Dynamic Collision Avoidance for an Anthropomorphic Manipulator Using a 3D TOF Camera.

Sankaranarayanan Natarajan, Andreas Vogt and Frank Kirchner
German Research Center for Artificial Intelligence (DFKI GmbH), Robotics Innovation Center, Bremen, Germany

Abstract

This paper describes a practical approach for dynamic collision avoidance for a position-controlled anthropomorphic manipulator. The collision avoidance problem is solved as a virtual spring-damper force control problem [16]. A 3D Time-Of-Flight (TOF) camera is used as a proximity sensor to identify the position of the obstacle near the elbow of the manipulator. The dynamic collision avoidance algorithm forms a virtual spring-damper around an obstacle which is near to the elbow of the manipulator. The inverse kinematics of the manipulator is solved analytically [17], in which the redundancy is used to avoid the obstacle. The experimental results show that the dynamic collision avoidance algorithm along with the 3D TOF camera can be used to avoid dynamic obstacle.

1 Introduction

These days robots are required to work closely and even interact with humans within its workspace. In order to avoid the risk of a collision with humans or machines in clustered and dynamic environments, a high safety standard has to be ensured. A proper path has to be planned to achieve a collision free path. Generally, a path can be planned globally or locally. The global path planning is done on configuration space and in a higher level of the manipulator control hierarchy. They require complete geometrical details of the manipulator and its environment and the planner will calculate a collision free path. Therefore global path planners are not suited for dynamic and unstructured environment. In recent years, researches have demonstrated different theories [7] and practical approaches to handle the local path planning [8]. This local path planning or the so-called dynamic collision avoidance can be used as *online* collision avoidance. They are not computationally expensive and suited for dynamic environment.
This paper describes a practical approach and shows experimental results for a position-controlled anthropomorphic manipulator to avoid dynamic collision avoidance with the help of a 3D TOF camera. This dynamic collision avoidance method is motivated by the classical contact force control problem for a position-controlled manipulator [15], [16]. Most of the dynamic collision avoidance methods use data from proximity sensors to avoid a manipulator colliding with an obstacle in its workspace. Commonly used proximity sensors such as stereo cameras and artificial skin have some drawbacks. For instance, cameras possess time delays caused by complex calculations or produces errors due to bad lighting conditions. Artificial skins are computational expensive and have complex hardware parts. To avoid these problems we are using a 3D-TOF camera which produces fast and adequate depth information even for a moving object. Out of the visual information from the camera, we are able to extract the manipulator and obstacles. The dynamic collision avoidance algorithm will form a user defined safety zone around an obstacle which is nearer to the elbow part of the manipulator. A virtual torque will be generated, as the elbow of the manipulator intrudes into the safety zone. The dynamic collision avoidance algorithm will reduce this *virtual torque* to zero by moving the elbow part of the manipulator away from the obstacle without perturbing the end-effector pose. This method can be applicable to any seven Degrees Of Freedom (DOF) anthropomorphic manipulator which also known as S-R-S manipulator.
In the following section, the kinematics and the inverse solution of an anthropomorphic manipulator are explained. Section 3 describes how the 3D TOF camera is used to get environment data and in section 4 the dynamic collision avoidance strategy is briefly described. Experimental results are shown in section 5 and the conclusion of our work in section 6.

2 Kinematics of An Anthropomorphic Manipulator

An anthropomorphic manipulator has a structure similar to a human arm. The industrial manipulator we considered offers a 7 DOF like a human arm through seven consecutive revolute joints, which are placed perpendicular to their axis of rotation. In this manipulator, the first three axis of rotation will intersect at a single point and similarly the last three axis of rotation will intersect at a single point. The sketch of an anthropomorphic manipulator is shown in **Figure 1a**. So one can virtually replace the first three joints by a virtual spherical joint (shoulder) and similarly the last three joints by a virtual spherical joint (wrist). These two virtual spherical joints are joined together through a revolute joint, which act as a elbow similar to human arm structure.

2.1 Analytical Inverse Solution for an Anthropomorphic Manipulator

The anthropomorphic manipulator considered here is a redundant manipulator because of its one excess degree of freedom. A lot of work has been done to use this excess degree of freedom to achieve tasks like smooth manipulation, dexterous manipulation, to avoid obstacles in its workspace, optimizing the torque, and singularity avoidance. One problem with the redundant manipulators is its solution to inverse kinematics. The inverse kinematics provides joint angles for the manipulator to reach any desired position and orientation. Because of its redundancy there will be infinite solution to its inverse kinematics. The most common way to solve the inverse kinematics problem is to use a iterative method which are quite computationally expensive and it would not provide all possible solutions. On the other hand analytical solution for an inverse problem is always preferable for its faster computation and for finding all the feasible solutions. Hence to manipulate a robot arm in an unstructured and dynamic environment, one needs to choose a closed form inverse solution because of its faster computation. A closed form solution for the inverse kinematics of a redundant robot arm was proposed by P. Dahm and F. Joublin [2]. Similar approaches have been described in [17], [14]. O. Ivlev and A. Graeser solved the redundancy through Imaginary Link [5].

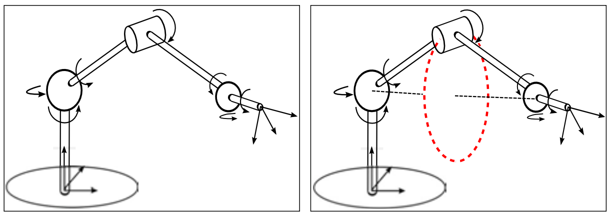

(a) Sketch of an anthropomorphic manipulator **(b)** Anthropomorphic manipulator with redundancy circle (red circle)

Figure 1: Anthropomorphic Manipulator

In our approach we use the method proposed by Shimizu et al. [17], where the inverse kinematics problem is solved analytically and all the feasible inverse kinematics solution in the global configuration space considering the joint limits are found. Due to spherical motion of the shoulder joint and the wrist joint, they form spheres. The intersection of those two spheres would be a circle where the elbow will be located [14]. This circle is called redundancy circle (see **Figure 1b** the elbow can move along this redundancy circle if there are no joint limits. This redundancy can be solved by parameterizing it, a generic parameterization for an anthropomorphic manipulator is proposed in [9]. The parameter "arm angle" which is used to represent the redundancy is the angle between arm plane and the reference plane. The arm plane passing through the shoulder, elbow and the wrist and the reference plane is passing through the shoulder, elbow and the wrist with a constraint that the joint 3 should be zero. The inverse kinematics solution will provide two values. One is a range of the arm angle and the other is the joint angle for the manipulator to reach its desired position and orientation. By using this arm angle the elbow of the manipulator is moving along the redundancy circle without perturbing the end-effector position and orientation and considering the joint limits too. The dynamic collision avoidance algorithm will control this arm angle to avoid the obstacle which is near to the elbow of the manipulator.

3 3D Time-Of-Flight Camera

Fast and accurate 3D sensing is becoming more and more important for mobile robots and manipulators. Nowadays, robots are used in complex and dynamic changing environments, not only in the field of outdoor scenarios like *Search and Rescue Robotics* (SARR) but also in office buildings or in production scenarios. In those scenarios the robots capture the actual scene with their sensors and generate data sets for position estimation, object recognition or collision avoidance. These scenarios need fast sensors, but also sensors which keep reasonable amount of data for an efficient processing time. For this purpose we selected the TOF camera, because of its high frame rate and the appropriate amount of produced data. The side-effects of these properties are errors and noise, as described in [13] and [10], which have to be handled by the application. In the following two sections we give a short overview of how to eliminate, respectively reduce, the error and noise.

3.1 Camera Calibration

There are three major systematic errors for TOF-based depth sensors, they are

- distance-related (circular error): This is caused by a non-harmonic sinusoidal signal resulting in the systematic distance error.
- amplitude-related: Different depth measurements are caused by objects in the same distance with different reflexivity.
- Latency-related / fixed pattern noise: Caused by different latency and different material properties in each CMOS-gate.

These errors can be controllable by appropriate calibration methods explained in [3, 4, 6] and [11]. In your approach we used the calibration toolbox from MATALB.

3.2 Filtering

The 3D TOF camera inherent some non-systematic errors which cannot be suppressed by camera calibration or by adapting camera parameters. Some of those errors are,

- Error Light Scattering: The scattering effect is caused by multiple reflections occurring on the camera sensor and optics. Therefore close objects can affect the measured distance of an element that belong to far objects. However, the newest SR-4000 is not affected by this error [1].
- Bad Signal to Noise ratio: The large amount of ambient radiation in comparison to the emitted radiation leads to less reliable measurements and cannot be eliminated. This error can be avoided by increasing the exposure time or amplifying the illumination. Furthermore, it provides the possibility to filter data points with low amplitudes.
- Multiple Reflection: This error is produced by an emitted signal that interfered with a signal that processed a longer distance caused by angled areas, for instance corners.
- There is a further effect, the so-called jumping edges, that has a massive influence on the measurements. Some objects are disconnected due to the occlusion. This means, the transition from one object to another changes suddenly but the TOF camera measures wrong data points between this transition. In [13] and [12] effective filter approaches are presented.

These errors can only be avoided by using appropriate filters. Section 3.4 explains the filter technique used to eliminate jumping edge error.

3.3 Cube-World

For specifying objects, a grid-based filter [18] was implemented. For the grid-based filter the pyramidal viewing range of the TOF camera was divided into quadratic space cells in the following called cubes. By a field viewing angle of about 45 degrees and a range of view about 8 meters and a resolution of 10 cm follows with

$$C_k = \sum_{n=1}^{k} \begin{cases} if\ k = odd & k^2 \\ else & 0 \end{cases} \tag{1}$$

a number of 102554 cubes. If each cube contain a minimum number of data points then the cube is accepted as an object. As search algorithm the kd-tree together with Best-Bin-First is used. Thereby three different parameters can be adapted empirically. The size of the cube (*SC*) which means to modify the search space. The amount of browsed child's (*CS*) inside the kd-tree which has an influence of the search time. And the number of points *NP* which are accepted in order to declare a cube as an object.

3.4 Jump Edge Filter

To reduce the problem of the jumping edges, the algorithm introduced in [13] was implemented and modified. This approach combines two thresholds for filtering the set of 3D points $P = \{\mathbf{p}_i | \mathbf{p}_i \in \mathbb{R}^3, i = 1, \ldots, N\}$. First of all, the angles ξ_i of the triangle are calculated which is formed by the focal point $\mathbf{f}$, the actual point $\mathbf{p}_i$ and its eight neighbours $\mathbf{p}_n$. If the angle ξ_i is greater then a certain threshold ξ_{th} the 3D point is sorted out. Then the set of sparse points S are removed with a certain threshold d_{th} by calculating the mean distance among the actual point $\mathbf{p}_i$ and its eight neighbours $\mathbf{p}_n$:

$$d(\mathbf{p}_i) = \frac{1}{k} \sum_{n=1}^{k} \|(\mathbf{p}_i - \mathbf{p}_{i,n})\|, \tag{2}$$

$$S = \{\mathbf{p}_i | d(\mathbf{p}_i) < d_{th}\}, \tag{3}$$

For calculation of the angle ξ trigonometric function are needed which consume computational time and slow down the whole process. Hence, instead of calculating angle ξ, the maximum value in z-direction of $\mathbf{p}_n$ was used (see **Figure 2**):

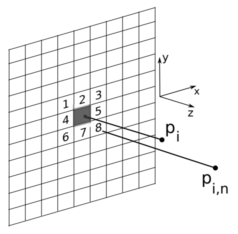


Figure 2: Jump Edge Filter

$$Z_i = max(\mathbf{p}_{i,n}(z)), \tag{4}$$

$$T = \{\mathbf{p}_i | Z_i > Z_{th}\}, \tag{5}$$

After filtering the set Q contains the remaining points:

$$Q = \{\mathbf{q}_i | \mathbf{q}_i \in P \setminus (T \cup S)\}. \tag{6}$$

4 Dynamic Collision Avoidance

The dynamic collision avoidance problem is solved as a simple force control problem. This approach is inspired by the work done by *Seraji and Bon*, who developed an approach for real-time collision avoidance for Position-Controlled manipulators [15]. According to this method, every obstacle in the environment has a safety zone. The safety zone is a user defined stand-off distance. When the manipulator intrudes into the safety zone, a virtual intrusion force will be generated. The collision will be avoided by changing the manipulator trajectory, which will nullify the virtual force. Once the force is nullified, the manipulator will be out of that safety zone.

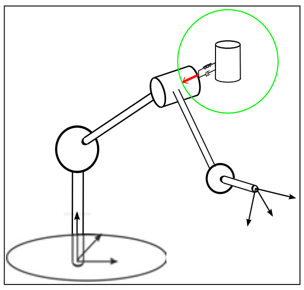
4.1 Virtual Spring-Damper System

Based on the above mentioned approach, we developed a Dynamic Collision Avoidance Controller which obtains the nearest obstacle position through the TOF camera. Then it forms a safety zone around the obstacle by using the user-defined stand-off distance d_{st} and manipulator link radius d_r. When the elbow of the manipulator intrudes into the safety zone, an intrusion vector C_f will be generated as shown in **Figure 3a** in the direction from obstacle position P_o towards the elbow position P_o. Its magnitude is the amount of intrusion of elbow into the safety zone.

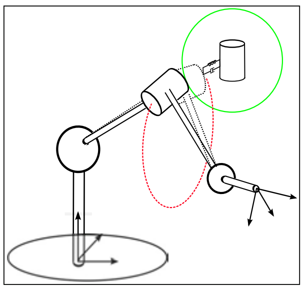
$$|C_f| = (d_st + d_r) - |P_o - P_e| \tag{7}$$

virtual torque T is given by:

$$T = K_i C_f + K_p \frac{dC_f}{dt}. \tag{8}$$



(a) An intrusion vector (red arrow) is generated by intrusion of the elbow into the safety zone



(b) how the manipulator uses its self-motion to move away from the safety zone

Figure 3: Obstacle is surrounded by safety zone represented as green circle

The sign of C_f is used to find into which direction the arm angle should perturb from its actual value. This *virtual torque* is nullified by perturbing the arm angle by the amount of:

$$\phi_c = (K_p C_f)/r + ((K(C_f)K_i)/r) \int C_f dt \tag{9}$$

where $K(C_f)$ is non-linear gain. A simple- piecewise-linear continuous function is used. Multiplying this gain with the integral term ensures that as the elbow moves away from the safety zone, the arm angle will not perturb abruptly to its actual value, but instead move smoothly to its actual value :

$$K(C_f) = \begin{cases} 0, & if \quad C_f \leq 0 \\ \frac{C_f}{d0}, & if \quad 0 < C_f < d0 \\ 1, & if \quad C_f \geq d0 \end{cases} \tag{10}$$

In the above equation $d0$ represents the value of C_f at which the full perturbation will be applied. By changing the arm angle by the amount ϕ_c, the elbow of the manipulator will be moved away from the safety zone as shown in **Figure 3b** . Hence, the elbow of the manipulator avoids the obstacle without perturbing the end-effector position and its orientation.

4.2 Dynamic Collision Avoidance Algorithm

In this subsection we will explain how the Dynamic collision Avoidance Algorithm works. A simple block diagram of our algorithm is shown in **Figure 4**.

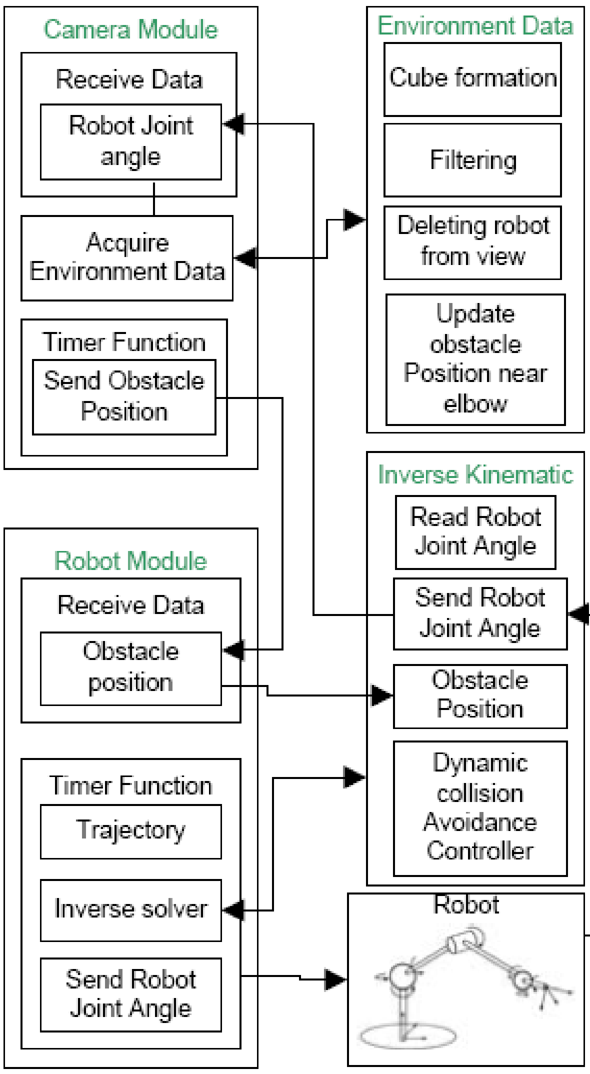


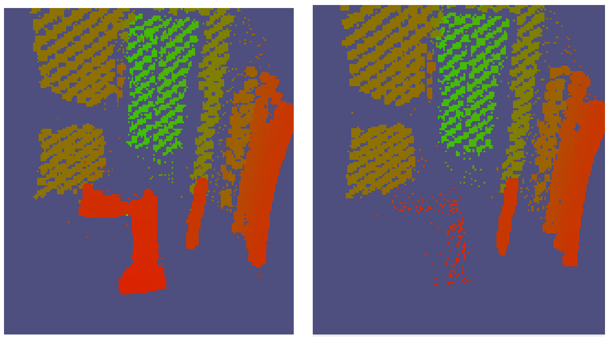
Figure 4: Block Diagram of Dynamic Collision Avoidance Algorithm

There are two main modules, one is the *Camera Module* and the other is the *Robot Module*. Each module will be explained in the following subsections.

4.2.1 Camera Module

The *Camera Module* has three main tasks, namely receiving data from the *Robot Module*, acquiring the environment data and sending the obstacle position to the *Robot Module*. The *Camera Module* will form a cube-world from the environment data as described before in the section 3.3.

The *Camera Module* receives the current joint angle of the manipulator, using this data it calculates the manipulator position with respect to the camera coordinates frame. The **Figure 5** shows the manipulator position from the perspective of the SR4000. The world is subdivided into cubes, and for an improved overview the cubes were overlaid over the original set of data points. The colour of the cubes and of the data points depend on the distance between the TOF camera and the objects in the environment. In the picture in **Figure 5a** the manipulator is visible as a red "L" upside down. After deleting the manipulator from the cube-world only the original data points are left as you can see in the **Figure 5b**. Hence, the manipulator is not treated as an obstacle anymore.



(a) The manipulator is recognized as an object **(b)** The manipulator was deleted from the camera view

Figure 5: Cube-world of the laboratory environment with the manipulator in the foreground (red). The parameters of the cube-world are: $SC = 10cm$, $CS = 20$, $NP = 20$

After deleting the manipulator from its view, it will check for any obstacle near the manipulator elbow based on the user-defined distance. If it found any obstacle it will send the obstacle position after converting its position with respect to manipulator coordinates frame.

4.2.2 Robot Module

The *Robot Module* has three main tasks namely, receiving data from the *Camera Module*, solving the inverse kinematics and checking the collision avoidance, and sending the joint angles to the manipulator. The inverse kinematics is done for desired position and orientation of the end-effector provided by the trajectory generator. The Dynamic collision Avoidance controller will check whether the elbow of the manipulator is inside the safety zone or not. If its inside the safety zone the controller will calculate appropriate arm angle as described in the section 4.1 which will move the elbow of the manipulator outside the safety zone. After that, the inverse kinematics uses this arm angle to find the joint angles that are to be send to the robot controller which will move the manipulator to the required position.

5 Experimental Results

In our experimental setup, we used Mitsubishi's PA-10 with 7 DOF as an anthropomorphic manipulator. The manipulator's base coordinate frame is considered as a world coordinate frame. As the TOF camera we used the SR4000, the camera coordinate frame is known with respect to the world coordinate frame.

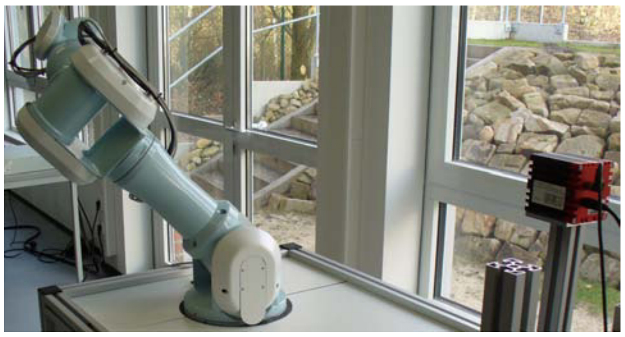


Figure 6: Experimental set up: Mitsubishi's PA-10 with the SR4000

The *Camera Module* is executed on an Intel-Core-Duo(TM) with 2,1 GHz with Ubuntu 8.04 (Hardy) as operational system. Building the cube-world as described in section 3.3 costs more processing time as more cubes are searched through. Table 1 shows results of the cycle time by varying the range of view and thus the numbers of cubes.

range of view [m]	cycle time [ms]
1	130
2	280
3	650
4	1380
5	2540

Table 1: Cycle time for building the cube-world by searching through all 102554 cubes depending on the range of view.

nth data point [pixel] (range of view 8m)	cycle time [ms]
1	1200
2	580
5	270
10	150
20	89

Table 2: Cycle time for building the cube-world by taking every nth pixel out of 25344.

To find an appropriate range of view and cycle time the original approach was modified. The cubes are now formed by taking every nth pixel of the data set as shown in table 2. This reduce the search space from 102554 to 25344 cubes or less. For the time measurement the parameters kept are stable by *SC=10cm*, *CS=25* and *NP=20*. The *Robot Module* runs on a AMD-Athlon(TM) with 2.4 GHz and Windows XP as an operation system. **Figure 7** and **Figure 8** shows how the elbow of the manipulator moves to avoid a dynamic obstacle. The reaction time which is the time taken for the calculation of the dynamic collision algorithm to generate new joint angles for the manipulator to avoid the obstacle, is 60 μs.

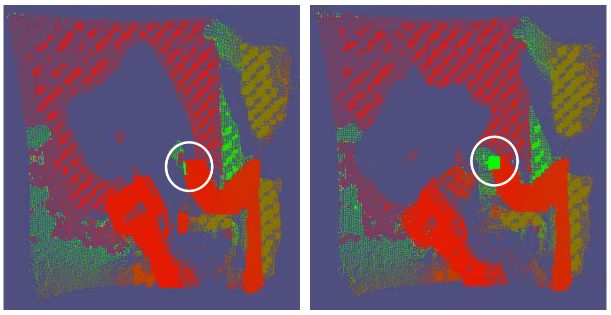

(a) An obstacle coming from right side of the elbow **(b)** The elbow moves in the left side to avoid the obstacle

Figure 7: The green cube (circled) represents the nearest obstacle to right side of the elbow. The parameters of the cube-world are: $SC = 5cm$, $CS = 25$, $NP = 20$

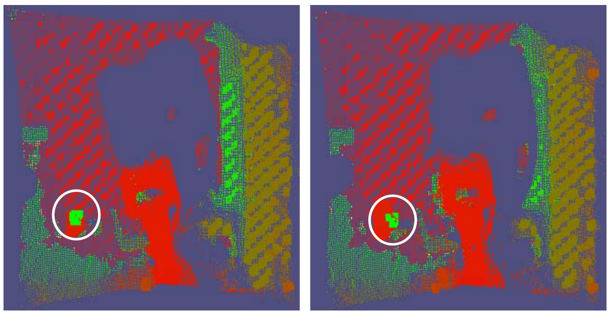

(a) An obstacle coming from right side of the elbow **(b)** The elbow moves in the right side to avoid the obstacle

Figure 8: The green cube (circled) represents the nearest obstacle to left side of the elbow. The parameters of the cube-world are: $SC = 5cm$, $CS = 25$, $NP = 20$

6 Conclusion

We have successfully implemented a Dynamic Collision Avoidance Algorithm for an anthropomorphic manipulator using 3D TOF camera. The experimental results show, how the elbow of the manipulator avoids a dynamic obstacle using the obstacle position measured by a 3D TOF camera. Despite the small data set produced by the TOF camera the latency of the processing time is still inadequate for a security scenario. But, as we have shown, the processing time is decreased by scaling down the data set or reducing the range of view. For future applications, we propose to fix the search space around the manipulator, which leads to a small and constant search space, and therefore reduces the time significantly. Furthermore, we can also see an opportunity for time reduction in code optimization.

7 Acknowledgment

The work presented in this paper is part of the Semantic Product Memory (SemProM) project, funded by the German Federal Ministry of Education and Research (BMBF), grant No. 01IA08002.

References

[1] Filiberto Chiabrando, Roberto Chiabrando, Dario Piatti, and Fulvio Rinaudo. Sensors for 3d imaging: Metric evaluation and calibration of a ccd/cmos time-of-flight camera. *Sensors*, 9(12):10080–10096, 2009.

[2] P. Dahm and F. Joublin. Closed form solution for the inverse kinematics of a redundant robot arm. Technical report, Institut für Neuroinformatik, Ruhr-Universität Bochum.

[3] S. Fuchs and G. Hirzinger. Extrinsic and depth calibration of tof-cameras. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Anchorage, USA*, June 2008.

[4] S. Fuchs and S. May. Calibration and registration for precise surface reconstruction. In *Proceedings of the DAGM Dyn3D Workshop, Heidelberg, Germany*, September 2007.

[5] O. Ivlev and A. Gräser. An analytical method for the inverse kinematics of redundant robots. In *in Proc.3rd ECPD Int. Conf. on Advanced Robots, Intelligent Automation and Active Systems*, pages 416–421, 1997.

[6] T. Kahlmann, F. Remondino, and H. Ingensand. Calibration for increased accuracy of the range imaging camera swissranger. In *IEVM06*, pages 136–141, 2006.

[7] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *International Journal of Robotics*, 1986.

[8] S. Klanke, D. Lebedev, R. Haschke, J. Steil, and H. Ritter. Dynamic path planning for a 7- dof robot arm. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[9] K. Kreutz-Delgado, M. Long, and H. Seraji. Kinematic analysis of 7-dof manipulator. International Journal of Robotics Research.

[10] R. Lange. 3d time-of-flight distance measurement with custom solid-state image sensors in cmos/ ccd technology. Universität Siegen, 2000.

[11] M. "Lindner and A." Kolb. Lateral and depth calibration of pmd-distance sensors. In *ISVC06*, pages II: 524–533, November 2006.

[12] Marvin Lindner, Martin Lambers, and Andreas Kolb. Sub-pixel data fusion and edge-enhanced distance refinement for 2d/3d images. *Int. J. Intell. Syst. Technol. Appl.*, 5(3/4):344–354, 2008.

[13] S. May, D. Droeschel, D. Holz, C. Wiesen, and S. Fuchs. 3D Pose Estimation and Mapping with Time-of-Flight Cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on 3D Mapping*, Nice, France, October 2008.

[14] H. Moradi and S. Lee. *Advances in Intelligent Computing*, chapter Joint Limit Analysis and Elbow Movement Minimization for Redundant Manipulators Using Closed form Method.

[15] H. Seraji and B. Bon. Real-time collision avoidance for position-controlled manipulators. *Robotics and Automation, IEEE Transactions on*, 15(4):670–677, Aug 1999.

[16] H. Seraji, R. Steele, and R. Ivlev. Experiments in sensor-based collision avoidance. *Robotics and Automation, IEEE Transactions on*, 1995.

[17] M. Shimizu, H. Kakuya, W.-K. Yoon, K. Kitagaki, and K. Kosuge. Analytical inverse kinematic computation for 7-dof redundant manipulators with joint limits and its application to redundancy resolution. *Robotics, IEEE Transactions on*, 24(5):1131–1142, Oct. 2008.

[18] J.W. Weingarten, G. Gruener, and R. Siegwart. A state-of-the-art 3d sensor for robot navigation. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2155–2160 vol.3, Sept.-2 Oct. 2004.