Autonomous Trajectory Planning and Following for Industrial Underwater Manipulators

Achint Aggarwal[1] and Jan Albiez[2]

DFKI GmbH
Robotics Innovation Center (RIC)
Bremen, Germany
Email: [1]achint.aggarwal@dfki.de and [2]jan.albiez@dfki.de

Abstract: While autonomous motion planning and control of ground based manipulators has received a lot of interest by researchers in the past three decades, autonomous operation of industrial underwater manipulators is not that well documented. The main bottlenecks in the autonomous operation of underwater manipulators is the dependance on factory supplied master-slave controllers and absence of precise, high speed computer based controllers. Recently, a computer based communication interface was presented in [1] for the industrial Orion7P manipulator from Schilling robotics. The present paper builds up on this computer interface and presents the application of existing and proven methods of autonomous motion planning and control from ground based manipulation to the underwater manipulation domain. Approaches for autonomous collision free path planning and trajectory following are presented for applications involving industrial standard underwater manipulators. These approaches are validated by experiments conducted on a real Orion7P manipulator. This paper thus presents a complete approach for autonomous collision free trajectory planning and execution for industrial underwater manipulators.

Keywords: Autonomous motion planning, Industrial underwater manipulators, Adaptation of ground based technology, Collision free path planning, Trajectory following, Schilling robotics Orion7P

1. INTRODUCTION

Underwater manipulators mounted on remotely operated vehicles are extensively used for shallow and deep-sea missions for marine science, oil and gas extraction and exploration. Teleoperation remains the most common way of controlling these manipulators which inherently requires high-bandwidth data transmission over long distance communication links. However, the existing low bandwidth communication lines and large time delays inherent to acoustic sub-sea communication pose major obstacles to the teleoperation of a remote manipulation system. Thus, there is a well established demand for autonomous manipulation with industrial scale underwater manipulators in the deep sea industry. Further, autonomous underwater manipulation will relieve the manual operators from lengthy, tedious and expensive tasks of arranging tools and equipment, and will enable them to concentrate on more productive procedures that demand dextrous human intervention and control.

Autonomous operation of ground based manipulators has been a topic of immense interest for researchers in the last three decades. Robust autonomy for pick and place tasks has been successfully achieved atleast in structured terrestrial environments [2], [3]. This has been enabled by high speed computing, high frequency communication channels for robust control, and a tremendous surge in the amount of off-the-shelf sensors available for feedback generation from terrestrial robots. However, this research from terrestrial applications is not directly applicable to underwater manipulators because of the classical technology that is still used to control the commercial underwater systems. Most of the industrial-scale underwater manipulators are still supplied with a master-slave based control system which neither needs precise and high frequency control, nor high speed operation. Further, these systems are primarily designed for rough manipulation tasks in harsh environments and thus support very limited amount of sensor feedback.

There are some instances of custom made manipulator systems that have been developed for experimenting autonomous underwater manipulation [4]. However, the algorithms developed for these systems are not useful for commercial underwater vehicles, since the capabilities of the manipulator systems differ. Thus, there is an urgent requirement for autonomous operation of industrial-standard underwater manipulators.

Recently, a computer based communication interface was presented in [1] for the industrial Orion7P manipulator from Schilling robotics. In this paper, we build up on this computer interface and present the application of existing and proven methods of autonomous motion planning and control from ground based manipulation to the underwater manipulation domain. We present approaches for fast collision free path planning for high degrees-of-freedom (DoF) manipulator systems, trajectory planning for smooth motion generation at various speeds, and middle level computer control for execution of the planned trajectories on the manipulators. We demonstrate the effectiveness of the application of these ground based methodologies on underwater manipulators by performing real tests on an industrial six-DoF Orion7P underwater manipulator (Figure 1 [5]). To our knowledge this is

This work was supported by the German Bundesministerium fur Bildung und Forschung (BMBF,grant FKZ 01IW10001)

978-0-9839671-4-0 ©2013 IEEE

the first demonstration of autonomous collision-free trajectory planning and following for a real industrial underwater manipulator system.

This paper is structured as follows. The approaches for autonomous collision-free path planning, manipulator kinematics, trajectory generation and trajectory following are discussed in Section 2. The results of applying these algorithms on the Orion7P manipulator are presented in Section 3. Conclusion and outlook are discussed in Section 4.

2. AUTONOMOUS TRAJECTORY PLANNING AND FOLLOWING

In this section we first discuss the approach for solving manipulator kinematics which is essential for trajectory planning. Then in Section 2.2, we present an approach for autonomous generation of collision free path plans between a start and destination configurations of the manipulator. The paths generated by the collision free path planner do not contain timestamp information and thus differ from complete trajectories. In Section 2.3, we discuss the generation of smooth trajectories from collision free paths, and the computer controller for sending these trajectories to the manipulator.

2.1 Manipulator kinematics

For autonomous task execution by manipulators, it is often required to transform from joint space to task space and vice versa. For mapping between joint space and task space of the manipulator system, it is necessary to solve its forward and inverse kinematics(IK). While forward kinematic solutions are trivial for serial chain manipulators, the inverse kinematics is a much harder problem. For manipulators with upto six DoFs, closed form inverse kinematics solutions can be generated from the geometry. In [1], close form solutions for the Orion7P manipulator kinematics are presented. For manipulators with higher DoFs, numerical methods are used to solve inverse kinematics. IKFast [6] is a library that uses several numerical and closed form approaches to solve IK for high DoF manipulators. We use the latter implementation. We achieve a 94 percent accuracy for IK for the Orion7P arm using IKFast i.e. out of 1000 random reachable configurations of the manipulator end effector, the IK solver was able to find solutions 94 percent of times. The computation speed for one IK iteration is less than one microseconds.

2.2 Collision free path planning

In order to facilitate the execution of pick and place tasks autonomously, we generate globally optimum and collision free path plans which define the coordinated movement of the manipulator joints between the start and destination configurations. For avoiding collisions, it is essential to have an environment map which includes all objects present in the vicinity and within the reachable workspace of the manipulator. For experiments discussed in this paper, we will assume this map to be already available from external sensors like sonar and laser scanners. The environment is also assumed to be static during the process of global path planning.

The algorithms for generating global path plans for a manipulator can be broadly classified as analytical or probabilistic. Analytical path planners like A-star [7] and Dijkstra [8], analyze the workspace of the manipulator thoroughly before generating a globally optimum path. Probabilistic planners like Rapidly Exploring Random Trees (RRT) [9] and Probabilistic Roadmaps (PRMs) [10] on the other hand try to search for a solution by randomly sampling the workspace and trying to connect the start and destination manipulator configurations. While the analytical solvers are guaranteed to generate a solution if it exists, it comes at a high computational cost, especially for high degree of freedom manipulators, since the complete workspace of the manipulator is being analyzed. Also the volume of workspace in joint space increases exponentially with increasing degrees of freedom. The probabilistic solvers are only probabilistically complete, i.e. the probability of an existing solution being found approaches one as the search time approaches infinity. However, in most cases the latter are most suited for practical applications given their low computational time. Thus we use an advanced variation of RRTs for global collision free path planning as discussed below.

A kinematic model of the manipulator is created in the OpenRAVE simulation framework [6] using the manipulator CAD model data (as shown in Figure 2). The environment objects are also added to the simulation by creating triangulated mesh models from pointcloud data collected by sonar and laser scanners. We use the collision checking library from Open Dynamics Engine for checking environment collisions. Since a pick and place task is defined in the end-effector space of the manipulator, the inverse kinematics solver is used to transform to the joint space. Thus, given the end-effector poses at the start and goal positions, inverse kinematics yields the manipulator joint angles at the start and goal configurations. Finally, the Bi-directional Rapidly Exploring Random Tree algorithm [11] is used for generating a collision free path connecting the start and final configurations in joint space. This is a randomized sampling based algorithm which is effective in solving single-query path planning problems in high-dimensional configuration spaces. The algorithm works by incrementally building two RRTs rooted at the start and the goal configurations. The trees each explore space around them and also advance towards each other through the use of a simple greedy heuristic. This is a probabilistic, sampling based planner which has been proven to be effective for fast path planning for high-DoF serial chain systems.

2.3 Trajectory interpolation based controller

Most of the present industrial underwater manipulators are supplied with a master-slave based control system, which is imprecise and operates at a low-frequency. Computer based control is a pre-requisite for any kind of autonomous control of the manipulator. A computer interface implementation for the Orion7P manipulator is presented in [1]. This interface can be used for feeding

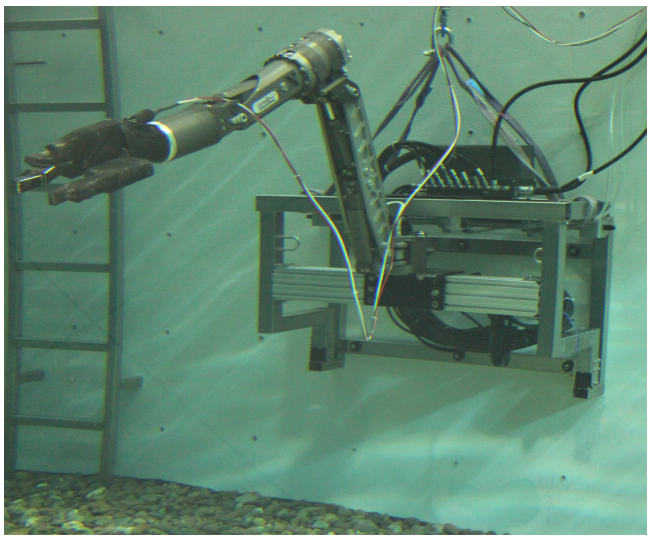

Fig. 1 The Orion7P industrial underwater manipulator with SeeGrip gripper attachment at the underwater testbed at DFKI RIC.

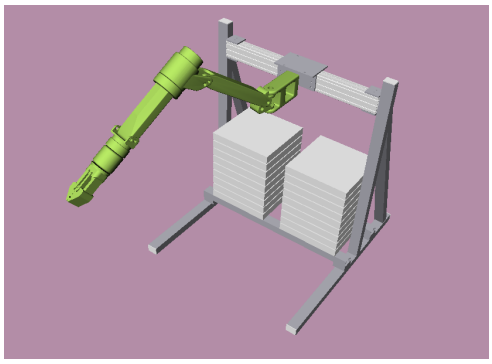

Fig. 2 Orion7P manipulator in OpenRAVE simulation environment

joint positions to the manipulator directly via a computer serial port which are then executed by the factory supplied lowest level position controller on the Orion manipulator. This lowest level position controller is imprecise and we have experienced an error of upto four degrees on each of the joints. Furthermore, directly feeding the joint positions to the lowest level position controller results in very jerky motions of the manipulator and is only suitable for master-slave operation with small joint increments at every update step. A high level speed controller is presented in [12] that can be used to attain a destination joint configuration smoothly and accurately. However, this controller could only be used for sending a single set of destination joint positions and sending a complete trajectory was not possible.

To solve these issues, we have implemented a trajectory interpolator based controller on top of the computer interface of [1]. The complete approach is shown in Figure 3. The path planner presented in Section 2.2 generates a set of joint configurations $P = \{p_0, p_1, ..., p_N\}$ where $p_i = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\}, i \in [1, N]$ between the start and the destination configurations. The *Assign Timestamps* phase then assigns timestamps to each of these configurations using the predefined maximum velocities for each joint. For each pair of joint configurations $\{p_j, p_{j+1}\}, j \in [0, N-1]$, the minimum possible time to reach p_{j+1} from p_j is computed such that the maximum joint velocities for each of the six manipulator joints are not violated. These minimum times are used to assign timestamps for each joint configuration p_i. In the *Trajectory Smoothing* phase, the joint velocities for each of the six joints in every joint configuration p_i are assigned by calculating the mean of the velocities in the two intervals before and after p_i. If the direction of motion changes, maximum velocity is assigned to p_i. Then a cubic polynomial from [13] is used to smooth the complete trajectory. It was determined that the serial computer interface of [1] could send joint positions to the arm at a frequency of 41.6 Hz. Thus, the trajectory is sampled at regular intervals of 24 ms in the *Trajectory Sampling* phase. The cubic polynomial from [13] is used for interpolating a point at the required time. The final interpolated trajectory is then sent to the manipulator via the serial computer interface.

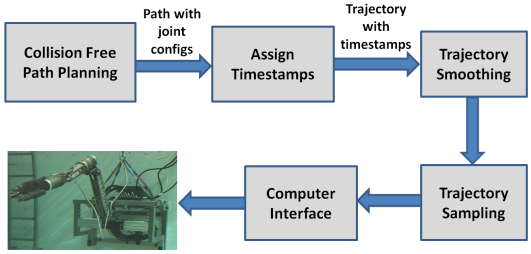

Fig. 3 The complete autonomous trajectory planning and execution methodology

3. EXPERIMENTS AND RESULTS

The trajectory planner, controller and trajectory follower discussed above were tested extensively on the Orion7P manipulator. The tests presented below were conducted with the manipulator in air.

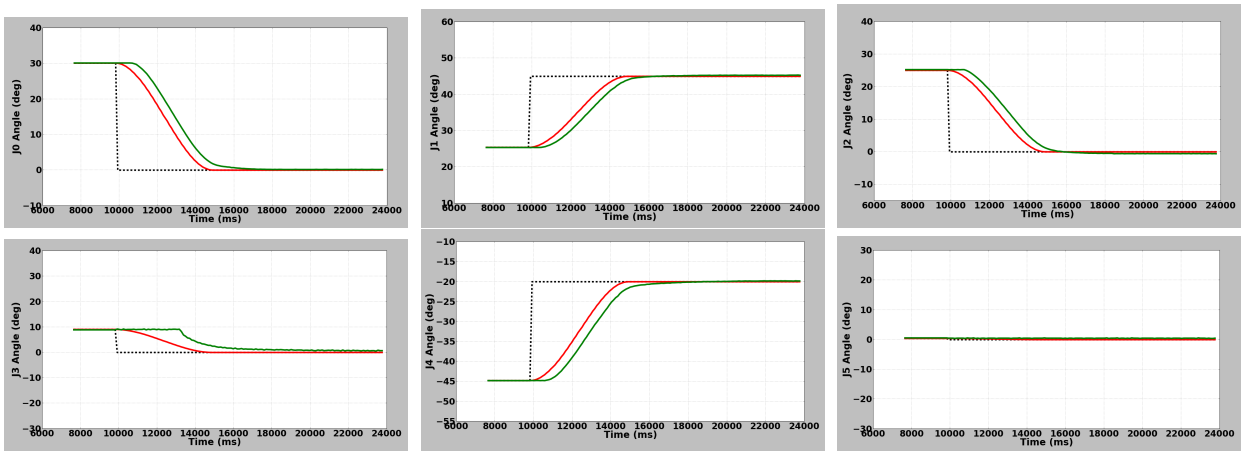

Fig. 4 Position control precision of the interpolation based controller. The six plots correspond to the six joints of the Orion7P manipulator. The desired final angles are shown by black dotted lines, interpolated trajectories sent from the controller are shown by red lines and the attained joint angles are shown by green lines.

Table 1 Position accuracy of interpolation based controller

Joint	Standard Deviation(Deg.)	Mean(Deg.)
Azimuth	0.138	0.973
Shoulder	0.049	1.164
Elbow	0.101	1.043
Roll	0.153	1.347
Pitch	0.189	0.872
Wrist	0.098	0.954

3.1 Position precision using the interpolation based controller

For testing the precision of the interpolation based controller, 100 randomly selected destination joint configurations were used. These joint configurations were sent to the Orion7P arm via the interpolation based controller. For each of the destination configuration, the interpolation based controller generated smooth trajectories from the current joint configuration to the destination joint configuration. Figure 4 shows the movement of the joints for one such iteration. It shows the destination joint position sent to the controller, the interpolated trajectory generated by the controller that is sent to the arm, and the joint positions attained by the arm. It can be seen that the interpolator generates smooth cubic polynomial based trajectories for each joint. Table 1 shows the position accuracy of the controller for all the joints. The mean for all 100 iterations is around one degree for each joint which is a large improvement over the factory supplied lowest level controller which yields an error of upto four degrees for each of the joints. The results are also comparable with the speed controller presented in [12].

3.2 Trajectory planning

The collision free path planner discussed in Section 2.2 was validated by generating collision free paths for 100 random configurations of the manipulator in the presence of static environment obstacles. The environment was only sparsely populated with obstacles, as one would expect in most underwater environments. The planner was able to find a solution 97 percent of times, with a mean computational time of 7.3 seconds on an Intel core I7 processor with a quad-core CPU of 2.9 GHz.

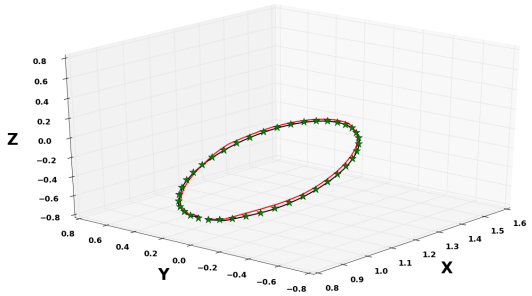

Fig. 5 Circular trajectory followed by the Orion7P arm. The commanded waypoints are shown by green * points. The black solid line shows the sent trajectory and red solid line shows the achieved trajectory.

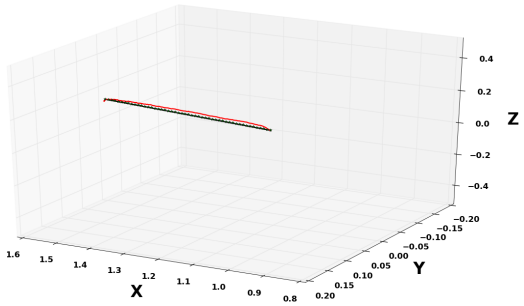

Fig. 6 Straight line trajectory followed by the Orion7P arm. The black solid line shows the sent trajectory and red solid line shows the achieved trajectory.

3.3 Precision of trajectory following by the manipulator

The accuracy of trajectory following was tested by sending the planned trajectories for the 100 random configurations stated above to the manipulator via the inter-

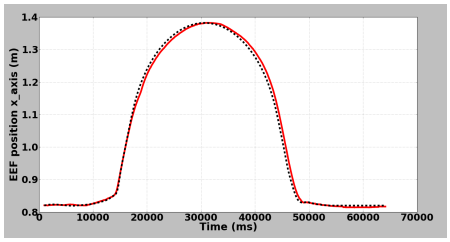

Fig. 7 Orion7P end-effector's X position while circular trajectory following. Black dotted line shows the commanded positions and solid red line shows the achieved position.

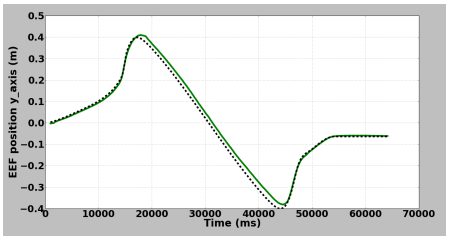

Fig. 8 Orion7P end-effector's Y position while circular trajectory following. Black dotted line shows the commanded positions and solid green line shows the achieved position.

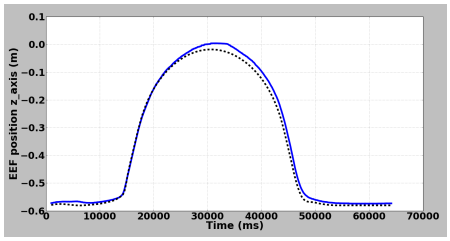

Fig. 9 Orion7P end-effector's Z position while circular trajectory following. Black dotted line shows the commanded positions and solid red line shows the achieved position.

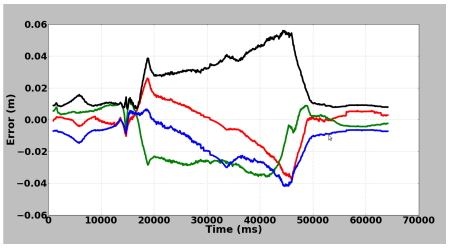

Fig. 10 Orion7P end-effector's position errors in euclidean space while circular trajectory following by the Orion7P arm. The red, green and blue plots represent errors along the x, y and z axes for paths traced in Figures 7, 8 and 9 respectively. The black plot represents cumulative position error in euclidean space.

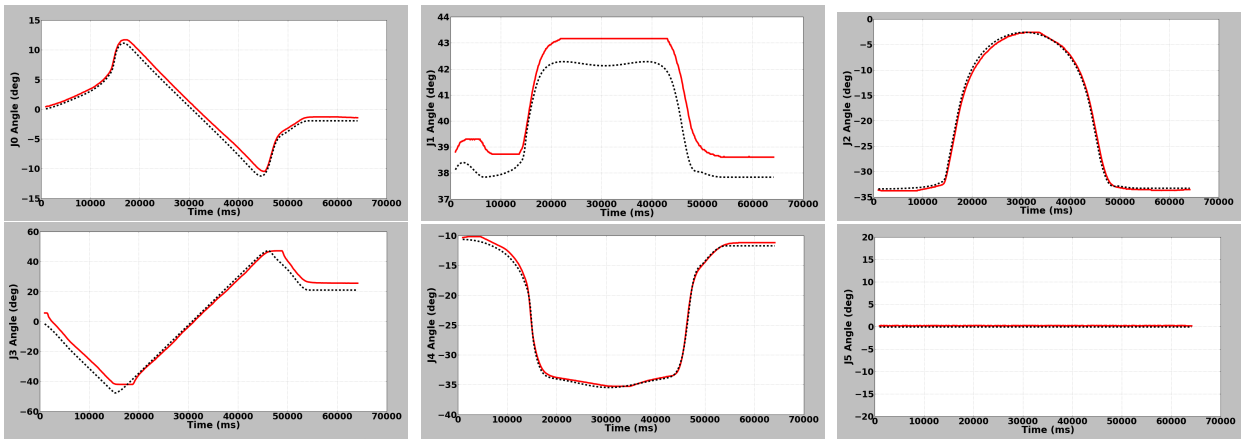

Fig. 11 Joint behavior while following a circular trajectory. Each plot represents one joint angle. Black dotted lines represent commanded position while red solid lines show the joint angles reached.

polation based controller. The joint positions read from the manipulator were logged at regular intervals and signify the achieved joint positions. The end effector (EEF) position was calculated for the sent and achieved joint configurations using the forward kinematics of the manipulator. The Euclidean distance between the sent and achieved EEF positions is defined as the error in trajectory following. For this experiment, the maximum joint speeds of 10, 10, 7.5, 5, 15 and 15 degrees per second (in order used in Table 1) were used. A mean trajectory following error of 2.387 cm with a standard deviation of 1.276 cm was achieved.

Further, the trajectory following capability of the controller was validated by making the EEF of the Orion7P arm follow circular and straight line paths in different directions. Straight line paths are particularly important for tool plugging operations in underwater environments (see [12]), while circular paths provide challenging test trajectories because of the involvement of all joints with changing directions of movements. The commanded EEF waypoints for one such circular and one such straight line paths are shown by starred points in Figure 5 and Figure 6 respectively. The EEF positions for trajectories generated by the controller and achieved by the arm are shown alongside by solid black and red lines. The position following along the x, y, and z Euclidean axes are plotted in Figure 7, Figure 8 and Figure 9 for the circular trajectory and in Figure 12, Figure 13 and Figure 14 for the straight line trajectory. The corresponding errors along the x, y, and z Euclidean axes are plotted in Figure 10

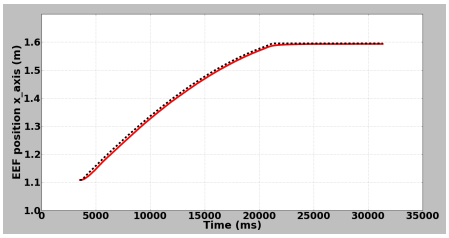

Fig. 12 Orion7P end-effector's X position while straight line trajectory following. Black dotted line shows the commanded positions and solid red line shows the achieved position.

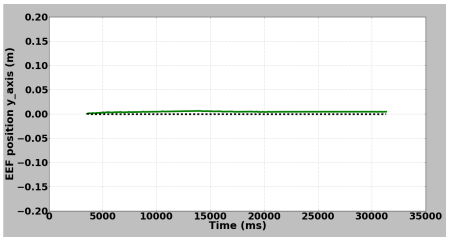

Fig. 13 Orion7P end-effector's Y position while straight line trajectory following. Black dotted line shows the commanded positions and solid red line shows the achieved position.

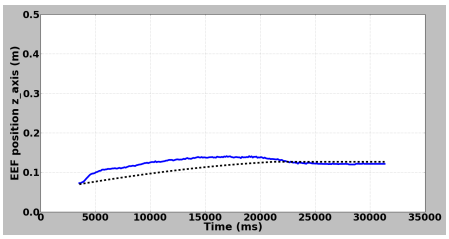

Fig. 14 Orion7P end-effector's Z position while straight line trajectory following. Black dotted line shows the commanded positions and solid red line shows the achieved position.

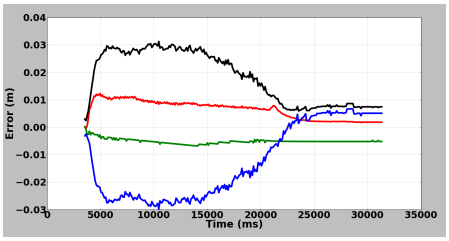

Fig. 15 Orion7P end-effector's position errors in euclidean space while straight line trajectory following by the Orion7P arm. The red, green and blue plots represent errors along the x, y and z axes for paths traced in Figures 12, 13 and 14 respectively. The black plot represents cumulative position error in euclidean space.

for the circular and in Figure 15 for the straight line trajectory. This shows a maximum position error of 4.1 cm along a single axis for circular path and 2.95 cm for the straight line path. The cumulative Euclidean position error is shown by the black line and denotes a maximum three-dimensional position error of 5.5 cm for the circular trajectory and 3.0 cm for the straight line trajectory. The sent and achieved joint positions are plotted in Figure 11 for the circular trajectory. A maximum error of 5 degrees was observed for circluar path.

3.4 Trajectory following at various speeds

For all experiments discussed above, maximum joint speeds of 10, 10, 7.5, 5, 15 and 15 degrees per second were used as mentioned in Section 3.3 In this experiment, the trajectory interpolator based controller was used for sending trajectories at different speeds to the Orion7P arm. The maximum speeds were specified as 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0 times the above specified maximum joint speeds. The circular and straight line trajectories shown in Figure 5 and Figure 6 were used to test the accuracy of trajectory following. The resulting position errors are shown in Figure 16 and Figure 17. As expected, the accuracy of trajectory following decreased with increasing speeds. We achieve a mean position error of 2.58 cm for the circular trajectory and of 1.56 cm for the straight line trajectory even for the 4.0 times speed test. For trajectory following at slower speeds the mean position error was 2.32 cm for the circular trajectory and of 1.52 cm for the straight line trajectory.

4. CONCLUSION AND FUTURE WORK

This paper presented a methodology for autonomous collision-free trajectory planning and following for industrial underwater manipulators. The methodology consists of a combination of existing state of the art trajectory planning algorithms from ground based robotics, and an existing computer interface for controlling industrial underwater manipulators. The Bi-directional Rapidly Exploring Random Tree algorithm was used for collision free path planning, maximum joint velocities were used for assigning timestamps, and a cubic interpolator was used for trajectory smoothing. Then the trajectory was sampled at the communication frequency defined by the computer interface and the resulting trajectory was sent to the manipulator. The experiments were conducted on a real industrial standard Orion7P manipulator from Schilling robotics. This interpolation based controller resulted in increased precision of less than two degrees per joint in reaching a desired arm configuration as compared to the four degree per joint precision attained while using only the lowest level position controller on the Orion7P manipulator.

The collision free path planner was consistently able to generate paths between starting and destination positions within seven seconds. The manipulator was successful in following the planned trajectories with an average end-effector position accuracy of 2.4 cm. Even for hard trajectories like circular paths and straight lines the average end-effector position accuracy of less than 3 cm was achieved which is acceptable for most underwater applications. Trajectory following was tested at higher speeds

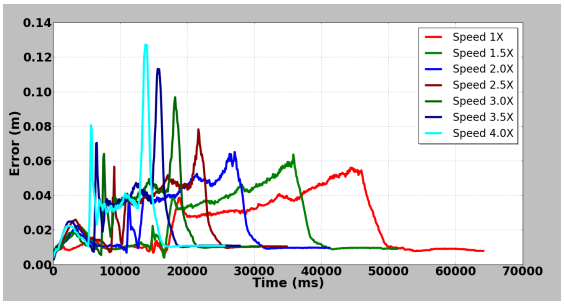

Fig. 16 Orion7P end-effector's position errors in euclidean space while following circular trajectories at different speeds.

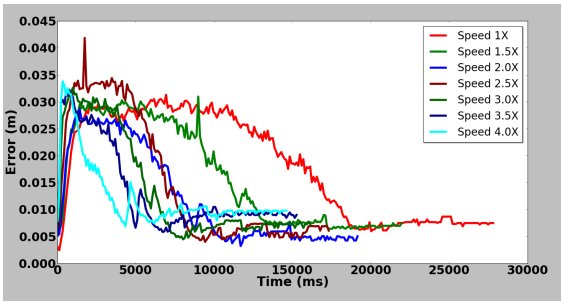

Fig. 17 Orion7P end-effector's position errors in euclidean space while following straight line trajectories at different speeds.

ranging between one to four times the joint speeds of 10, 10, 7.5, 5, 15 and 15 degrees per second. The accuracy of trajectory following decreased with increasing speeds and an average accuracy of 2.6 cm was achieved for the highest tested speed.

To our knowledge, this is the first demonstration of autonomous collision-free trajectory planning and following for a real industrial underwater manipulator system. The methods and results presented in this paper have proven that it is possible to adapt the advanced research in autonomous motion planning from terrestrial applications for application on industrial standard underwater manipulators. In future, the open loop interpolation based controller should be replaced with a closed loop speed controller. This will lead to increased accuracy of trajectory following. Other techniques for fast dynamic trajectory re-planning from terrestrial robots can also be applied to industrial underwater manipulators.

REFERENCES

[1] M. Hildebrandt, J. Albiez, and F. Kirchner, "Computer-based control of deep-sea manipulators," in *OCEANS 2008-MTS/IEEE Kobe Techno-Ocean*. IEEE, 2008, pp. 1–6.

[2] R. Rusu, I. Sucan, B. Gerkey, S. Chitta, M. Beetz, and L. Kavraki, "Real-time perception-guided motion planning for a personal robot," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 2009, pp. 4245–4252.

[3] S. S. Srinivasa, D. Ferguson, C. J. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M. V. Weghe, "Herb: a home exploring robotic butler," *Autonomous Robots*, vol. 28, no. 1, pp. 5–20, 2010.

[4] G. Marani, S. K. Choi, and J. Yuh, "Underwater autonomous manipulation for intervention missions auvs," *Ocean Engineering*, vol. 36, no. 1, pp. 15–23, 2009.

[5] J. Lemburg, P. Kampmann, and F. Kirchner, "A small-scale actuator with passive-compliance for a fine-manipulation deep-sea manipulator," in *OCEANS 2011 - Kona, Hawaii, 2011*, 2011.

[6] R. Diankov and J. Kuffner, "OpenRAVE: A planning architecture for autonomous robotics," Robotics Institute, Tech. Rep. CMU-RI-TR-08-34, July 2008. [Online]. Available: http://openrave.programmingvision.com

[7] K. I. Trovato, "Differential a*: An adaptive search method illustrated with robot path planning for moving obstacles and goals, and an uncertain environment," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 4, no. 02, pp. 245–268, 1990.

[8] Y. K. Hwang and N. Ahuja, "Gross motion planning a survey," *ACM Computing Surveys (CSUR)*, vol. 24, no. 3, pp. 219–291, 1992.

[9] S. M. LaValle, "Rapidly-exploring random trees a ew tool for path planning," 1998.

[10] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, 1996.

[11] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, April 2000, pp. 995–1001.

[12] M. Hildebrandt, J. Kerdels, J. Albiez, and F. Kirchner, "A multi-layered controller approach for high precision end-effector control of hydraulic underwater manipulator systems," in *OCEANS 2009, MTS/IEEE Biloxi-Marine Technology for Our Future: Global and Local Challenges*. IEEE, 2009, pp. 1–5.

[13] D. Mronga and A. Aggarwal, "Motion control of the humanoid robot aila," DFKI Robotics Innovation Center Bremen, Robert-Hooke-Strasse 5 28359 Bremen, DFKI Research Report, May 2013.