RESEARCH ARTICLE

A real-time safe path planning system for cooperative robots

[version 1; peer review: awaiting peer review]

Yan He¹, Tao Lu², Yinghao Cai²

¹School of Artificial Intelligence, Chinese Academy of Sciences, Beijing, 100098, China
²State Key Laboratory of Management and Control for Complex Systems, Chinese Academy of Sciences Institute of Automation, Beijing, 100053, China

First published: 11 May 2022, 1:12
https://doi.org/10.12688/cobot.17505.1
Latest published: 11 May 2022, 1:12
https://doi.org/10.12688/cobot.17505.1

Abstract
Background: A cooperative robot is a robot requested to co-work with humans efficiently and safely in an environment with flexible arrangements. Safe path planning is a crucial issue which must be resolved during human-robot cooperation. In this paper, we present a safe path planning system that could plan the manipulation path in real-time based on the environmental changes and guarantees safety when the robot interacts with the environment and humans.

Methods: In this system, we first build a real-time obstacle Octomap from the environment RGB-D (red green blue-depth) images, which can effectively differentiate the robot from other obstacles in the environment and eliminate the robots influence during the map building. And then, we adopt the rapidly exploring random trees-Connect method to plan the safe path in the Octomap. When the planning path is obstructed by the dynamic objects, the system will re-plan the new safe path based on the changed Octomap.

Results: The experimental results show that our system can effectively avoid obstacles in a dynamic environment and safely reach the manipulation destination.

Conclusions: We propose a real-time safe path planning system for cooperative robots, which can guarantee the safety of manipulation.

Keywords
Cooperative Robot, Safe Path Planning, Octomap, RRT-Connect
Introduction
A cooperative robot is designed to work with humans in the same working space. At present, most cooperative robots guarantee safety by stopping when they contact unanticipated objects or humans during manipulation. The sensors which are used to perceive these objects, include joint torque sensors and skin pressure sensors. These sensors are now relatively expensive and also have to be exposed to obstacles\(^1,2\). Visual sensors could perceive the changes in the environment and predict the dangers before the contactation between the robot and other objects\(^3\). Early research has been limited by camera performance, and the monocular cameras used as visual sensors. A notable drawback of the monocular camera is it can not obtain the depth information of the environment. Tan et al. proposed a visual system to capture the motion\(^4\) of the sitting operator (upper body only) by having the operator wear a colour mark. This method relies on colour consistency and is not suitable for uneven ambient lighting conditions.

Detection of depth information based on binocular or depth camera is a more commonly used environmental perception method. Schiavi\(^5\) and Fischer\(^6\) respectively put forward the method of obstacle detection based on 3D depth information collected by a time of flight (ToF) camera and light detection and ranging (LiDAR). ToF cameras provide high-performance solutions, but pixel resolution level is insufficient for deep image acquisition. To address these challenges, we propose a safe path planning system with an RGB-D (red green blue-depth) camera to avoid obstacles in real-time. After obtaining the spatial coordinates of each sampling Point on the object surface, a collection of points is obtained, which is called "point cloud". When you start the Kinect V2 driver, point cloud information is published in real time via the /kinectV2/qhd/points topic. By converting the point cloud into an Octomap\(^7\), the proposed system plans a safe path based on the rapidly exploring random trees (RRT)-Connect method\(^8\) in the Octomap. The whole system is built under the ROS (Robot Operating System) framework.

System details
In this section, we introduce the path planning system in detail, as shown in Figure 1.

Firstly, we use Octomap to build a 3D obstacle map of the environment. Octomap\(^7\) is an efficient probabilistic 3D mapping framework based on octrees in which each node of octrees represents whether the current space of the cube is occupied by an obstacle. We obtain point cloud information through an RGB-D camera and turn it into an Octomap of the environment as shown in Figure 2a. In this map, the robot area is also interpreted as an obstacle. Therefore, it is necessary to filter out the points that represent the robot. To address this problem, we adopt the method of enveloping the ball based on the robot model. Set spheres at specific intervals in the URDF(Unified Robot Description Format) model of the robotic arm, so that these spheres can wrap around the robot model. The Octomap when the robot has been filtered out is shown in Figure 2b.

Secondly, we customize the motion planning algorithm\(^9\) based on RRT-Connect\(^6\) through MoveIt in Robot Operating System (ROS). The critical problem of path planning is to avoid collisions. Let \( C \) be the configuration space of the robotic arm, then \( \mathbb{C}^n \), where \( n \) is the degrees of freedom of the robot. For a robotic arm, a group of joint configurations corresponds to a robotic arm’s pose state, which is expressed as \( q = [\theta_1, \theta_2, ..., \theta_n] \), \( \theta_i \sim \theta_{i+1} \) represents the position of the \( n \) joints. The configuration space \( C \) can be divided into a free configuration space \( C_{\text{free}} \) that does not overlap with the robotic arm configuration \( q \) and an obstacle configuration space \( C_{\text{ob}} \) that overlaps with \( q \). These configurations are complementary. Therefore, the path planning problem can be described as: Find a set of continuous configuration \( P \) in the free configuration space such that \( P \in C_{\text{free}} \), the start point is the initial configuration \( q_{\text{init}} \) and the end point is the target configuration \( q_{\text{goal}} \). The obtained \( P \) is the collision-free feasible path of the robot. The characteristic of robotic arm path planning is that it needs to search for paths in a high-dimensional joint space, which often leads to a large amount of calculation. Here we adopt the RRT-Connect path planning method. It can quickly and effectively search the state space. Algorithm 1 shows the RRT-Connect algorithm flow.
The path planning strategy of RRT-Connect is as shown in Figure 3. RRT-Connect begins by initializing the two subtrees $I_{init}$ and $G_{init}$ from the start point $q_{init}$ and goal point $q_{goal}$. The kinodynamic planning of the two subtrees extending was suggested in 11. To eliminate the randomness of the extending, we first select a random point to label as $q_{rand}$ in the configuration space, and find the nearest point ($q_{near}$) in the subtree $I_{init}$. From $q_{near}$, we extend a short distance toward $q_{near}$ to get a new point ($q_{new}$). If there is no collision in this process, add $q_{new}$ to $I_{init}$, otherwise delete the $q_{new}$ and continue to sample the random point until a new point ($q_{near}$) is found. After this process has been found, find the nearest point $q_{near}$ to $q_{new}$ in the subtree $G_{init}$. From $q_{near}$ extend a short distance toward $q_{near}$ to get a new point as $q_{new}$. If a collision does not happen in this process, add $q_{new}$ to $G_{init}$, otherwise delete the $q_{near}$. Repeat the above process until $q_{new}$ equal to $q_{new}$. Then the path is found.

### Experiments and results

Our system was tested in an environment with dynamic obstacles. The experiments were conducted using real hardware: the UR3 robotic arm (Universal robots), one Kinect V2 (Microsoft, 2014). And Algorithm 1 are all running in the ROS framework.

Building real-time obstacle scene map based on Octomap

We first use the Kinect V2 to obtain the obstacle depth information, and then converting the obstacle depth information into the point cloud information which would be published as a ROS topic to the Octomap module. In the Octomap module, the octomap is built based on the point cloud data. The built Octomap in three different environments is shown in Figure 4.[12]

Experiment of global autonomous path planning

The whole system is verified by planning and executing a trajectory between two designated points, A and B in a work space. In the first experiment we don’t place any obstacles, letting the robotic arm move from point A to point B to test the capacity of our method in planning trajectories. In the second scenario we set an obstacle (randomly) between A and B (arbitrarily selected points) and let the robotic arm move from point A to point B to test the obstacle avoidance effect of our algorithm. In the final experiment we changed the obstacle position to test the effect of real-time obstacle avoidance. When the Octomap is updated based on the detected changes in environment, the RRT-connect algorithm is called to re-plan the obstacle-free path. The path-planning results in simulation are shown in Figure 5. The execution process of the planned safe path in real scene is shown in Figure 6.

---

**Algorithm 1. rapidly exploring random tree (RRT)-Connect algorithm**

```
1: $I_{init}(q_{init})$, $G_{init}(q_{goal})$
2: for $k = 1$ to $N$ do
3:   $q_{rand} \leftarrow$ Sample($k$), $k \leftarrow k + 1$
4:   $q_{near} \leftarrow$ Nearest($I$, $q_{rand}$)
5:   $q_{near} \leftarrow$ Extend($q_{rand}$, $q_{near}$)
6:   if Collision_Free($q_{near}$, $q_{near}$) then
7:     AddNew($q_{near}$)
8:   $q_{near} \leftarrow$ Nearest($G$, $q_{near}$)
9:   $q_{new} \leftarrow$ Extend($q_{near}$, $q_{new}$)
10: if Collision_Free($q_{new}$, $q_{new}$) then
11:   AddNew($q_{new}$)
12: else
13:   Delete($q_{near}$)
14:   Break;
15: end if
16: else
17:   Delete($q_{near}$)
18:   Break;
19: end if
20: if Connect($q_{new}$, $q_{new}$) then
21:   Return ($I$, $G$)
22: end if
23: if $|G|<|I|$ then
24:   Swap($I$, $G$)
25: end if
26: end for
27: Return Failure;
```
data associated with the experiments is available in Underlying data.

Conclusions
In this paper, we present a safe path planning system that could plan a path in real-time based on environmental changes and guarantee safety when the robot interacts with the environment and humans. Through the real-time Octomap, the system re-plans the safe path according to the changes in position of any obstacles. The experimental results show the effectiveness and safety of our system in a dynamic environment.

Data availability
Underlying data
Figshare: Data of A Real-time Safe Path Planning System For Cooperative Robot.
https://doi.org/10.6084/m9.figshare.19158710.v3
This project contains the following underlying data:
- Trajectory.script (The safe path obtained by the algorithm).
- RRTConnect.cpp (The path planning algorithm).
- Octomap.zip (Build Octomap).
- Data.zip(RGB-D data and point-cloud data).
  - cloud0.pcd (point-cloud data with one obstacle)
  - cloud1.pcd (point-cloud data with two obstacles)
  - cloud2.pcd (point-cloud data with three obstacles)
  - RGB-D0.png (RGB-D data with one obstacle)
  - RGB-D1.png (RGB-D data with two obstacle)
  - RGB-D2.png (RGB-D data with three obstacle)

Extended data
Figshare: Data of the result for path planning. https://doi.org/10.6084/m9.figshare.19158710.v3
This project contains the following extended data:
- Path Planning.mp4 (Video showing the results of path planning in a real situation).

Data are available under the terms of the Creative Commons Attribution 4.0 International license (CC-BY 4.0).

Software availability
- Source code available from: https://github.com/thinking-ASI/path-plannning.git
- Archived source code at time of publication: https://doi.org/10.5281/zenodo.590590
- License: Creative Commons Attribution 4.0 International license (CC-BY 4.0)

References
   Publisher Full Text
   Reference Source
   Publisher Full Text
   Publisher Full Text
   Publisher Full Text
6. Fischer M, Henrich D: 3d collision detection for industrial robots and unknown obstacles using multiple depth images. In Advances in robotics

Publisher Full Text

Publisher Full Text

Publisher Full Text

Publisher Full Text


Publisher Full Text