Obstacle avoidance for robotic manipulator using Mixed reality glasses

A. Evlampev, M. Ostanin
Center for Technologies in Robotics and Mechatronics Components, Innopolis University, Innopolis, Russia
a.evlampev, m.ostanin@innopolis.ru

***Abstract*—The problem of finding the shortest path for manipulators with obstacle avoidance has many solutions. The paper presents the usage of well-known methods for solving that problem using mixed reality glasses. Glasses contain the Spatial mapping module, presented in the form of an continuously updating mesh. Analyzing space through mixed reality glasses, we have developed algorithms for obtaining the shortest path in the Cartesian and Joint spaces. In the beginning, we implement A* simple algorithm and then realized the improved RRT algorithm. The approaches were verified using Unity, Microsoft HoloLens and 6 DOF robot UR10e.**

***Index Terms*—Shortest path; obstacle avoidance; robotic manipulator; path planning; mixed reality; RRT algorithm;**

I. Introduction

Many real-world tasks, ranging from opening doors and pushing carts to align beams for construction and removing rubble, exhibit workspace constraints. In these circumstances, the robot must not only preserve the task constraint but also detect obstacles, avoid collisions and joint limits throughout a planned motion [1]. Jen-Hao Chen and Kai-Tai Song showed with practical experiments that the 6-DOF robot arm can effectively avoid an obstacle in a constrained environment and complete the original task [2].

Recently a lot of research was done in the field of collaborative robotics [3]. Collaborative robots are working with a human in dynamic unstructured environment. Path-planning becomes an important problem since different obstacles can exist in the workspace of robot and robot path is required to obey workspace constraints. Obstacles are defined as any portion of an object with which contact is undesirable [4].

Detecting obstacles can be done with Mixed reality (MR) glasses. MR is an incorporated of the real world and virtual environment [5]. MR glasses provide a direct connection between 2 worlds. In our research, we use Microsoft HoloLens holographic device. HoloLens has Spatial mapping module that represents the real environment, in other words, obstacles.

In recent years there have been many developments in field MR and Robotic. In 2015, Hoenig et al. [6] defined benefits of using MR in various applications of robotics. Other researches proposed the system for interactive programming of industrial robots based on Mixed Reality [7]–[9]. We use this kind of systems to set tasks for the robot [7].

The work presented in this paper was supported by the grant of Russian Science Foundation 17-19-01740.

The goal task is to find a shortest path with obstacle avoidance. It is a problem of a graph theory and it has a lot of solutions. In our research we used the A* for find path in Cartesian space and Rapidly-exploring Random Tree (RRT) for robot Joint space. RRT is a randomized algorithm that used to perform path-planning with obstacle avoidance [10] [11] that have been widely used in autonomous robotic motion planning. In our research we implemented this algorithm to work with mixed reality and discovered efficiency of this approach. Details of RRT implementation well described in paper Wei K, Ren B [12]. They tested algorithm with UR robot and used Kinect as main sensor.

II. System overview

System consists of 3 main parts: mixed reality glasses, robot controller and robot. We use HoloLens as example of MR glasses. Robot controller based on ROS realised a connection between glasses and robot. MR glasses application has some modules: Interface, Path Planner, Spatial mapping and Robot model. External computer runs ROS Kinetic and controls robot. Controller receives commands from HoloLens, transforms them in required format and sends to robot.

Fig. 1. Mixed reality based robots control system architecture

Application for HoloLens was developed in Unity 2018 with help of Mixed Reality Toolkit for Unity. Integrated part of HoloLens application is the Spatial mapping which geometrically represents a real environment. Core of the system is Path planner module, because it generates path with obstacles avoidance. For connection to ROS we used Ros-Sharp library for UWP (Universal Windows Platform) and Rosbridge.

Interaction script is following: user specifies start and final points. The path that satisfies all constraint is generated between these points. After that user can run virtual simulation or execute path on real robot.

Path planning can be performed in 2 coordinate systems: Cartesian space and Joint space. For each of it, we have implemented own path construction algorithm. To find the

978-1-7281-2470-4/19/$31.00 ©2019 IEEE

shortest path in the Cartesian coordinate system, we used the algorithm A*. We used RRT to find the path in the joints space. Let's consider it in more detail.

III. Shortest path in Cartesian space

Finding the shortest path in Cartesian space is necessary to simplify and speed up the process of manipulator programming since the user only sets the start and final position of the tool. Such a method will be especially useful for pick and place operations. We realize a A* algorithm for finding shortest path in 3D space. It was implemented in Unity/C#.

Before starting the search for a path, it is necessary to tune the system, namely, indicate the model of the manipulator and determine the position of the base operation. Determining the position and orientation of the robot for such systems is described in detail in [13]. Below the pseudo code of the path finding algorithm is presented.

Input: $isObstacleCollider, ControlPoints, CellSize,$ *Workspace Radius*
Output: $Path$
Initialisation :
$Path \leftarrow \emptyset$
$N \leftarrow WorkspaceRadius * 2/Cellsize$
Fill environment map
for *every cell in 3D area* **do**
  if *CurrentCell inside workspace and isObstacleCollider(CurrentCell)* **then**
    $EnvMap(CurrentCell) \leftarrow 1$
  else
    $EnvMap(CurrentCell) \leftarrow -1$
  end if
end for
Build path
for $i = 0$ to $size(ControlPoints) - 1$ **do**
  $LocalPath \leftarrow Astar(ControlPoints[i], ControlPoints(i + 1))$
  $Path \leftarrow Path + LocalPath$
end for
return $Path$

For each robot model we know the robot work-space, therefor in the beginning we divided it into small cells. After that each cell checked by Unity tools whether it contains obstacle or not, results collected into boolean array. Finally, A* algorithm runs.

We verified our algorithm based on the Microsoft HoloLens and the UR10e robotic manipulator. By the HoloLens interface we set 2 point in different sides of the wall and build a shortest path. The result of the work is represented by the figure 2.

IV. Shortest path in Joint space

Shortest path in Cartesian space is helpful in the robot programming process, but it has the main drawback - corresponding method cannot detect a collision of robot links with obstacles. Therefore we developed the path planning algorithm which takes to account a robot structure. Our algorithm is based on RRT and works in joint space.

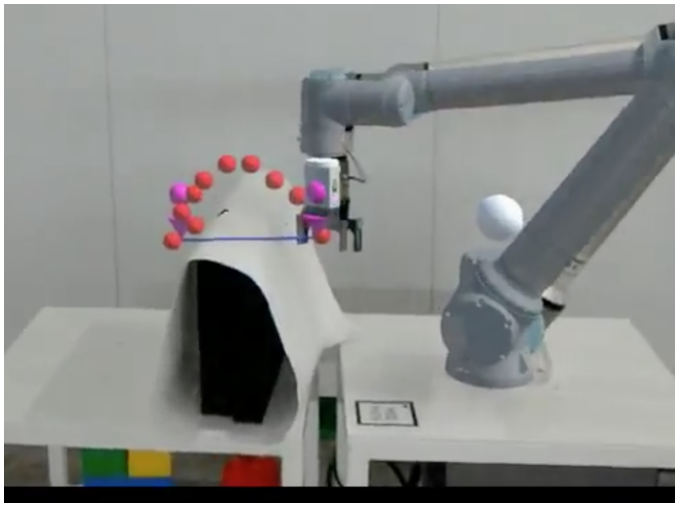

Fig. 2. Path for end-effector created by A* algorithm

The basic idea of the RRT algorithm is to explore space in the form of the tree from the initial state of the manipulator and to generated samples randomly to extend the branches and leaves until the exploring tree covers the target region. Search performed for first 3 joints because a motion space of last 3 joints we can define as a part of third link. RRT search can be performed in high-dimension spaces but our assumption help us to minimize the search space. Start and finish points should be specified with joint angles. Pseudo code of developed algorithm is presented bellow.

Input: $qInit$
Output: $Tree$
$initializeTreeRoot(Tree, qInit)$
while not reached goal **do**
  $qRand \leftarrow RandomState()$
  $Extend(Tree, qRand)$
end while

Each new node should be reached without collisions. To verify this, robot motion is simulated each time between old and new configuration. Unity has methods for collision detection. Also, we implemented forward kinematics for the robot to know the position of links.

RRT method should know how far one point from another to find the nearest neighbor. We used Euclidean vector norm to handle this.

After reaching the goal point it is easy to reconstruct the path because each node has a link to its parent. A resulting path generated by RRT contains many redundant nodes. We remove them by applying pruning technique that described in [12].

We verified our algorithm based on the Microsoft HoloLens, Unity Editor and the UR10e robotic manipulator. The results presented in figure 3.

V. Algorithms Comparison

To compare A* based and RRT based path planning algorithms we had run them in simulation. A virtual environment

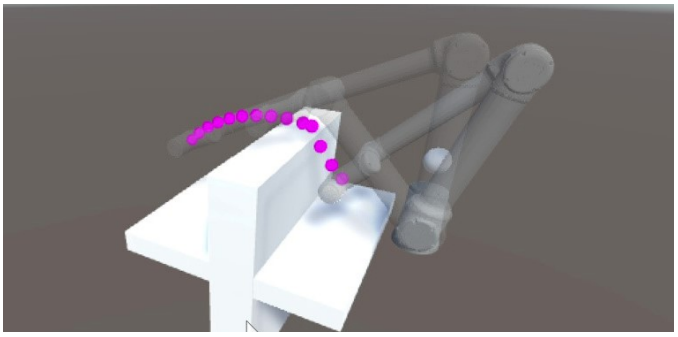

Fig. 3. One of paths generated by RRT algorithm

is a Unity scene that contains room mesh and UR10e model. Robot should avoid contact with mesh. All experiments were run on MacBook Pro 2014 with 2.6 GHz Intel Core i5 CPU, OS Windows. Microsoft Hololens has weaker cpu, 1,04 GHz Intel Atom.

In figure 4 we can see paths generated by both algorithms. Pink balls represent the position of robot end-effector for path generated by RRT and white balls represent A* path. With this configuration of initial and final points, we had run 10 experiments with RRT. Average time of execution was 200ms. It had some variation because the algorithm relies on randomness. The fastest search was done in 83ms with 560 iterations.

Most time-consuming operations for RRT are collision detection and nearest neighbor search (NNS). Generally, NNS search took 10 times more CPU time than collision detection. NNS search is so slow because it just linearly checks all nodes. It can be optimized in further research.

A* algorithm is more stable and in most experiments, it completed in 400ms. As we mentioned before it builds an internal 3D grid to detect obstacles and free spaces. Grid size is 30 and totally it does 27000 calls to check collisions. The path generated by A* tends to be straight in Cartesian space, it should be smoothed.

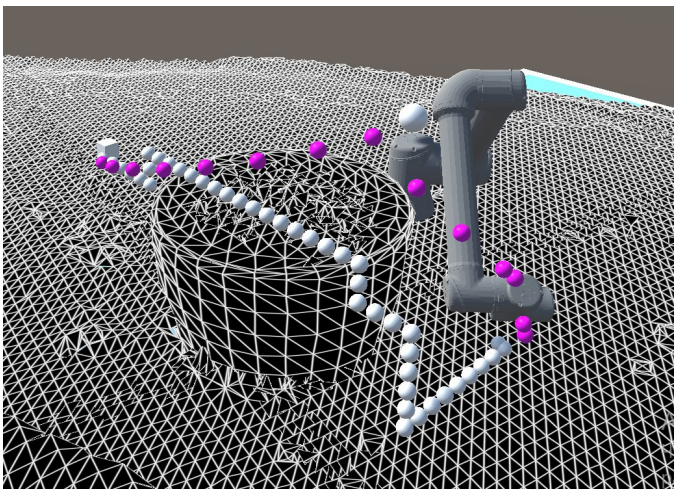

Fig. 4. Both algorithms in virtual environment with mesh

VI. Conclusion

This paper proposes a collision-free path planning for robotic manipulators with using mixed reality and describes 2 methods with different approaches.

Experiments showed that joint space space method based on RRT works faster than method based on A* and also generated trajectory with arbitrary shape. Searching for nearest neighbor is most time-consuming path here. Method works in joint space, however inverse kinematics still needed to provide initial and final state.

Cartesian space method gives path for end-effector without considering robot body so it does not depend on robot structure. But additional effort needed to do inverse kinematics for all points. We used A* algorithm which is deterministic and result always can be repeated in the exact same way.

References

[1] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 576–584, 2010.

[2] J.-H. Chen and K.-T. Song, "Collision-free motion planning for human-robot collaborative safety under cartesian constraint," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–7, 2018.

[3] L. W. Abdullah Mohamme, Bernard Schmidt, "Active collision avoidance for humanrobot collaboration driven by vision sensors," *International Journal of Computer Integrated Manufacturing*, vol. 30, no. 9, pp. 970–980, 2017.

[4] A. A. Maciejewski and C. A. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *International Journal of Robotics Research*, vol. 4, no. 3, pp. 109–117, 1985.

[5] P. Milgram and F. Kishino, "A taxonomy of mixed reality visual displays," *IEICE Trans. Information Systems*, vol. vol. E77-D, no. 12, pp. 1321–1329, 12 1994.

[6] W. Hoenig, C. Milanes, L. Scaria, T. Phan, M. Bolas, and N. Ayanian, "Mixed reality for robotics," pp. 5382–5387, 2015.

[7] M. Ostanin and A. Klimchik, "Interactive robot programing using mixed reality," *IFAC-PapersOnLine*, vol. 51, pp. 50–55, 01 2018.

[8] J. Guhl, S. Tung, and J. Kruger, "Concept and architecture for programming industrial robots using augmented reality with mobile devices like microsoft hololens," pp. 1–4, 2017.

[9] J. W. S. Chong, S. Ong, A. Y. Nee, and K. Youcef-Youmi, "Robot programming using augmented reality: An interactive method for planning collision-free paths," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 3, pp. 689–701, 2009.

[10] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[11] J. J. LaValle, Steven M.; Kuffner Jr., "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[12] R. B. Wei K, "A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved rrt algorithm," 2018.

[13] M. Ostanin, R. Yagfarov, and A. Klimchik, "Interactive robot programing using mixed reality," *ACCEPTED, Manufacturing Modelling, Management and Control - 9th MIM 2019*.