available on Emerald Insight at: **https://www.emerald.com/insight/0143-991X.htm**

Industrial Robot: the international journal of robotics research and application
49/2 (2022) 301–310

[DOI 10.1108/IR-06-2021-0127]

This work was supported Key-Area Research and Development Program of Guangdong Province(2020B010166006).

Received 2 April 2021
Revised 16 September 2021
9 October 2021
Accepted 10 October 2021

planning under constrained conditions. However, in these methods, obstacles are simplified to some major geometric elements, such as lines, points and balls. Although they work well in tests, they do not reflect actual working environments and, thus, have no concrete use in the industry (Yair *et al.*, 2016). Recently, some researchers have designed an end-to-end algorithm to directly predict a robot's obstacle avoidance trajectory from original RGB-D images. In the model's training process, either human demonstration is needed to collect the labeled data sets (Ravichandar and Dani, 2015) or reinforcement learning is used to design and automatically generate real obstacle avoidance track labels based on the designed cost function. At present, it has been proved that deep reinforcement learning is effective in solving complex control problems (Tobin *et al.*, 2017; Sergey *et al.*, 2016), but the methods used for exploring the environments during the learning process may cause damages to the natural environment and robot themselves. Simulation training, however, provides a new way to solve this problem. In recent years, with the development of the deep reinforcement learning and virtual technology, the gap between simulated environment and reality is bridged gradually. By randomizing the simulator's parameters in the training process, an optimal strategy can be obtained to adapt to the dynamics in training and extended to the dynamics in the real world without additional training on the physical system (Antonova *et al.*, 2017).

Although most collision avoidance studies have made remarkable achievements in improving collision detection efficiency, there still is room to improve:

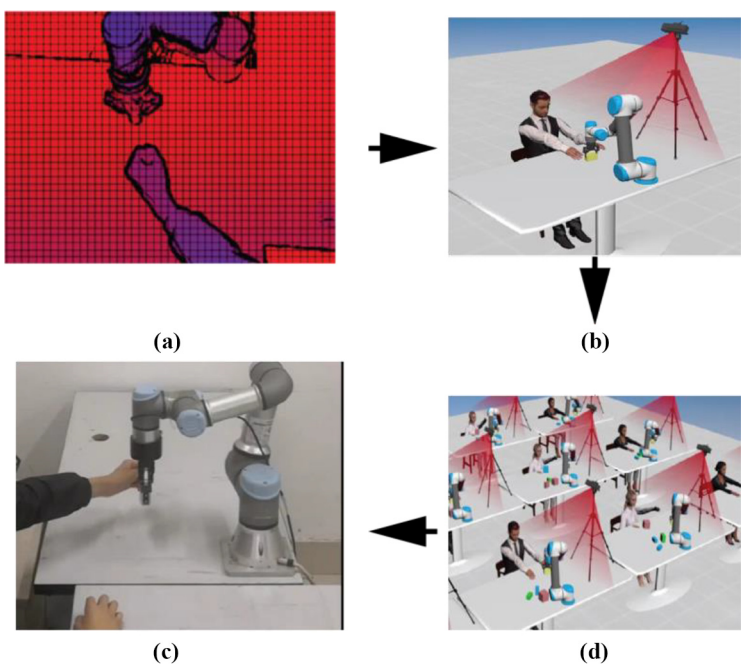
- Most obstacle avoidance algorithms, for example, in the collision avoidance research (Brito *et al.*, 2017; Mohammed *et al.*, 2017; Luo and Chung, 2015), tend to make a compromise between the optimality and completeness of planning and mostly rely on the detailed modeling of the manipulators and complex operation with formulas.
- When the visual method is used for obstacle avoidance, once other objects block human limbs, signals about their movements will not be obtained, and thus, obstacle avoidance fails.
- Research on obstacle avoidance based on neural network (NN) needs to collect a large amount of labeled data via human demonstration, for which NN-based methods are inclined to certain scenarios where demonstrations occur or specific tasks exist or autonomous learning-based one is easy to cause damages to the surrounding environment or robots.

To solve the abovementioned problems, an automatic robot obstacle avoidance system is proposed based on virtual training. First, actual collision avoidance scenarios are modeled in the simulation environment. Then the environmental parameters in the simulation are adjusted automatically with the automatic domain randomization (ADR) algorithm to generate enough diversified data, so that the model has strong generalization ability. Finally, virtual training of deep reinforcement learning is conducted. The well-trained control strategy may be directly transferred to the real world without additional fine tuning. The human–machine collaboration process is shown in Figure 1.

In the virtual environment, it is necessary to set a random range manually by use of the domain randomization (DR) algorithm to set the environmental parameters. If the random range is too large, then learning is complex; if too small, then it is challenging to realize the transfer from simulation to reality.

To solve the problems above, deep reinforcement learning is applied to the obstacle avoidance planning of robots, and an automatic obstacle avoidance system of robots is proposed based on virtual training in this paper. By mapping the

Figure 1 Teach robots how to avoid collisions for the purpose of human–machine cooperation



scenarios in the actual collision avoidance environment to the simulation environment, a large amount of virtual data is obtained, and then the designed obstacle avoidance NN can be trained. Because of the difficulty in accurately modeling the real-life scenarios in the simulation environment in terms of the location of the collision point, the collision angle and other physical properties, the ADR method is introduced into the virtual platform to automatically generate various environmental parameters in the proposed active obstacle avoidance system of robots. The ADR algorithm used in this research automatically adjusts the virtual environmental parameters, eliminates the gap between the virtual situation and the reality and makes the control strategy which can be directly transferred to the natural environment without additional fine tuning. The system framework is shown in Figure 2.

The main contributions of the proposed method include the following:

- In the proposed virtual obstacle avoidance training platform, no accurate modeling of the virtual environment is needed, which reduces the operators' burden. Besides, thanks to the end-to-end nature, the proposed network can directly derive action commands from depth images, which is advantageous in terms of effectiveness and training efficiency.
- A deep learning network is designed based on normalized advantage function (NAF) to predict the collision and the next strategy in real time and effectively predict the collision problem in case of blocking.
- A virtual training environment is introduced to tackle difficulties in sampling, insufficient data, low training efficiency and other problems. Also, the ADR algorithm is applied to solving the lack of data diversity and enables the model to possess sufficient generalization ability.

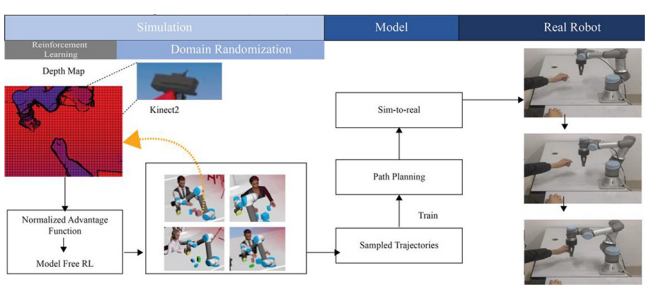
The remainder of this paper is organized as follows. Section 2 briefly introduces the related work in this field. Section 3 lays the foundation for our work by formulating the problems and explains in detail how we proceed to solve it. In Section 4, the experiments, results and discussion are introduced. Section 5 outlines the conclusions derived from our work.

2. Related work

Real-time active collision avoidance is mainly composed of three parts (Flacco *et al.*, 2012): 1) environment perception; 2) collision avoidance; and 3) robot control. Collision avoidance is one of the most important areas in robotics (Chen *et al.*, 2015). To detect obstacles directly and quickly and plan collision avoidance paths in advance, it is necessary to use various sensors to obtain the information on the surrounding environment. With the method introduced by Brito *et al.* (2017), the path planning of the UR3 manipulator is implemented based on the Kinect point cloud, to complete the placing and picking tasks while avoiding obstacles; with the method introduced by Mohammed *et al.* (2017), three-dimensional (3D) virtual models of robots and real images of human operators from depth cameras are used to achieve monitoring and collision detection; in some methods, the robot moves to a target position while avoiding obstacles by summing the repulsion vector of obstacles and the attraction vector of the target point of human–robot collaboration (Luo and Chung, 2015; Gai *et al.*, 2019). In the method introduced by Zhang *et al.* (2016), the depth camera is used to detect the relative position of the obstacle and the robot, and the artificial potential field method is used to generate the obstacle avoidance path. In the method introduced by Ahmad and Plappera (2016), Time of Flight and other sensors are used to locate obstacles and avoid obstacles through automatic path planning. In the method introduced by Kabutan *et al.* (2016), a novel sensor system for autonomous industrial robots was developed, which can measure 3D real-time information on the surroundings of the industrial vertical articulated robots for realizing object detection and path planning. To reduce damages to human body caused by possible accidental collisions, post-collision detection methods can be used to minimize the impact of collisions. Such methods limit the impact between humans and robots in a counteractive way, that is, detection after collisions, for example, is conducted through joint torque (Haddadin, 2014) or artificial skin (Mittendorfer *et al.*, 2015). Another method is to build intrinsically safe soft robots (Verl *et al.*, 2015). In the method introduced by Gao *et al.* (2020, 2021), an situational assessment method was proposed based on uncertainty risk analysis. Under uncertain conditions, the collision probability between multiple vehicles is estimated by comprehensive trajectory prediction according to the random environment model and Gaussian distribution model. In the method used by Golz *et al.* (2015), machine learning is used to identify interactive contacts and incidental collisions. Recently, researchers have designed a deep neural network (DNN) to evaluate whether a collision has occurred with high-dimensional signals from the robot joints (Heo *et al.*, 2019) or position signals of manipulator and signals from joint moment sensors (Sharkawy and Aspragathos, 2018) as necessary inputs. However, it is difficult to predict a collision which has not occurred and plan the path in advance in this way, and this may bring some safety risks to the operator when dealing with sharp objects or when the manipulator moves fast.

In the method introduced by Heo *et al.* (2019), an NN path planning algorithm was proposed based on reinforcement learning, to avoid obstacles in complex unknown environments. In the method used by Ravichandar and Dani (2015), a new way of shrinking the dynamic motion model in the learning state space was proposed. The learning model generates human demonstration data via Kinect and then the movement trajectory of robots. In the method used by

Figure 2 The system framework



Note: The training uses the model-free policy generator of normal ized advantage function to achieve obstacle avoidance in the real world

Rai *et al.* (2017), the coupling term functions are modeled via NN, which is trained by the human obstacle avoidance demonstration. Next, an active obstacle avoidance method was proposed based on deep reinforcement learning. This method uses an end-to-end obstacle avoidance network to generate obstacle avoidance commands for robots which are directly from the input depth images.

3. Problem formulation

In the proposed method, ADR is used to train the strategic model. In each case, the distribution in the environment is generated by randomizing some parameters, such as the mass of each connecting rod, damping, mass and friction of each joint and observed noise in the robot's body. DR requires human intervention to manually define the range of randomization distribution and keep it unchanged during the whole model training process. However, the range of randomization distribution is automatically defined and allowed to change in ADR.

ADR obtains the training strategy of the active obstacle avoidance model of robots by gradually expanding the distribution of model performance. This training strategy allows the robot's active obstacle avoidance model trained in the virtual environment to directly transfer to the real environment without distortion. In the training strategy of ADR, the initial distribution is concentrated in a single environment. For example, the initial environmental distribution is only for the rotational damping of the joint angle of the robot. ADR generates training data by sampling the environment and evaluating the model performance, independent of the algorithm used to train the model.

In ADR, each environment e_λ is parameterized by $\lambda \in R^d$, where d is the number of parameters that can be randomized in the simulation. In DR, the environmental parameter λ comes from the frozen-in distribution P_ϕ which is parameterized by $\lambda \in R^d$. However, in ADR, ϕ changes dynamically with the training progress. To quantify the expansion of ADR, the entropy of ADR is defined as $HP_\phi = -\frac{1}{d}\int P_\phi(\lambda)\log P_\phi(\lambda)d$. The higher the ADR entropy, the wider the random sampling distribution. Normalization allows us to compare different environmental parameter settings. In this respect, the factorial distribution parameterized by $d' = 2d$ is used. To simplify the definition, ϕ^L and $\phi_i^H \in \phi$ are defined as an exact partition of ϕ. For the ith ADR parameter λ_i $(i = 1,\ldots, d)$, (ϕ_i^L, ϕ_i^H) is used to describe a uniform distribution for sampling λ_i, so that there is $\lambda_i \sim U(\phi_i^L, \phi_i^H)$. It should be noted that the boundary values are inclusive here.

The overall distribution can be expressed as follows:

$$P_\phi(\lambda) = \prod_{i=1}^{d} U(\phi_i^L, \phi_i^H) \tag{1}$$

where, the ADR entropy is as follows:

$$HP_\phi = \frac{1}{d} P\sum_{i=1}^{d} \log(\phi_i^L - \phi_i^H) \tag{2}$$

In each iteration, the ADR algorithm randomly selects the dimensions λ_i of the environment and fixes it to a boundary value ϕ_i^L or ϕ_i^H, while other parameters are sampled according to P_ϕ. The model performance of the sampling environment is then evaluated and attached to the buffer range associated with the selected boundary of the selected parameter. Upon enough performance data are collected, they are averaged and compared with the threshold. If the average model performance is better than the high threshold t_H, then the parameters of the selected size will be increased; or if it is worse than the low threshold t_L, then the parameters of the selected size will be decreased.

3.1 Obstacle point cloud processing

In this research, the autoencoder (AE) network is used to embed the obstacle point cloud into the potential space. A kind of automatic encoders based on the study conducted by Heo *et al.* (2019) is used. Such an encoder consists of a convolutional layer and the maximum pooled layer following it, as shown in Table 1. Compared with the original structure of the AE network used in this research, the activation function in the layer before the maximum pooling layer is changed to the hyperbolic tangent and the activation function in the output layer to sigmoid.

In general, a point cloud represents a geometry, which is usually a set of 3D positions in an object's surface in a Euclidean coordinate system. In a 3D space, position information is usually defined with the x, y and z coordinate axes. Therefore, for an obstacle, the point cloud is usually a matrix of $N \times 3$, where N is the number of points and also known as the resolution of the point cloud.

In this research, an invariant permutation metric was used to compare orderless point sets. Earth mover's distance (EMD) (Rubner *et al.*, 2000), as a solution to transportation problems, attempts to transform one set into another. For two subsets of the same size, $S1 \subseteq R3$ and $S2 \subseteq R3$, their EMD is defined as follows:

$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2 \tag{3}$$

where, φ is a bijection.

The automatic encoder can greatly reduce the dimensional space of the point cloud, which is expressed as follows (Figure 3):

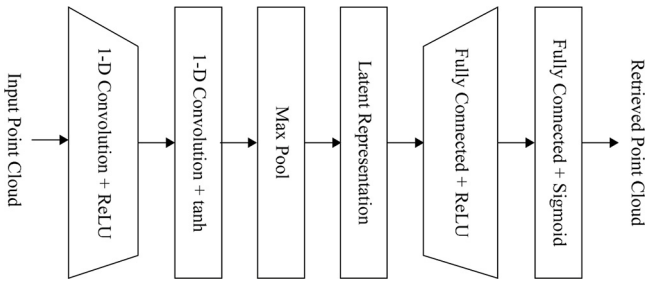
$$x_{enc} = Encoder(x_{obs}|W^e) \tag{4}$$

where: $x_{obs} \in R^{N\times 3}$ and $x_{enc} \in R^L$.

Table 1 Original architecture of the point cloud autoencoder

Layer	Activation	Features
1	ReLU	64
2	ReLU	128
3	ReLU	128
4	ReLU	256
5	Tanh	L_c
6	–	L
7	ReLU	256×3
8	ReLU	256×3
9	Sigmoid	$N \times 3$

Figure 3 Architecture of the modified Vanilla PC-AE architecture used for encoding the geometries



3.2 Basics of reinforcement learning

Reinforcement learning is realized with a machine-based learning framework that learns from interaction to achieve the desired goal. Learners and decision-makers are referred to as the agent and whichever interacts with the agent is referred to as the environment. In the process of continuous interaction between the agent and the environment, the agent chooses action according to certain strategies, and the environment responds to the action, making the agent enter into another new environment. Meanwhile, the environment gives rewards to the agent, and the agent tries to maximize those rewards over a period of time.

The process of reinforcement learning can often be represented as a Markov decision-making process that contains a set of four tuples $< S, A, P, R >$. At the time point t, the state of the agent is $S_t \in S$, where S is all possible states. $A_t \in A(S_t)$ is the action selected by the agent at the time point t, where $A(S_t)$ is all the actions that the agent can perform in the state S_t. At the next time point $t + 1$, the agent will receive a reward signal $R_{t-1} \in R$ for performing the action A_t, and meanwhile, the agent will enter a new state S_{t+1}. The cumulative rewards obtained by the agent in the state S_t are as follows:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{5}$$

where, (γ) is the discount coefficient of rewards, which is used to weigh the relationship between current rewards and long-term rewards. The greater the coefficient is, the more attention is paid to long-term rewards; otherwise, the more attention is paid to current rewards. The goal of reinforcement learning is to maximize the expected value of cumulative rewards from learning to strategy, that is:

$$\pi(a|s) = \arg\max_a E[R] \tag{6}$$

3.3 Normalized advantage functions

Deep deterministic policy gradient (DDPG) algorithm, as a model-free off-policy (DQN) reinforcement learning algorithm, uses a DNN for function approximation. Compared with DQN that can only solve the problems with a discrete and low-dimensional action space, DDPG can solve those with a continuous action space. Like DDPG, NAF also extends to deal with the problems with continuous motion. In the DQN framework, actions are selected by calculating the maximum value of Q. When it is extended to continuous control, a method should be designed to input the state and then output action and ensure the maximum Q value of output action.

One idea is to use two networks, among which one is the policy network for inputting state and outputting action and the other is the Q network for inputting state and outputting Q value. This is a typical AC structure for the DDPG method. Another idea is to prepare only one network that can output both action and Q value. That is also followed by the NAF method.

In 2017, DeepMind *et al.* proposed the simplest NAF model and a model-based NAF version with acceleration (Mohammed *et al.*, 2017). For DDPG, two networks have to be trained to achieve continuous control, while only one model for NAF. NAF makes use of the dueling net idea similar to dueling-DQN: Q value function can be expressed as the sum of the state value function V and dominance value function A.

As advantage, that is, the merits of each action, is introduced in a specific state, choosing the best action becomes choosing the action with the greatest advantages. As a result, there are the following definition:

$$Q(x, u|\theta^Q) = V(x|\theta^V) + A(x, u|\theta^A) \tag{7}$$

In this case, Q value is obtained by adding the state value function V to the action value function A:

$$A(x, u|\theta^A) = -\frac{1}{2}(u - \mu(x|\theta^u))^T P(x|\theta^P)(u - u(x|\theta^u)) \tag{8}$$

where, x presents state, u presents action, θ is the corresponding network parameter and function A can be regarded as the advantage of action u in the state x. The goal is to maximize the value of Q corresponding to the action u output by the network:

$$Q(s, a) = -(a - \mu(s))^T \sum(s)(a - \mu(s)) + V(s) \tag{9}$$

Where:

$$\mu(s) = \arg\max_a Q(s, a) \tag{10}$$

A batch of transitions s_t, a_t, r_t and s_{t+1} is sampled from the buffer. With the state S_t and the action a_t as its inputs, the new function Q will obtain the outputs $\mu(s_t)$ (vector), $\sum S_t$ (matrix) and $V(s_t)$ (scalar) according to the input state S_t. The construction of the state space S, action space A and reward function is described in detail as follows. The state space S is defined as follows:

$$S = \{q, \dot{q}, p_E, p_T, x_{enc}\} \tag{11}$$

where q represents the position of each joint of the industrial robot; $\dot{q}$ represents the angular velocity of each joint of the industrial robot; p_E represents the position of the end effector of the robot; p_T represents the position of the target point to be reached; and x_{enc} represents the point cloud image coding of the

obstacle. It should be noted that robots are usually equipped with resolvers for measuring positions only. However, in case of a high-quality mechanical design of robots, the velocity q can be well estimated via finite differentiation of the position signal. Otherwise, an estimate of the velocity q can be obtained by using observers.

It is assumed that the position of the end-effector of the robot is known and provided by a camera located in the workspace, as shown in Figure 4. Similar consideration also applies to the position of an obstacle, and it is assumed that its velocity is correctly estimated.

The action space is defined as follows:

$$A = \{\dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dot{q}_5, \dot{q}_6\} \tag{12}$$

where, $\dot{q}_k$ represents the rotatory angular velocity vector of the kth joint.

The reward function plays a guiding role in the whole learning process of the agent and evaluates the actions of the agent via the scalar. For the obstacle avoidance path planning of industrial robots, reinforcement learning aims to maximize the cumulative reward value obtained by the robot and find an optimal obstacle-free path from the starting point to the target point so as to finally complete the path planning properly. The reward function consists of two parts, that is, the distance between the end-effector and the target position and that between the obstacle and the robot in this research, as shown below:

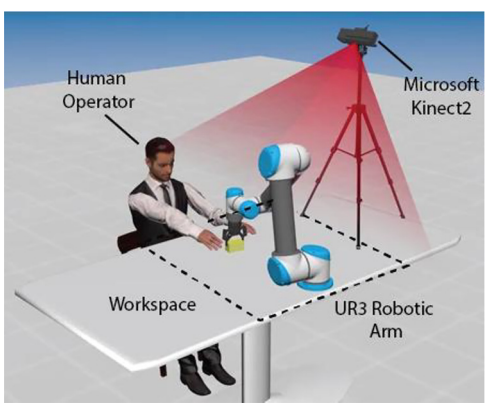
$$r = \lambda_1 R_T^E + \lambda_2 R_0^R + \lambda_3 R_A \tag{13}$$

Weights are used to fine tune the weight of each component in the reward function. Herein, the distance between the end-effector EE of the robot and the target point is determined by the Huber-loss function:

$$R_T^E = \begin{cases} \frac{1}{2}d^2 & |d| < \delta \\ \delta\left(|d| - \frac{1}{2}\delta\right) & otherwise \end{cases} \tag{14}$$

where, d represents the Euclidean distance, a parameter determining smoothness.

Figure 4 Working environment of robots



The distance between the robot end and the obstacle is expressed as follows:

$$R_0 = \left(\frac{d_C}{d_O^E + d_C}\right)^k \tag{15}$$

where, d_c represents a constant value and k means the exponential decay of negative rewards:

$$R_A = -\|a\|^2 \tag{16}$$

where, R_A represents the motion size. The robot is encouraged to make small action changes, so as to meet the safety requirements of the robot and human cooperation.

4. Experiments

To evaluate the effectiveness of the method presented in this paper, a series of experiments were carried out, from trivial situations to very tricky ones. The experiments aimed to answer the following questions: 1) Can robots develop safety-aware behaviors through self-supervised learning? 2) Can a robot adjust its trajectory in real time using just image data from RGB-D sensors? As the experiments involve humans working in proximity to a robotic manipulator, it is crucial, from an ethical standpoint, therefore, to follow the necessary safety protocols. In all the experiments, the joint's acceleration was decreased within MoveIt. Not allowing quick movements from the robot will leave enough time for the operator to get himself out of the way.

4.1 Experiment design

This experiment attempted to mimic an actual working environment where a robotic arm must work alongside a person. The experiment tasked the robot to reach a specific fixed spot with its end-effector while adding some obstacles along its trajectory. The difficulty level of this task ranges from easy to difficult depending on how the obstacle behaviors. The task is seen as easy when the obstacle is immobile; in this case, the robot can easily stop and replan its trajectory. A medium level of difficulty means those cases where the obstacle is moving at a slow pace, while the task is considered to be hardest when the obstacle is moving very fast. For all the experiments conducted, a straightforward method was used for evaluation. For each task, the robot can either fail or succeed. A failure means the robot approaches the obstacle at a distance lower than a certain threshold assigned for safety or when it could not replan its trajectory accurately. Any other situation where the target is reached while avoiding the obstacle constitutes a success.

The framework is trained within a simulated environment first to ease the problem of inefficient sampling with the reinforcement learning method and the safety issues that may arise when an untrained robot moves in the proximity of persons. In this work, the simulated experiments were done within Gazebo because of the fidelity of its physical engine. The setup is shown in Figure 4. The human operator sits in front of the robot, sharing the same working space with the robot. An RGB-D camera, Microsoft Kinect 2, is placed above the scene for overlooking the workspace at a 30° angle. The experiment

was completed with a UR3 manipulator. The connecting rod parameters of the D-H model for UR3 robots are shown in Table 2. The real experiment has a similar setup, but the robot operates slowly to protect the worker, as shown in Figure 1(d).

4.2 Expression ability of the autoencoder

First, the advantages of the proposed AE are demonstrated, and the generalization ability of the AE is reported by using the MMD-EMD metric.

Generalization ability: In this research, AEs are able to reconstruct invisible shapes with almost the same good quality as those used for training. As shown in Figure 5, AEs are used to encode the invisible samples from the test segmentation (left of each pair of images) and then decode them and visually compare them with the input (the right image).

Shape completion: The proposed AE architecture can be used to solve the problems in shape completion. Specifically, an incomplete expected output could be provided for the network instead of inputting and reconstructing the same point cloud.

4.3 RL training

Super-parameter settings: In deep reinforcement learning, super parameters are the values that must be set to adjust the learning environment of the model. In the process of reinforcement learning training, the formulation of strategies is mainly determined by the following values:

- *Noise type* △: This is the random process type that is added to every action taken by the agent to enable the agent to continuously explore the environment. This may be Brownian or other noise.
- *Noise scale*: This is the noise calibration factor in the noise detection process.
- *Noise loss factor*: This represents the speed of noise loss in the training iteration process.

Parameters related to the NAF learning algorithm are as follows:

- *Update factor* τ: This is the soft target update at the end of each iteration.
- *Loss factor* λ: This represents the value of future rewards compared with present rewards.
- *Yield* η: This represents how much new knowledge the agent can acquire.

Parameters related to the value function are as follows:

- λ_1, λ_2 *and* λ_3: They represent the distance to the target, the movement range and the distance to the obstacle, respectively.
- δ: This represents the discriminant parameter for Huber loss.
- *Reference distance* d_{ref}: This represents the minimum default distance between the robot skeleton and the obstacle.
- *Exponential decay factor* p: This represents the decay of negative reward when the distance between the robotic arm and the obstacle increases (Figure 6).

With the movement of obstacles and the position of target points as its variables, the algorithm's performance is evaluated under different settings. All the experiments used the same parameter sets, as shown in Table 3.

The settings considered for the experiment are shown as follows:

- *Fixed target and moving obstacle*: The target point's position is the same for each stage, and the obstacle moves from one end to the other of the linear path at a constant speed.
- *Fixed target and randomly moving obstacle*: The position of the target point is the same for each stage, and the obstacle moves randomly along the path and may change its direction or stop for a period of time at any time.

Table 2 Parameters of the kinematic model of a UR3 robot

Kinematics	θ [rad]	a [m]	d [m]	α [rad]
Joint 1	0	0	0.1519	$\pi/2$
Joint 2	0	−0.24365	0	0
Joint 3	0	−0.21325	0	0
Joint 4	0	0	0.11235	$\pi/2$
Joint 5	0	0	0.08535	$\pi/2$
Joint 6	0	0	0.0819	0

Figure 5 Point cloud data extracted by the RGB-D sensor

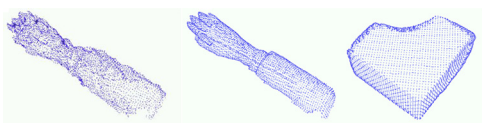


Figure 6 Accumulated rewards

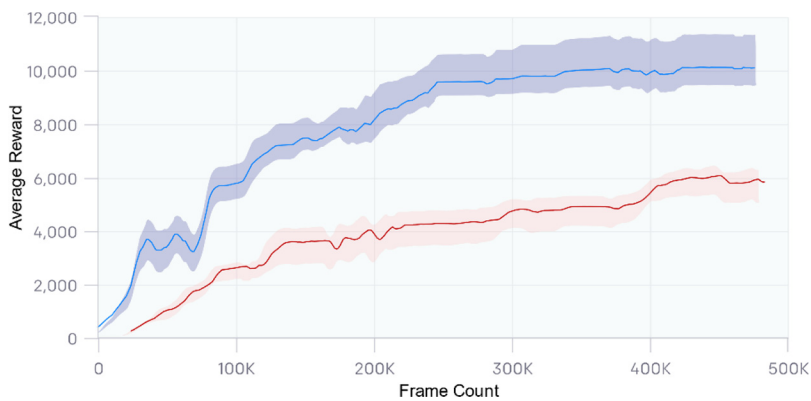


Table 3 Hyperparameters used to train the framework

Parameter	Value
Number of time steps	360
Time step	50 ms
λ_1	1,000
λ_2	100
λ_3	60
δ	0.1
p	8
d_{ref}	0.2
Discount factor λ	0.99
Update factor τ	0.001
Learning rate η	0.001
Noise type D	*Ornstrin-Uhlenbeck*
Noise decay factor	0.01
Noise scale	1

- *Random target and moving obstacle*: The target point's position is randomly initialized at the beginning of each stage, and the obstacle moves back and forth in a determined manner.
- *Random target and randomly moving obstacle*: The target point's position is randomly initialized at the beginning of each stage, and the obstacle moves randomly along the path.

4.4 Transfer to the natural environment

In the machine learning, transfer learning refers to the application of previously acquired knowledge to performing different tasks. After extensive trainings, the results of transfer learning have to be evaluated accordingly, for which three different approaches are considered:

- *Model transfer*: It means the reuse of approximator parameters during model initialization.
- *Experience transfer*: It means the reuse of the information included in the replay buffer. For example, the four tuples information $< S_t, A_t, R_t, S_{t+1} >$.
- *Transfer of models and experiences*: As described in the paragraphs (1) and (2) above.

4.5 Results

To show the usefulness of the method, it was tested both in a simulated environment and in a real-world setup. In the first part of the experiment, the goal was to assess the replanning process's accuracy and speed. In Table 4, the time required to reach the fixed point while avoiding the obstacle is reported. As we see, the robot could reach the point faster when the obstacle was immobile, both in the simulation and with the real robot. The real robot was relatively slower than the simulated one because of the decrease in the joint's acceleration for safety reasons. The comparison of robot trajectories in the virtual and real environments is shown in Figure 7.

During the simulated training, many aspects of the environments were randomized to ensure the smooth sim-to-real transfer and increase generalization ability. In the last experiments, the goal was to find how such a decision can affect the overall collision avoidance system.

First, the target was tried to an unreachable location. Then, the goal is to find out whether the robot would force its way if it cannot reach its objective. For this, a confidence score is generated, with 0 meaning the target cannot be reached and the robot has to give up and 1 meaning the target can be reached easily. In Figure 8, the x-axis denotes the distance from the

Table 4 Comparison of time between the simulated and real experiments

Motion	Real robot	Simulation
Static	3.12 s	4.89 s
Moving Slow	4.44 s	6.12 s
Moving Fast	5.57 s	7.24 s

Figure 7 Comparison of trajectories in the virtual and real environments

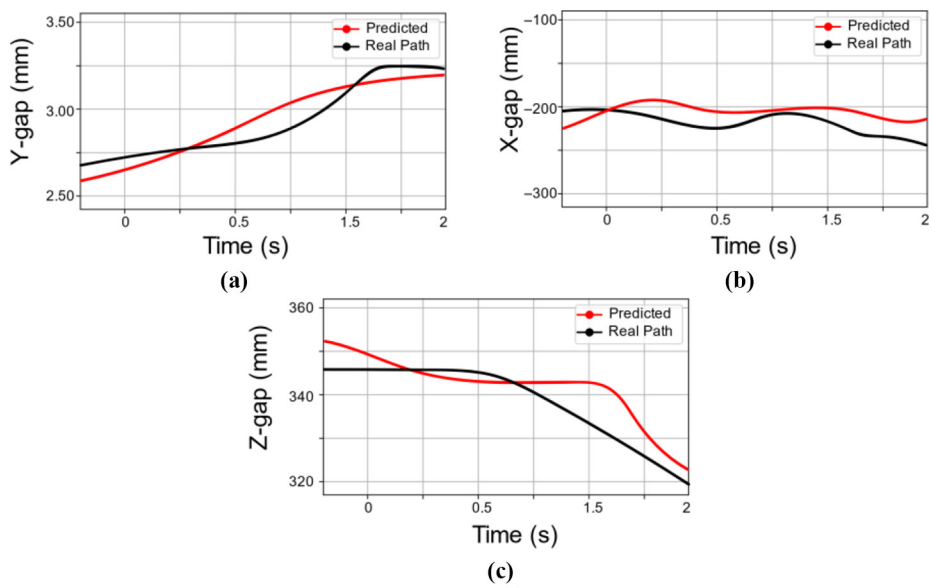
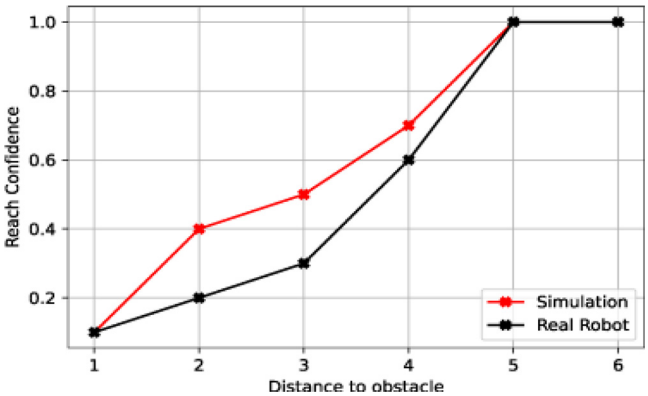


Figure 8 Confidence score on the possibility of reaching the target



obstacle to the target, and the *y*-axis represents the confidence score, ranging from 0 to 1. When the confidence score falls below 0.3, the robot gives up and does not try to reach the target any more.

5. Conclusion

This paper provides a novel collision avoidance framework that allows robots to work alongside human operators in unstructured and complex environments. The proposed method functions in real time, so the robot can quickly react and change its path to avoid possible collisions. The robot is trained end-to-end with raw images as its input to generate control commands that it modeled as NNs through self-supervised reinforcement learning. To reduce the effect of sample inefficiency and safety during initial training, a virtual reality platform was used to simulate a natural working environment. Furthermore, the ADR technique was used to generate randomly distributed obstacle situations and enable the framework to be more robust to any previously unseen situations and, therefore, smoothly transferred to a real robot. Finally, a set of experiments were carried out to test the framework's efficacity and practicability in both the simulation and the UR3 robot. The results indicate that robots can effectively learn to be aware of safety risks and change their paths accordingly in real time, so as to avoid collision and ensure the operators' safety even in unstructured and complex working environments.

Many questions still need to be fully answered before robots are completely integrated in our daily lives, among which the first priority is about safety concerns. We plan to extend this approach by increasing the robot's speed and efficiency in future work, without affecting the safety of the system. Also, we will integrate other techniques such as Kalman filtering and recurrent neural network to further increase the robot's efficiency in the face of moving obstacles.

References

Ahmad, R. and Plappera, P. (2016), "Safe and automated assembly process using vision assisted robot manipulator", *Procedia CIRP*, Vol. 41, pp. 771-776.

Antonova, R., Cruciani, S., Smith, C. and Kragic, D. (2017), "Reinforcement learning for pivoting task", CoRR, available at: arXiv:1703.00472.

Brito, T., Lima, J., Costa, P. and Piardi, L. (2017), "Dynamic collision avoidance system for a manipulator based on RGB-D data", *Iberian Robotics conference*, *Cham*, Springer, pp. 643-654.

Chen, P., Shan, C., Xiang, J. and Wei, W. (2015), "Moving obstacle avoidance for redundant manipulator via weighted least norm method", *The 27th Chinese Control and Decision Conference*, Qingdao, China, pp. 6181-6186.

Flacco, F., Krger, T., Luca, A. and Khatib, O. (2012), "A depth space approach to human-robot collision avoidance", *IEEE International Conference on Robotics and Automation*, *Saint Paul, MN*, pp. 338-345.

Gai, S.N., Sun, R., Chen, S.J. and Ji, S.T. (2019), "6-DOF robotic obstacle avoidance path planning based on artificial potential field method", *2019 16th International Conference on Ubiquitous Robots (UR)*, *Korea(South)*, pp. 165-168.

Gao, H., Kan, Z. and Li, K. (2021), "Robust lateral trajectory following control of unmanned vehicle based on model predictive control", *IEEE/ASME Transactions on Mechatronics*, pp. 1-1.

Gao, H., Luo, L., Pi, M., Li, Z., Li, Q., Zhao, K. and Huang, J. (2021), "EEG-Based volitional control of prosthetic legs for walking in different terrains", *IEEE Transactions on Automation Science and Engineering*, Vol. 18 No. 2, pp. 530-540.

Gao, H., Zhu, J., Zhang, T., Xie, G., Kan, Z., Hao, Z. and Liu, K. (2020), "Situational assessment for intelligent vehicles based on stochastic model and gaussian distributions in typical traffic scenarios", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1-11.

Golz, S., Osendorfer, C. and Haddadin, S. (2015), "Using tactile sensation for learning contact knowledge: discriminate collision from physical interaction", *IEEE International Conference on Robotics and Automation (ICRA)*, *Seattle, WA*, pp. 3788-3794.

Haddadin, S. (2014), *Towards Safe Robots: Approaching Asimov's 1st Law*, Heidelberg, Springer.

Heo, Y.J., Kim, D., Lee, W., Kim, H., Park, J. and Chung, W.K. (2019), "Collision detection for industrial collaborative robots: a deep learning approach", *IEEE Robotics and Automation Letters*, Vol. 4 No. 2, pp. 740-746.

Kabutan, R., Tanaka, R., Oomori, S., Morita, M., Inohira, E., Yoshida, K. and Nishida, T. (2016), "Development of robotic intelligent space using multiple RGB-D cameras for industrial robots", *Proc. of ICT-ROBOT 2016.*

Luo, R.C. and Chung, Y.T. (2015), "Dynamic multi-obstacles avoidance of a robot manipulator based on repulsive vector summation for human-robot co-works", *IEEE International Conference on Advanced Intelligent Mechatronics*, *Busan, Korea (South)*, pp. 1676-1681.

Mittendorfer, P., Yoshida, E. and Cheng, G. (2015), "Realizing whole-body tactile interactions with a self-organizing, multi-modal artificial skin on a humanoid robot", *Advanced Robotics*, Vol. 29 No. 1, pp. 51-67.

Mohammed, A., Schmidt, B. and Wang, L. (2017), "Active collision avoidance for human–robot collaboration driven by

vision sensors", *International Journal of Computer Integrated Manufacturing*, Vol. 30 No. 9, pp. 970-980.

Rai, A., Sutanto, G., Schaal, S. and Meier, F. (2017), "Learning feedback terms for reactive planning and control", *2017 IEEE International Conference on Robotics and Automation (ICRA)*, *Singapore*, pp. 2184-2191.

Ravichandar, H. and Dani, A. (2015), "Learning contracting nonlinear dynamics from human demonstration for robot motion planning", *ASME 2015 Dynamic Systems and Control Conference*, *American Society of Mechanical Engineers Digital Collection*, *Columbus*, *OH*.

Rubner, Y., Tomasi, C. and Guibas, L.J. (2000), "The earth mover's distance as a metric for image retrieval", *International Journal of Computer Vision*, Vol. 40 No. 2, pp. 99-121.

Sergey, L., Chelsea, F., Trevor, D. and Pieter, A. (2016), "End-to-end training of deep visuomotor policies", *Journal of Machine Learning Research*, Vol. 17 No. 39, pp. 1-40.

Sharkawy, A. and Aspragathos, N. (2018), "Human-robot collision detection based on neural networks", *International Journal of Mechanical Engineering and Robotics Research*, Vol. 7 No. 2, pp. 150-157.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W. and Abbeel, P. (2017), "Domain randomization for transferring deep neural networks from simulation to the real world", *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, *Vancouver*, *BC*, pp. 23-30.

Verl, A., Albu-Schäffer, A. and BrockO Raatz, A. (2015), *Soft Robotics: Transferring Theory to Application*, Springer, Berlin.

Yair, M., Takeo, K. and Yaser, S. (2016), "How useful is photo-realistic rendering for visual learning?", *European Conference on Computer Vision*, pp. 202-217.

Zhang, P., Jin, P.G., Du, G.L. and Liu, X. (2016), "Ensuring safety in human-robot coexisting environment based on two-level protection", *Industrial Robot: An International Journal*, Vol. 43 No. 3, pp. 264-273.

Corresponding author

Bin Ren can be contacted at: Renbinphd@163.com