Real-time Distance Query and Collision Avoidance for Point Clouds with Heavy-duty Redundant Manipulator

Tuomo Kivelä, Pauli Mustalahti, and Jouni Mattila
Laboratory of Automation and Hydraulics
Tampere University of Technology
33720 Tampere, Finland
Emails: tuomo.kivela@tut.fi,
pauli.mustalahti@tut.fi, jouni.mattila@tut.fi

***Abstract*—This paper presents a real-time method for generating joint trajectories for redundant manipulators with collision avoidance capability. The coordinated motion control system of the heavy-duty hydraulic manipulator resolves joint references so that a goal position can be reached in real-time without any collisions. The proposed method is able to detect and prevent different types of possible collisions, including self-collisions and collisions with obstacles. When the control system detects the risk of collision, the collision server searches the points where the collision is about to occur and calculates the shortest distance between the colliding objects. The collision server is used to retain static point clouds and to calculate the shortest distance between objects that are too close to each other. The point clouds on the server are kept up to date with the manipulators' joint sensors and laser scanner-based measurements. During coordinated motion control, the joint trajectories of the redundant manipulator are modified so that the collisions can be avoided, while at the same time, the trajectory of the end-effector maintains its initial trajectory if possible. Results are given for a 4-DOF redundant heavy-duty hydraulic manipulator to demonstrate the capability of this collision avoidance control system.**

I. Introduction

During the last several decades, many researchers have described collision avoidance methods for manipulators in the field of autonomous robotics. The main reason for this is the increased need for automated operations where the ability to avoid collisions plays an important role. Long-standing problems include obtaining accurate information from the manipulator environment and how to actually prevent the collision if a potential collision is detected. The general features of collision avoidance methods are well-known, and different approaches to solving the collision avoidance problem have been reported in the literature. Two main strategy types have been proposed for solving collision avoidance problems: global strategies and local strategies.

Global collision avoidance methods [1], [2] find a collision-free path from the starting position to the goal position if such a path exists. These methods are typical for mobile robots, where the task is to find a collision-free path to the desired location. Existing knowledge of the robot environment is used to plan the route. Therefore, these methods are sometimes called planning algorithms. However, global methods are computationally heavy and may not be applicable for real-time control applications.

Local strategy solutions are very popular real-time collision avoidance methods [3]. These methods treat obstacle avoidance as a control problem instead of a path planning problem. This enables modification of the manipulator's path if obstacles are detected or if the obstacles move too close to the manipulator. Therefore, these methods can be used in an unknown environment where there is no information about the location of the obstacles during the desired manipulator operations.

Local methods [4], [5], [6], [7] are based on potential fields in the manipulator's operational space. These methods are further divided into two categories based on how the methods use potential fields. When the collision avoidance method uses potential fields to generate forces on the manipulator [4], [5], the solution requires position-based impedance control (PBIC). Another approach is to use potential fields to generate joint velocities directly [6], [7]. This approach does not require a dynamic model of the manipulator and therefore is easier to implement.

Successful collision avoidance requires accurate information where a potential collision is about to happen. Despite the importance of detecting exact locations of potential collisions, only a few papers have examined how to detect these collision points of two geometrically complex objects. Often, the complex geometry of the objects is ignored by simplifying the geometry or by representing the objects as bounding boxes (BB). One of the most common approaches for finding the minimum distance between two objects is based on BBs [8], [9], [10], [11]. These sample-based methods are fast and require low computational power, but they are applicable only to simple geometries. A method for detecting collisions between more complex geometry is presented by [12], in which graphics processing unit (GPU)-based calculations for real-time implementation are required. Solutions with real-time capability, accurate enough collision detection, and a shortest distance query method seem to be lacking in the literature. Therefore, additional studies on detecting collisions exactly and calculating the shortest distance are needed.

The aim of this paper is to present a collision avoidance system. We propose a method for detecting the collision points between two objects, presented as point clouds, and describe how to use this information in a collision avoidance system. The process for finding the shortest distance between two point clouds is divided into two phases. In the first phase, we find the approximate collision location of both point clouds and then use this information to find the exact collision points. Oriented bounding boxes (OBBs) are used in the first phase to detect possible collisions and to find the approximate collision location. After this, the actual shortest distance is calculated by using Euclidean distance. If there is a risk that the objects are about to collide, the calculated shortest distance is used with the proposed collision avoidance method to move the manipulator away from the obstacles and toward the desired goal.

Experiments validated that the proposed method can be used in real-time control. Multiple point clouds, representing the manipulator, were converted from three-dimensional (3D) computer-aided design models, where the number of points in each point cloud was restricted to 1000. Point clouds representing the environment and obstacles were created with a 3D laser scanner. With these point clouds, we created several collision pairs and monitored the shortest distance between the pairs. The results show that the proposed method is suitable for real-time applications to avoid collisions in which there can be one or more simultaneous collision risks.

The remainder of this paper is organized as follows: The next section presents the proposed collision detection method and how to calculate the shortest distance between two point cloud-type objects. The proposed collision avoidance algorithm is presented in the following section. Next, we describe the obstacle and environment detection system based on a 3D laser scanner. Finally, we show the experimental results for the proposed method, draw some conclusions, and describe future work.

TABLE I
FINDING THE SHORTEST DISTANCE BETWEEN TWO POINT CLOUDS.

Initialization phase	
1.	Input: Point clouds
2.	Create octrees for the point clouds
3.	Create axis-aligned bounding boxes (AABBs) with δx larger than the tight-fit AABBs
Calculation phase	
4.	Input: Object poses
5.	Create extended oriented bounding boxes (EOBBs) from AABBs with the input poses
6.	Intersection of the EOBBs using the separating axis theorem (SAT)
	• No intersection $\rightarrow$ No risk of collision
7.	Calculate the overlapping volume of the EOBBs
	• Create new OBBs of the volumes within each other
8.	Transform the new OBBs into the new AABBs with the inverse of the input poses
9.	Find the octree points that are inside the new AABBs
	• No points found $\rightarrow$ No risk of collision
10.	Transform the found points with the input poses
11.	Find the shortest distance within the transformed points

II. DISTANCE QUERY

To prevent the manipulator from colliding with obstacles or with itself, a manipulator control system must detect obstacles, as well as find the minimum distance between different parts of the manipulator and the obstacles. In the case of possible self-collisions, the distance between different parts of the manipulator has to be calculated. Finding the exact shortest distances between the manipulator and obstacles is crucial for an accurate collision avoidance system. For example, if the manipulator's workspace is small, it might not be possible to simplify the manipulator and the obstacles with BBs, because the volumes of the BBs are always larger than the actual volumes of the manipulator parts and the obstacles. This might lead to a situation where there is no more free space for the manipulator to move. Commonly, two types of BBs are used to simplify the distance query: an axis-aligned bounding box (AABB) and an oriented bounding box (OBB). AABBs are aligned with global coordinate axes, and the size of the AABB might change if the object inside is rotated. OBBs, instead, are aligned with the local coordinate axes of the object; therefore, the size of the OBB is not changed when the object inside is rotated. Choosing between the different BBs depends on the application.

Several studies have used BBs to represent obstacles and the manipulator and then found the shortest distance between BBs; for example, [10] and [11] used OBBs. Another popular method for finding the shortest distance between obstacles is to use vision-based techniques [13], [14]. Vision-based methods are widely used and are effective for calculating the shortest distance if the conditions are correct. For example, lighting should be appropriate so that the camera is able to see the objects. If the conditions are not good enough, vision-based methods cannot be used, for example, in applications with dirty and harsh environments, such as in underground mining applications. In applications with challenging environments, a laser-based measurement system that can produce point clouds is preferable.

Implementation of the point cloud-based distance query method is straightforward. However, calculating the shortest distance between two point clouds is a challenge for a real-time control system, because the calculation time depends on the size of the point clouds. Therefore, it is necessary to reduce the number of point cloud points among which the shortest distance can be found. This can be achieved by using a two-state calculation process. In the first state, an approximate location of the shortest distance is discovered. The second state is then used to calculate the actual shortest distance based on the approximate location. The overall process for calculating the shortest distance is described in Table I. In short, different types of BBs are used to minimize the number of point cloud points among which the shortest distance can be found. BBs are used to find an overlapping volume of two BBs. Then, an

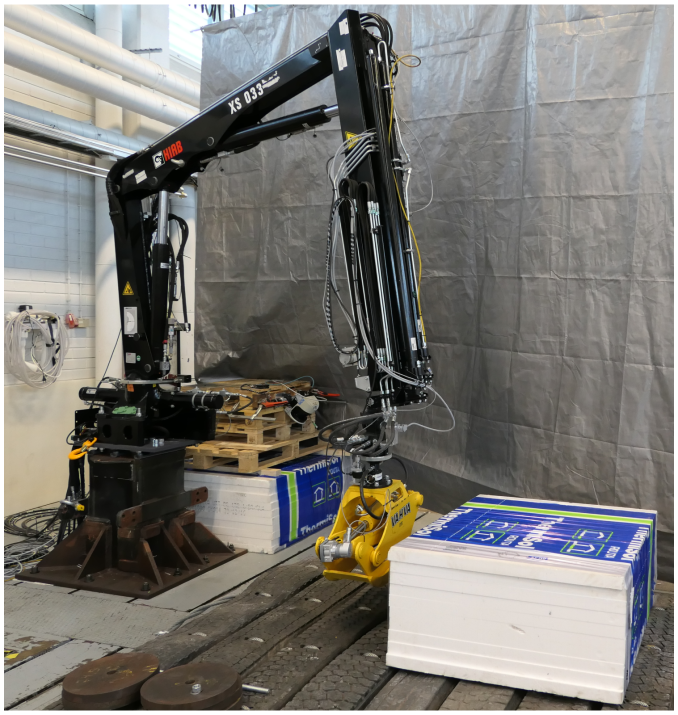

Fig. 1. Hiab crane with an obstacle in its workspace.

octree data structure [15] is used to extract subsets of points from the original point clouds according to the overlapping volume. Finally, the shortest distance between these subsets can be calculated, for example, using the Euclidean distance calculation.

Fig. 1 shows a typical hydraulic 4 degrees of freedom (4-DOF) manipulator used, for example, to lift logs. This figure also shows an obstacle inside the manipulator's workspace that needs to be avoided while operating the manipulator. To detect collisions and calculate the shortest distances between the manipulator and the obstacle, both are modeled with point clouds. The critical part of the manipulator, as a point cloud, is then used to detect potential collisions with obstacles. Fig. 2 shows these critical point clouds of the manipulator and the obstacle and the OBBs around these point clouds. EOBBs are used to detect potential collisions. The use of EOBBs enables us to detect collisions early enough so that the control system is able to react to the collision warnings. If the EOBBs do not intersect, then there is no risk of collision.

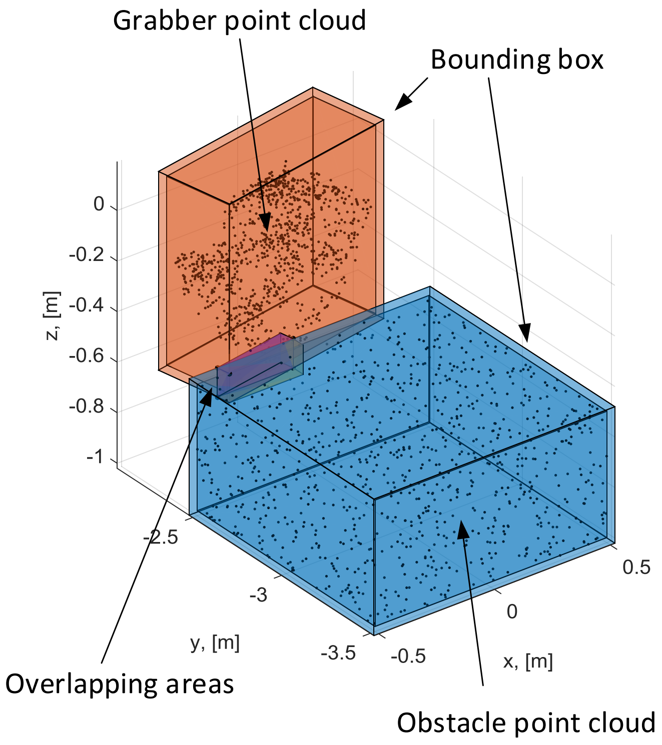

Fig. 2. The grabber of the manipulator and an artificial obstacle. The point clouds inside bounding boxes represent the grabber and the obstacle.

III. Collision Avoidance

The collision avoidance method used in this paper is based on the principle of the artificial potential field (APF) method. The APF method was pioneered by Khatib [5]. Thereafter, the principle of the APF method was extended to support different collision avoidance algorithms. Within the APF method, the manipulator and obstacles can be thought to have the same charge, and the goal position of the manipulator acts as a different charge. Therefore, the manipulator and the obstacles repel each other by generating a repulsive force between each other, and the goal position attracts the manipulator due to the opposite charge.

The final force of the manipulator is achieved by combining the attractive force and the repulsive forces caused by obstacles:

$$\mathbf{F}(\mathbf{p}) = -\nabla \mathbf{U}(\mathbf{p}), \tag{1}$$

$$\mathbf{U}(\mathbf{p}) = \mathbf{U}_{att}(\mathbf{p}) + \mathbf{U}_{rep}(\mathbf{p}), \tag{2}$$

where $\mathbf{F}$ is the final force, $\mathbf{U}$ is the sum of the potentials, and $\mathbf{p}$ is the position of the manipulator. The potential functions $\mathbf{U}_{att}$ and $\mathbf{U}_{rep}$ must be selected properly to achieve suitable results. In this paper, we rely on the potential functions proposed by [5].

A. Collision Avoidance for an End-Effector

A path for the end-effector of the manipulator to follow is created using a proportional position controller. Let $\mathbf{p}_d$ represent the goal position of the manipulator's end-effector. Then the position controller can be described as follows:

$$\dot{\mathbf{p}}_G = k_v \left(\mathbf{p}_d - \mathbf{p} \right), \tag{3}$$

where $\mathbf{p}_d$ is the goal position of the end-effector, and k_v is the position control coefficient.

The shortest distance between the manipulator parts and the obstacles is calculated by using the method proposed in section II. When the manipulator is too close to the obstacles, it is necessary to produce a repulsive force that prevents the manipulator from colliding with the obstacles. This can be achieved by producing an obstacle avoidance point velocity. The obstacle avoidance point velocity is calculated by using the shortest distance calculation method described in section II and by using the artificial potential field proposed by [5]:

$$\mathbf{U}_{O_i} = \begin{cases} \frac{1}{2}\mu\left(\frac{1}{\rho_i} - \frac{1}{\rho_O}\right), & \text{if } \rho_i \leq \rho_O \\ 0, & \text{if } \rho_i > \rho_O, \end{cases} \tag{4}$$

where ρ_O is the limit for collision avoidance, ρ_i is the distance between two obstacles, and μ is a scalar coefficient. Now $\dot{\mathbf{p}}_{O_i}$ can be formulated with (4),

$$\dot{\mathbf{p}}_{O_i} = \begin{cases} \mu\left(\frac{1}{\rho_i} - \frac{1}{\rho_O}\right)\frac{1}{\rho_i^2}\frac{\delta\rho_i}{\delta\dot{\mathbf{p}}_{O_i}}, & \text{if } \rho_i \leq \rho_O \\ 0, & \text{if } \rho_i > \rho_O, \end{cases} \tag{5}$$

where $\frac{\delta\rho_i}{\delta\dot{\mathbf{p}}_{O_i}}$ is the direction of the collision line. The path for the manipulator's end-effector can be calculated to be the sum of the attractive velocity and the repulsive velocities as follows:

$$\dot{\mathbf{p}} = \dot{\mathbf{p}}_G + \sum_{i=1}^{n_e}\dot{\mathbf{p}}_{O_i}, \tag{6}$$

where n_e is the number of end-effector obstacle avoidance points.

The linear velocity of the manipulator's end-effector must be bounded. This allows the use of a larger controller coefficient for the velocity, which ensures that the demanded end-effector velocity is also suitable near the goal position. The end-effector velocity is bounded as follows:

$$\dot{\mathbf{p}} = \begin{cases} \frac{\dot{\mathbf{p}}\dot{\mathbf{p}}_{max}}{max(|\dot{\mathbf{p}}|)}, & \text{if } max(|\dot{\mathbf{p}}|) > \dot{\mathbf{p}}_{max} \\ \dot{\mathbf{p}}, & \text{otherwise}, \end{cases} \tag{7}$$

where $\dot{\mathbf{p}}_{max}$ is the maximum linear velocity of the end-effector.

B. Collision Avoidance for a Manipulator's Body

If the manipulator's body parts are about to collide, these collisions have to handled differently from end-effector collisions. Repulsive velocities can be calculated as described in (5), but these velocities cannot be added directly to (6). Instead, a collision-free path planner based on null-space projection and the approach proposed by [16] have to be used. In [17], the general form of the solution for a null-space projection is shown to be

$$\dot{\mathbf{q}} = \tilde{\mathbf{J}}_G^{\dagger}\dot{\mathbf{p}} + \left(\mathbf{I} - \tilde{\mathbf{J}}_G^{\dagger}\mathbf{J}_G\right)\mathbf{z}, \tag{8}$$

where $\tilde{\mathbf{J}}_G^{\dagger}$ is a right damped pseudo-inverse of the manipulator's Jacobian matrix ($\mathbf{J}_G$), and $\mathbf{z}$ is an arbitrary vector in the $\dot{\mathbf{q}}$ space. A projection operator $\left(\mathbf{I} - \tilde{\mathbf{J}}_G^{\dagger}\mathbf{J}_G\right)$ describes the redundancy of the system and can be used to map an arbitrary $\dot{\mathbf{q}}$ into the null space of the transformation.

The primary goal is to reach the goal position with the manipulator's end-effector; the secondary goal is to avoid collisions with the manipulator's body. In [16], the authors presented a method for avoiding collisions with one obstacle avoidance point. In the present paper, this approach is extended

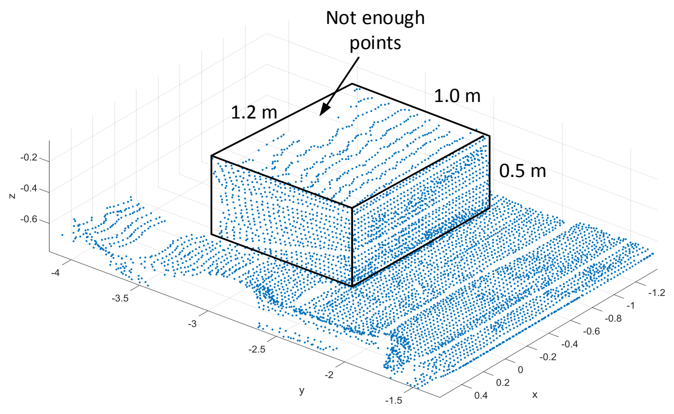

Fig. 3. The point cloud produced by the laser scanning system. Due to the position of the laser scanner, there is an area in the point cloud where the number of points is too low.

to cover multiple obstacle avoidance points. The primary and secondary goals are described by the equations

$$\mathbf{J}_G\dot{\mathbf{q}} = \dot{\mathbf{p}}, \tag{9}$$

$$\mathbf{J}_{O_i}\dot{\mathbf{q}} = \dot{\mathbf{p}}_{O_i}, \tag{10}$$

where $\mathbf{J}_{O_i}$ is the obstacle avoidance point Jacobian, and $\dot{\mathbf{p}}_{O_i}$ is the obstacle avoidance point velocity.

The secondary goal solution for $\dot{\mathbf{q}}$ is used as an arbitrary vector $\mathbf{z}$ to modify the solution of (8). By combining (8) and (10) with multiple obstacle avoidance points, the following solution can be derived:

$$\dot{\mathbf{q}} = \tilde{\mathbf{J}}_G^{\dagger}\dot{\mathbf{p}} + \sum_{i=1}^{n_o}\left[\left(\mathbf{I} - \tilde{\mathbf{J}}_G^{\dagger}\mathbf{J}_G\right)\tilde{\mathbf{J}}_{O_i}^{\dagger}\dot{\mathbf{p}}_{O_i}\right], \tag{11}$$

where n_o is the number of obstacle avoidance points, and $\tilde{\mathbf{J}}_{O_i}^{\dagger}$ is the damped right pseudo-inverse solution for $\mathbf{J}_{O_i}$. If the desired path cannot be maintained, (11) gives up the desired velocity $\dot{\mathbf{p}}$ to ensure a collision-free path.

IV. Obstacle Detection System

In order to avoid manipulator collisions with obstacles, they need to be detected and identified. Identification of obstacle dimensions is difficult and especially, in the case of unforeseen obstacles, might be impossible. Therefore, in this paper we used point cloud-based approach where it is not required to identify the obstacles. Instead, in the point cloud-based methods, it is possible to use the point cloud directly. The point cloud of the obstacles was produced with a laser scanning system.

The laser scanning system consisted of a SICK LMS511 laser scanner and a mechanism that rotated the scanner along one horizontal axis. The Scanner itself provided scans in only two dimensions, and a three-dimensional point cloud was acquired by rotating the scanner. A Maxon DC motor equipped with a planetary gear rotated the scanner. There was also an absolute encoder at the end of the shaft, providing accurate measurements of the scanner position.

The angular resolution of the laser scanner (the difference between two consecutive points in one scan) was chosen to be 0.25 degrees. As the scanner was mounted in an upright position, this defined the horizontal resolution of the point cloud. The scanner was configured to send measurements constantly at a frequency of 25 Hz, and at the same time, the scanner was rotated at a speed of roughly 5 degrees/s, which resulted in a vertical resolution of 0.2 degrees. Knowing the exact rotation speed was not necessary, because the angle measurements were assumed to be accurate at all times.

The laser scanner output was the distance value for each point in a scan. Combining these values with the angle information from the laser scanner and the encoder yielded the Cartesian coordinates of each point. The collected scanner data and relevant kinematic calculations were implemented using a dSpace Microautobox. Measurements from multiple scans were then combined into a point cloud using point cloud-handling methods provided by matlab (Computer Vision System Toolbox). Next, points located outside the working area were removed from the data set, and the point cloud was downsampled so that there were only points within 3 cm of each other. The number of points in the final cloud was approximately 5500, which proved to be sufficient for this application.

Fig. 3 shows the scanned and processed point cloud. Due to the position of the laser scanner, the point cloud points are not distributed evenly across the obstacle and the floor. The areas which are closer to the laser scanner have more points than the areas which are farther. There is also a small area at the back of the obstacle where there are no points (Fig. 3).

V. Experimental Case Study

The collision avoidance control system was developed to enable autonomous operation of a redundant serial manipulator. To verify this functionality, an experimental test was carried out with a heavy-duty 4-DOF hydraulic serial manipulator, the Hiab XS 033. The Hiab XS 033 manipulator is a hydraulic multi-purpose manipulator for almost a limitless range of applications. It has three revolute joints and one prismatic joint. In the following case study, the manipulator was equipped with a grabber that included two free non-actuated joints. During the experiment, the tool center point (TCP) of the manipulator was controlled, and the grabber hung from the TCP parallel to the gravity.

During the experiment, the collision server for distance queries ran on a laptop with an Intel Core i7 processor. The server program to calculate the shortest distance between point clouds was programmed with C++/CLI and compiled to a standalone program. The average time needed to calculate the shortest distance between all collision pairs was 1.0 ms with a standard deviation of 1.7 ms. This proved that the proposed method could be used in a real-time control system where the sample time was 10 ms. However, the time required to calculate all potential collisions depends on the current configuration of the manipulator. If a different starting position and goal position are used, then the calculation time might also be different.

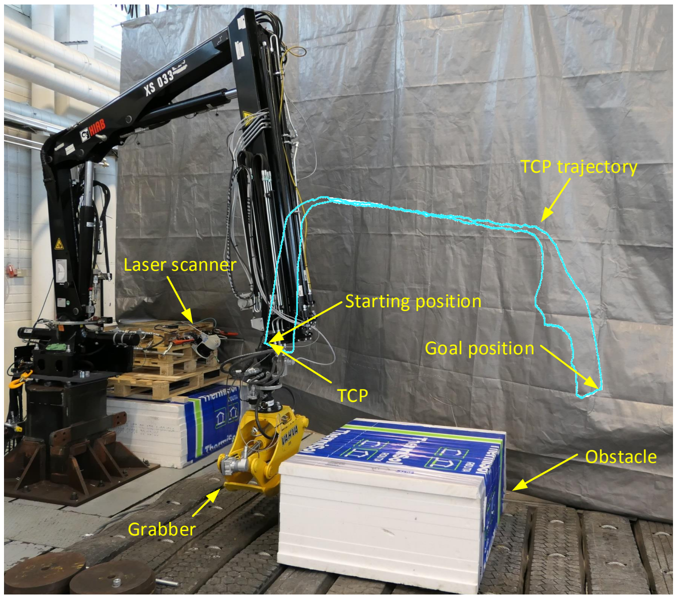

Fig. 4. The collision-free trajectory for the manipulator from the starting position to the goal position and back.

In order to validate the collision avoidance system, we conducted an experimental test. The task for the experiment was to move the grabber of the manipulator from a starting position ($\mathbf{q}_s$) to a goal position ($\mathbf{q}_g$) by avoiding collisions with obstacles. The starting position was at the left of the obstacle, and the goal position was at the other side of the obstacle (Fig. 4). With a conventional control system without collision avoidance, the grabber of the manipulator would collide with the obstacle if the grabber tried to move from the starting position to the goal position. The proposed collision avoidance control system of the manipulator together with the proposed distance query method execute the desired task without collisions by finding a collision-free path.

For the experiment, the 3D part of the grabber that could collide with the obstacle was converted into a point cloud, containing 1000 points (Fig. 2). The environment and the obstacle were scanned with the laser scanning system, containing 5500 points (Fig. 3). Thereafter, we created rules to monitor the shortest distance between possible collision pairs. If the distance between the collision pair was larger than the collision margin, then the distance had no influence on the control of the manipulator, but if the distance was smaller than the collision margin, we used (11) to modify the trajectory of the manipulator based on the calculated shortest distance between two point clouds.

Fig. 4 shows the overall experimental case study setup and how the manipulator, laser scanning system, and obstacle were located compared to each other. The trajectory of the TCP of the manipulator is also shown is this figure. This trajectory was automatically modified so that the grabber does not collide with the obstacle.

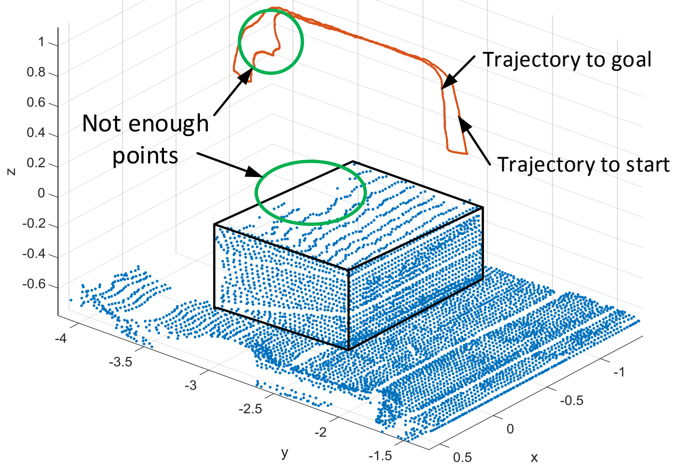

Fig. 5. The collision-free trajectory for the TCP of the manipulator from the starting position to the goal position and back with the scanned point cloud.

Fig. 5 shows the TCP trajectories of the manipulator from the starting position to the goal position and back. This figure reveals that the trajectory from the goal position to the starting position is not ideal and contains unwanted movements. These movements are caused by the lack of points at the back of the obstacle's point cloud. This area is shown in Fig. 5. If the points of the obstacle's point cloud are distributed evenly, it can be assumed that the trajectories on both sides of the obstacles would be similar.

VI. Conclusion

For autonomous manipulators to accomplish the given task without human supervision, one of the most challenging problems is avoiding collisions. A shortest distance-based collision avoidance method was used to obtain a collision-free path from the starting position to the goal position and back with a novel shortest distance query algorithm. Multiple simultaneous collision avoidance points can be used to prevent self-collisions, as well as collisions with other manipulators, and any part of the manipulator from colliding with obstacles or the environment. The proposed method can be used with a real-time control system, and the method does not require any path planning time because all decisions are made spontaneously.

The proposed method for calculating the shortest distance ensures that the shortest distance between two point cloud-type objects can be calculated in real-time. This method minimizes the calculation power needed to find the shortest distance by using a method that reduces the number of points among which the shortest distance can be found. The use of point cloud data extracted from CAD design models and from an external laser scanning system ensures that the possible collision locations are found with high accuracy. This enables the use of only small margins around obstacles to avoid collisions.

In future work, we intend to improve the collision avoidance system to cover more complex systems with a fully automatized sequence and position control with moving obstacles in the workspace of the manipulator.

Acknowledgment

This work was funded by the Doctoral School of Industry Innovations (DSII) of Tampere University of Technology. This funding is greatly appreciated. The DSII supports effective progress toward doctoral degrees, offers companies novel avenues for addressing research questions that are relevant in terms of their future competiveness, and generates new scientific knowledge.

References

[1] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[2] T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Transactions on Computers*, vol. C-32, no. 2, pp. 108–120, Feb 1983.

[3] L. Zlajpah and T. Petric, *Obstacle Avoidance for Redundant Manipulators as Control Problem*. InTech, 2012. [Online]. Available: https://www.intechopen.com/books/serial-and-parallel-robot-manipulators-kinematics-dynamics-control-and-optimization/obstacle-avoidance-for-redundant-manipulators-as-a-control-problem

[4] O. Khatib, "Dynamic control of manipulator in operational space," in *Proc. 6th IFToMM World Congress on Theory of Machines and Mechanisms*, 1983, pp. 1128–1131.

[5] ——, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, Mar 1985, pp. 500–505.

[6] J. O. Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 338–349, Jun 1992.

[7] F. Padula and V. Perdereau, "A new pseudoinverse for manipulator collision avoidance," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 14 687 – 14 692, 2011.

[8] S. Gottschalk, M. C. Lin, and D. Manocha, "Obbtree: A hierarchical structure for rapid interference detection," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 171–180.

[9] G. van den Bergen, "Efficient collision detection of complex deformable models using aabb trees," *J. Graph. Tools*, vol. 2, no. 4, pp. 1–13, Jan 1998.

[10] D. Puiu and F. Moldoveanu, "Real-time collision avoidance for redundant manipulators," in *2011 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, May 2011, pp. 403–408.

[11] I. Lee, K. K. Lee, O. Sim, K. S. Woo, C. Buyoun, and J. H. Oh, "Collision detection system for the practical use of the humanoid robot," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Nov 2015, pp. 972–976.

[12] K. B. Kaldestad, S. Haddadin, R. Belder, G. Hovland, and D. A. Anisi, "Collision avoidance with potential fields based on parallel processing of 3d-point cloud data on the gpu," in *2014 IEEE Int. Conf. Robotics Autom. (ICRA)*, May 2014, pp. 3250–3257.

[13] T. Asfour, P. Azad, N. Vahrenkamp, K. Regenstein, A. Bierbaum, K. Welke, J. Schrder, and R. Dillmann, "Toward humanoid manipulation in human-centred environments," *Robotics and Autonomous Systems*, vol. 56, no. 1, pp. 54 – 65, 2008, human Technologies: Know-how.

[14] S. M. Grigorescu, D. Ristić-Durrant, and A. Gräser, "Rovis: Robust machine vision for service robotic system friend," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2009, pp. 3574–3581.

[15] H. Samet, *An Overview of Quadtrees, Octrees, and Related Hierarchical Data Structures*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 51–68. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-83539-1-2

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

[17] T. N. E. Greville, "The pseudoinverse of a rectangular or singular matrix and its application to the solution of systems of linear equations," *SIAM Review*, vol. 1, no. 1, pp. 38–43, 1959. [Online]. Available: http://www.jstor.org/stable/2028031