Virtual Submerged Floating Operational System for Robotic Manipulation
Qin Zhang, Jialei Zhang, Ahmed Chemori, Xianbo Xiang

To cite this version:

HAL Id: lirmm-01920187
https://hal-lirmm.ccsd.cnrs.fr/lirmm-01920187
Submitted on 13 Nov 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License
Research Article

Virtual Submerged Floating Operational System for Robotic Manipulation

Qin Zhang,1 Jialei Zhang,2 Ahmed Chemori,3 and Xianbo Xiang1,2,4

1 State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
2 School of Naval Architecture and Ocean Engineering, Huazhong University of Science and Technology, Wuhan 430074, China
3 LIRMM, CNRS-University of Montpellier, 161 rue Ada, 34392 Montpellier, France
4 Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen 518057, China

Correspondence should be addressed to Qin Zhang; qin.zhang@mail.hust.edu.cn and Xianbo Xiang; xbxiang@mail.hust.edu.cn

Received 18 August 2018; Accepted 28 October 2018; Published 12 November 2018

Guest Editor: Junpei Zhong

Copyright © 2018 Qin Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In this work, a virtual submerged floating operational system (VSFOS) based on parallel and serial robotic platforms is proposed. The primary aim behind its development lies in carrying out simulated underwater manipulation experiments in an easier and safer way. This VSFOS is consisted of a six-degree-of-freedom (6-DOF) parallel platform, an ABB serial manipulator, an inertial sensor, and a real-time industrial computer. The 6-DOF platform is used to simulate the movement of an underwater vehicle, whose attitude is measured by the inertial sensor. The ABB manipulator, controlled by the real-time industrial computer, works as an operational tool to perform underwater manipulation tasks. In the control system architecture, software is developed to receive the data collected by the inertial sensor, to communicate and send instructions. Furthermore, the real-time status of the manipulator is displayed in this software. To validate the proposed system, two experiments have been conducted to test its performance. In the first experiment, the test is carried out to check the communication function of VSFOS, while in the second one, the manipulator is intended to follow the movement of the parallel platform and perform simulated operational task in the space. The obtained results from these two experiments show clearly the effectiveness and the performance of the proposed VSFOS.

1. Introduction

As the ocean attracts great attention on environmental issues and resources, the need for and use of underwater operational systems become more and more demanding [1–3]. With significant advances of different types of mobile robots and manipulators in a bunch of application fields [4–9], marine robots including unmanned surface vehicles (USVs) [10–12], remotely operated vehicles (ROVs) [13, 14], and autonomous underwater vehicles (AUVs) [15–18] have been well developed in last decades [19–22]. Recently, submerged floating operational systems (SPOS), represented by underwater vehicles equipped with manipulators, have evolved from laboratory engineering prototypes into mature field tools for underwater operational applications [23–27]. As illustrated in Figure 1, SPOS can replace a human diver to perform underwater robotic tasks in harsh and dangerous environments, such as underwater salvage, and underwater pipeline inspection, ship maintenance [18, 28–31].

Owing to its great application perspectives, research and investigations concerning SPOS have already been launched all over the world [32, 33]. The Deep Submergence Operations Group developed the Jason ROV at the WHOI Deep Submergence Laboratory in 1989 [34]. Equipped with a custom-designed 5-axis robotic arm, this ROV is able to perform a wide variety of scientific tasks including temperature measurements and underwater deployment, as well as sampling of sediments and biological organisms. Within the European project TRITON, a manipulator installed on the autonomous underwater vehicle was used to open/close a valve [35]. The Nereus vehicle designed by Deep Submergence Lab at WHOI, USA, in 2008 was carrying a six-degree-of-freedom (DOF) electrohydraulic robot arm to conduct the benthic survey operations [36, 37]. The Polar-ARV designed by the
Shenyang Institute of Automation (SIA) can work as a moving underwater platform for monitoring marine environment under ice [38].

Although these applications of SFOS have already been so extensive, there are still a lot of limits when it comes to the experimental research and system test of SFOS. These limitations are as follows: First of all, based on underwater operational experiments of SFOS, it may cost a lot of time and money as well as many labor force. Secondly, suitable environment conditions such as weather condition will be required for the success of the underwater operations. Thirdly, harsh and dangerous undersea environments with high uncertainty may render the experiments very delicate to be carried out [39–43].

In order to help researchers performing all kinds of underwater operational experiments, a virtual submerged floating operational system (VSFOS) was built up at the school of Naval Architecture and Ocean Engineering in Wuhan, China. As illustrated in Figure 2, the VSFOS is mainly composed of four parts, the parallel kinematic platform, the ABB serial manipulator, the inertial navigation sensor, and the real-time industrial computer. The ABB manipulator is installed on the upper surface of the parallel platform executing an operational intervention tool. At the same time, the 6-DOF platform completes its translational and rotational motions in the 3D space to simulate the movement of the underwater vehicle and environmental disturbances. The inertial navigation sensor is mounted on the moving plate of the parallel platform to monitor its attitude change. The main control part of the VSFOS is a real-time industrial computer, responsible for providing instructions to the ABB serial manipulator. In fact, the underwater vehicle and carrying manipulator are disturbed with complex underwater environment. The manipulator and the platform are joined rigidly in underwater operation system and VSFOS. And the ocean disturbance on the manipulator robot is difficult to simulate actively. Hence, for the control scheme of the manipulator, the disturbances from underwater environment and platform can be equivalent to the slow disturbance from the platform. In this sense, the proposed VSFOS is used to replace the underwater vehicle and support the manipulator. Yet, for the manipulator, the parallel platform is used to generate all external disturbances to the manipulator. An inertial navigation sensor is mounted on the lower surface of the 6-DOF platform to monitor its attitude change. The main control part of our VSFOS is a real-time industrial computer, responsible for providing instructions to the ABB serial manipulator.
In terms of control point of view, two critical questions may arise. The first one deals with the communication problem with the ABB manipulator, where a lot of efforts have been made to break the communication blockade of this industrial manipulator to be able to have access to the low level control. Indeed, commercial industrial robotic manufacturers typically provide a joystick or a static off-line programming interface for the customers, where the control type is mainly appropriate for simple and respective tasks, but not for the VSFOS since a sophisticated real-time control of the manipulator is necessary [44]. To overcome this problem, we developed control software with an open monitoring interface [45], thanks to which, the communication bridge between the manipulator and the industrial computer was successfully built up.

The second critical question is how to simulate the underwater operational tasks making use of the hardware equipment of the VSFOS. Controlling the 6-DOF parallel platform and the ABB manipulator separately is a relatively easy task. However, when it comes to controlling the manipulator to do some accurate operations while its base is following the movement of the parallel platform, the task will be much more challenging. To achieve this function, two requirements ought to be met. Firstly, the attitude change of the parallel platform should be measured in real-time; secondly, the manipulator ought to compensate the movement of the platform in real-time. Indeed, the inertial navigation sensor can work under the condition of high speed, and control software was developed to perform the real-time control of the manipulator. In this way, the two requirements are fulfilled. As a result, the VSFOS can be used to conduct the simulated underwater experiments in a laboratory environment.

To sum up, this paper mainly introduces a new virtual submerged floating operational system (VSFOS) designed to make it easier and safer to conduct underwater operational simulations. The organization of this paper is as follows. In Section 2, a detailed description of our VSFOS, including hardware and software, is presented. The solution to the two critical questions regarding the control of the VSFOS system is detailed in Sections 3 and 4. In Section 5, simulation and real-time experiments are presented and discussed. The last part presents the concluding remarks and future works.

### 2. System Architecture

In this section, the hardware framework of VSFOS is presented firstly with a description of each component, and then the connections of all the hardware equipment are presented in the software description part.

#### 2.1. Description of the Hardware

VSFOS consists of five main equipment pieces, the six-degree-of-freedom (6-DOF) parallel robotic platform, the ABB serial robotic manipulator, the manipulator controller, the inertial navigation sensor, and the real-time industrial computer. The overall architecture of the system is illustrated in Figure 3.

The 6-DOF parallel platform is fixed on the ground with three bolts. A laptop is connected to the platform through an ethernet cable to send the control commands to drive the platform. The ABB manipulator is mechanically attached to the moving base of the parallel platform by three positioning screws. Consequently, the base of the serial manipulator follows the movement of the 6-DOF parallel platform. The inertial navigation sensor is fixed on the upper surface of the platform to measure in real-time the orientation of the platform. The information collected by this sensor is transmitted to the real-time industrial computer through a USB data cable. The industrial computer and the manipulator controller are connected through another ethernet cable, thanks to which, the control instructions can be sent to the manipulator controller and the real-time status can be fed back to the computer.

The features description of the ABB manipulator, the 6-DOF platform, and the inertial navigation sensor is summarized in Table 1.

The ABB manipulator IRB 1600 is a 6-axis industrial manipulator produced by ABB Company. This manipulator is equipped with a strong power system, and its end-effector can carry a payload do up to 10 Kg; moreover, it is also suitable for high precision operations, since its positioning accuracy is 0.35 mm and its repeatability is 0.05 mm. Owing to these characteristics, the IRB 1600 is widely used in many industrial tasks, like arc welding, packaging, and material handling. Taking advantage of its strong power and high precision, we make it an operational tool in the proposed VSFOS.

The 6-DOF parallel platform created by MOTUS Company is a six-degree-of-freedom parallel kinematic machine (PKM). Its six legs are composed of electric cylinders, equipped with six AC servo motors inside the legs to control the piston rod of the electric cylinders. This platform can be typically used to provide a simulation environment for automobile driving, aircraft piloting, and some other entertainment devices. Therefore, we combine its translational and rotational motions to simulate the movement of underwater vehicles in VSFOS, as illustrated in Figure 16.

<table>
<thead>
<tr>
<th>Equipment</th>
<th>Composition</th>
<th>Industrial applications</th>
<th>Role in VSFOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABB manipulator</td>
<td>Manipulator/Controller</td>
<td>welding/assembly handling</td>
<td>Operational tool</td>
</tr>
<tr>
<td>6-DOF parallel platform</td>
<td>electric cylinders/AC servos/motor/steel platform</td>
<td>virtual reality for simulated driving cars, aircraft</td>
<td>operational: Simulates the underwater vehicles</td>
</tr>
<tr>
<td>Inertial navigation sensor</td>
<td>gyroscope/accelerometer/thermometer</td>
<td>automobile/aircraft/ocean</td>
<td>measures the movement of the 6DOF platform</td>
</tr>
</tbody>
</table>

*Table 1: Hardware features of the VSFOS.*
The INS (inertial navigation sensor) IG-500A is used to accurately measure the attitude angles of the VSFOS. Based on the MEMS (Microelectromechanical Systems) technology, the INS IG-500A can calculate the precise 3D direction data by 3 gyroscopes, 3 accelerometers, 3 magnetometers, and a thermometer. It is commonly used in automobile, aerospace, and other vessels. In our VSFOS, the INS IG-500A is used to measure the attitude angles of the 6-DOF parallel platform.

2.2. Software Description. In order to control the whole system and monitor it in real-time, control software programmed in C SHARP language is developed. Taking advantage of this software, different functions have been developed for the following purposes:

(i) receiving the information collected by the inertial navigation sensor through serial port communication;
(ii) planning the trajectory for the serial manipulator by combining the received data, the manipulator kinematics, and the manipulator motion constraints;
(iii) transmitting motion instructions to the IRC5 controller and get the real-time status of the manipulator back through network communication.

In summary, three main functions of the software which ensure well operating conditions are the serial port communication, the manipulator control, and the network communication, as illustrated in Figure 4.

The sensor sticking to the upper moving base of the 6-DOF parallel platform can measure the attitude angles of the platform. The collected data will then be sent to the industrial computer through the serial port, after being filtered and analyzed. The attitude measurement information will be parsed out from the processed data. As illustrated in Figure 4, the letter S refers to the processed data, and the letter T denotes the calculated homogeneous transformation matrix. Once the 6-DOF parallel platform starts moving, the manipulator ought to follow the motion. In order to make each joint of the manipulator moving to the target position smoothly, we need to plan carefully the reference joint trajectory for the mobile manipulator. Finally, the joint angles instructions newly generated will be sent to the controller through the network communication; meanwhile, the real-time status of the manipulator will be fed back to the industrial computer.

2.2.1. Serial Port Communication. The data collected by the inertial navigation sensor are transmitted to the industrial computer through the serial port communication. The received data should be packaged in a certain format.
according to the communication protocol; thus, we can parse and analyze it according to this protocol. The communication protocol is described in Table 2.

As shown in Figure 5, the serial port communication unit is displayed at the bottom of the control software interface. To receive the collected data, we need to choose the proper port and bound rate. Then by clicking the “Open serial port” button, the software will parse the data and display it in the corresponding frame. For example, the “Angle data” frame tells the Euler angle of the platform relative to the base coordinate system as well as its linear velocity and angular acceleration; the “Angle initialization” button is set to define...
2.2.2. Manipulator Control. The manipulator control scheme gives joint angles instructions to joint motors to make the end-effector reach the right position and orientation. Therefore, it is necessary to set a target value for each joint and monitor the real-time position and orientation of the end-effector. As shown in Figure 5, in the manipulator control unit, the “Operational panel” frame is used to choose the planning trajectory for the manipulator, and the “Position data” and “Joint angles” frames are designed to display the position and orientation information of the TCP (tool center point). In addition, there is a “Manipulator data” frame, in which the target and current position and joint angles of the manipulator are displayed.

2.2.3. Network Communication. In order to perform the real-time control of the manipulator, we choose to communicate with the manipulator controller through the network communication. As illustrated in Figure 5, the network communication part is located at the top of the interface. Once starting the software, by clicking the “Connect” button, it will search the on-line manipulator controller automatically. There are two pieces of information to identify the correct controller, which are the “system name” and the “IP address”. Once the connection with the manipulator controller is set, we get access to exchange data with the manipulator controller. In other words, we can write variables to the controller to update the position and orientation of the end-effector. Meanwhile, the system variables are available to observe the real-time status of the manipulator, which include joint angles and the position information of the TCP.

3. Manipulator Kinematics

The ABB serial manipulator IRB 1600 is a robotic arm with six rotational joints, equipped with an operation tool installed at the end joint. In order to get the position and orientation of this TCP (tool center point) with respect to the base, it is necessary to model the manipulator mathematically. The universal way to model a manipulator is based on the Denavit-Hartenberg (D-H) method [46]. The D-H method introduces the four steps of manipulator model, which are manipulator modeling, forward kinematics, inverse kinematics, and path planning.

3.1. Manipulator Modeling. The first step in the Denavit-Hartenberg (D-H) method is to build the D-H model of the manipulator; this model contains two parts, namely, the joint diagram of the manipulator and the D-H parameters table. The joint diagram can be drawn according to the joint features and mechanical structure parameters, as illustrated in Figure 6.
Based on the joint diagram and the products brochure of the ABB manipulator IRB 1600, the key parameters can be listed in Table 3. The key parameters are reconfirmed by manoeuvring the manipulator and observing the data feedback on the host computer.

Four important D-H parameters used in D-H method, namely, \(a_i\), \(\alpha_i\), \(d_i\), \(\theta_i\), are explained in Table 4.

### 3.2. Manipulator Forward Kinematics.

#### 3.2.1. Manipulator Inverse Kinematics.

In the general sense, the inverse kinematics of a manipulator means calculating the joint angles when the position and orientation of the end-effector is given. This problem is much more complex than the forward kinematics due to the fact that there often exists more than one set of joint angles that satisfy the given position and orientation. For the case of the IRB 1600 manipulator, on the premise of knowing the transformation matrix \(A_0\) between the base center and the TCP of the manipulator, the process of solving the inverse kinematics equation to get the joint angle of joint 1 is as follows.

\[
0^A_6 T = \begin{bmatrix}
  x_n & o_x & a_x & p_x \\
  y_n & o_y & a_y & p_y \\
  z_n & o_z & a_z & p_z \\
  0 & 0 & 0 & 1
\end{bmatrix}
\]

The elements \(n_{x,y,z}, o_{x,y,z}, a_{x,y,z}\) refer to the attitude of the operational tool, and the element \(p_{x,y,z}\) refers to its position with respect to the base. Similarly, another expression of the transformation matrix \(A_0^A T\) is as follows.

\[
T = A_1 \times A_2 \times A_3 \times A_4 \times A_5 \times A_6
\]

Multiplying both sides of (4) by the inverse of the matrix \(A_1\), we get the following.

\[
A_1^{-1} \times T = A_2 \times A_3 \times A_4 \times A_5 \times A_6
\]

Calculating both sides of (5), the matrix on the left side of the equation equals the one on the right. Thus, we get the following.

\[
\theta_1 = \pm \arctan \frac{p_y - 650o_y}{p_x - 650o_x}
\]

From (6), it can be seen that the joint angle \(\theta_1\) has two solutions.

\[
\theta_1 = \arctan \frac{p_y - 650o_y}{p_x - 650o_x}
\]

\[
\theta_1 = \arctan \frac{p_y - 650o_y}{p_x - 650o_x} + \pi
\]
In the same way, the rest of joint angles from $\theta_2$ to $\theta_6$ can be computed. Hence, there are multiple sets of solution for the manipulator robot. And there is a great deal of difference between the two solutions for each joint angle. In the numerical control process, the solution which has smaller difference with the joint angle in the last control period is preferred. Hence, the optimization principle can be ensured.

### 3.3. Path Planning

Path planning, also referred to as trajectory generation, is the act of describing the desired motion of a manipulator in multidimensional space over a period of time. To decrease the vibration of the manipulator at the initial and final points, it is very common to use cubic polynomials interpolation methods to generate point-to-point trajectories of the joint angles. Taking the current joint angles as the initial value and the joint angles calculated by solving the inverse kinematics equations as the target value, the planned joint angles expression with respect to time can be expressed by the following.

$$\theta_i(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3, \quad i \in \{1, 2, 3, 4, 5, 6\} \quad (8)$$

Four constraints about the initial and final conditions of the joint angles are used to compute the coefficients of (8). These constraints are expressed by the following formula.

$$\begin{align*}
\theta_i(0) &= \theta_{0i}, \\
\dot{\theta}_i(0) &= \dot{\theta}_{0i}, \\
\theta_i(t_f) &= \theta_{fi}, \\
\dot{\theta}_i(t_f) &= 0.
\end{align*} \quad (9)$$

By solving (8) and (9), we can evaluate the designed joint angles at any sample time.

### 4. Communication with the Robotic Manipulator

For the ABB manipulator, a simulation and off-line programming software, RobotStudio, are provided by the manufacturer. Programming with RAPID language and downloading it to the manipulator controller are allowed by using the RobotStudio software. Indeed, this might be the simplest method to achieve the communication with the manipulator compared with other manual control methods. However, it is not suitable for the VSFOS, due to the high real-time control demands of the VSFOS. Therefore, getting rid of the traditional control method for the manipulator and breaking the communication blockade of this industrial manipulator are demanded. During our research and development of VSFOS, three methods have been tested to communicate with the manipulator controller, which are “read/write files”, “socket communication”, and “develop software using PC SDK”; detailed descriptions of these methods are presented in the following paragraph.

#### 4.1. Read/Write Files

Read/write files make use of files to transmit information between the industrial computer and the manipulator controller. At the computer side, a “.doc” file is edited with control instructions inside. Meanwhile, at the controller side, a RAPID program is edited to receive and parse control instructions. Similarly, the current status of the manipulator can be written in another file and then fed back to the industrial computer. To realize the “sending” function, we take advantage of RobotStudio software, thanks to which, the industrial computer can get the “write access” of the manipulator controller. Detailed description of this communication architecture is illustrated in Figure 7.

The control variables of the manipulator, such as the “jointtarget” variables (joint angle) and the “robtarget” variables (position of TCP), are written in those files and then transmitted from the computer to the controller. Therefore, with RobotStudio software, conducting the transmission job, the values of these variables can be changed by reading/writing these files. Indeed, control instructions can be sent to the controller and the status information of the manipulator can be fed back. The principle of this method is very simple; however, when it comes to the problem of real-time constraints, this method cannot meet the needed requirements.

#### 4.2. Socket Communication

This method ensures the communication function based on the client/server mode. The RobotStudio software installed on the industrial computer works as a client, and the manipulator controller connects to the industrial computer with an ethernet cable acting as a server. When the program starts, the client will first request for the “write access” from the server and then send motion instructions to the server periodically after obtaining the “write access”. On the server’s side, the controller will get these instructions to control the manipulator; meanwhile, real-time status of the manipulator will be fed back to the client; the specific process of the socket communication is illustrated in Figure 8.

It is necessary to edit programs on both RobotStudio software and the controller if this method is applied to realize the real-time control of the manipulator. Definitely, this method
allows us to control the manipulator in real-time; however, we cannot get rid of this RobotStudio software which does not provide an external device input interface. In other words, reading the data sent from the inertial navigation sensor with this software is not available; therefore, this method is also not suitable for our VSFOS.

4.3. Developing Software Using PC SDK. PC SDK (Software Development Kit) allows end-users to add their own customized operator interfaces for the IRC5 controller. Hence, users can control the manipulator over a network from a tailored application on a PC. Furthermore, this application can be developed with any programming language in any development environment. In our case, we developed a control software application written in C Sharp language in Visual Studio compiling environment. This software makes use of the API functions provided by the ABB Company to get access to the manipulator controller very easily; the proposed solution is illustrated in Figure 9.

The classes used to access to the manipulator controller functionality together make up the controller API, in which there are many functions that can be used to obtain or change the value of the control variables of the manipulator. A part of the controller API object model is shown in Figure 10. Instead of requesting the “write access” in advance, the control software can send instructions to the manipulator directly by invoking the API functions, as illustrated in Figure 10. Having connection with the controller, going through the "Rapid-Task-Module-Routine-RapidData" route, we can get access to the "RapidData". This last one model stores the information of the control variables of the manipulator. Hence, we can read the values of the variables to get the real-time information about the manipulator and send instructions to control the manipulator by writing new values for
the variables. The software interface is very simple and easy to operate with many integrated functions, making use of which we can truly realize the real-time control of the manipulator. Not restricted by the given RobotStudio software, as well as the programming language, this method is confirmed to be the best one to ensure the communication with the manipulator compared to the above two methods.

5. Simulation and Experimental Results

For the sake of validation of the performance of the proposed solution, two experiments have been carried out, one of which is a simulation test to demonstrate the network communication and function of the control software, while the second one is performed in the laboratory with the real VSFOS working in 3D space.


In this simulation test, the control software and the RobotStudio software are running on the same industrial computer. The RobotStudio software is used to build the manipulator model and create a virtual controller for IRB1600. The default IP address of the virtual controller is “127.0.0.1”, and for distinction, we change its system ID to "manipulator-1". The control software can be connected to the manipulator by searching the on-line controllers. Once the virtual controller is found, the network connection will be established after confirming the IP address and the system ID.

As shown in Figure 11, when the communication bridge between the controller and the control software is built up, the button name will change from “Connect” to “Connected”. At the same time, a prompt box will pop up to inform the user of the connection status. Since the communication is unblocked, motion instructions can be sent to the controller. A particular geometrical path for the end-effector of the manipulator beforehand and the path function is expressed as follows.

\[
\begin{align*}
x &= 800 + 200 \sin \left( 2t + \frac{\pi}{2} \right) \\
y &= 200 \cos \left( 2t + \frac{\pi}{2} \right) 
\end{align*}
\]

(0 ≤ t < 100) \hfill (10)

\[
\begin{align*}
x &= 1000 - 100(t - 100) \\
y &= - \frac{10t}{\sqrt{3}} 
\end{align*}
\]

(100 ≤ t < 130) \hfill (11)

\[
\begin{align*}
x &= 700 \\
y &= -100\sqrt{3} + 10\sqrt{3}t 
\end{align*}
\]

(130 ≤ t < 150) \hfill (12)

\[
\begin{align*}
x &= 700 + 10t \\
y &= 100\sqrt{3} - 10\sqrt{3}t 
\end{align*}
\]

(150 ≤ t < 180) \hfill (13)

\[
\begin{align*}
x &= 1000 - 7.5t \\
y &= -7.5t \sqrt{3} 
\end{align*}
\]

(180 ≤ t < 200) \hfill (14)

\[
\begin{align*}
x &= 800 + 100 \sin \left( 2t + \frac{5\pi}{6} \right) \\
y &= 100 \cos \left( 2t + \frac{5\pi}{6} \right) 
\end{align*}
\]

(200 ≤ t < 300)

The corresponding trajectory is illustrated in Figure 12. In order to get a better observation of the simulation results, we tracked the actual trajectory of the TCP. This trajectory is able to be displayed on RobotStudio software in real-time, as illustrated in Figure 13.

While conducting the simulation test, it can be seen from the RobotStudio software that the manipulator TCP moved along the designed path very accurately and quickly. In addition, the actual end trajectory is quite similar to the designed one according to Figures 12 and 13. In order to demonstrate the network communication function of the control software, further analysis of the control result is necessary. Since the motion instructions are distributed to
the joint motors with a series of joint angles, it is intuitive to compare the planned joint angles to the actual joint angles. Therefore, we collect the planned joint angles data and the actual joint angles data which are recorded in the simulation test. The joint 2 and joint 3 with quick movement in the test are taken as an example. A time plot of these joint angles is presented in Figure 14, and the time plot for the tracking error of the joint angles is displayed in Figure 15.

From Figures 14 and 15, it can be seen that the joint angles curves of the planned trajectory and the actual one are almost
overlapping and the tracking errors of the joint angles of axis 2 and axis 3 are less than 0.12°. Therefore, we can conclude that the manipulator moves in strict accordance with the planned joint angles; this also means that the generated instructions have been correctly transmitted to the controller, which can prove that the network communication is well working experiment.

5.2. Experiment Scenario: 3D Space Operation Test. Since the research of underwater docking is becoming a timely research topic recently, then before we put the manipulator on the 6-DOF parallel platform, we conduct a 3D space operation test in our laboratory with the current system. This 3D space operation test is carried out to test the performance of the hardware equipment of VSFOS while performing the simulation operations of underwater docking. In this test, a marking pen is installed at the end-effector of the serial manipulator as an operation tool, and the 6-DOF parallel platform rotates along its central axis to simulate the underwater vehicle which is disturbed by the current and waiting for docking. In the end, the marking pen will draw circles in the corresponding plane to which the 6-DOF parallel platform is moving, as illustrated in Figure 16. The operational tool is installed on the mobile vehicle platform. And the operating objective is a fixed underwater vehicle. Since the INS cannot accurately measure the linear movements in 3 axes, the underwater salvage scenario is equivalently translated into the experiment scenario as shown in the right subgraph of Figure 16. The operational tool, underwater vehicle, and underwater interfacing are simulated with ABB manipulator, 6-DOF platform, and drawing circles, respectively. The relative movement situations of simulation scenario are similar to the actual underwater scenario.

When the 6-DOF parallel platform starts moving, the inertial navigation sensor measures the attitude change of the platform, namely, the roll and the pitch angles of the plane that the 6-DOF parallel platform changed to. Since the inertial navigation sensor cannot measure the position change information, we make the central point of the 6-DOF parallel platform remain unchanged. As a result, we get the new position and orientation information about the
target points of the operation tool of the manipulator. At the beginning, when the 6-DOF parallel platform stays in the horizontal plane, the manipulator is designed to follow the planned circular trajectory. The coordinates of its central point are denoted by O (a, b, c), and the radius of the planned circle is R. If we consider a point M on this circle, the position coordinates of this point M (X, Y, Z) can be expressed as follows.

\[
X = a + R \sin(\theta) \\
Y = b + R \cos(\theta) \\
Z = c
\]

\((0 \leq \theta \leq 2\pi)\) (13)

When the platform rotates JX degrees along its X axis, JY degrees along its Y axis, and JZ degrees along its Z axis, the upper surface of the platform changes from the horizontal plane to plane 1, whose yaw, pitch, and roll angles are JX degrees, JY degrees, and JZ degrees. Similarly, point M will change to point M’; the position coordinates of point M’ (X’, Y’, Z’) are as follows.

\[
X' = R \cos(\theta) \cos(JY) + a \\
Y' = R \sin(\theta) \sin(JX) \sin(JY) + R \cos(\theta) \cos(JX) + b \\
Z' = R \sin(\theta) \sin(JY) \sin(JX) - R \cos(\theta) \sin(JX) + c
\]

\((0 \leq \theta \leq 2\pi)\) (14)

From (13) and (14), it is obvious that as long as we get the position coordinates of the central point and the attitude change information, we can calculate the corresponding designed trajectory in plane 1. In this test, considering the height of the 6-DOF parallel platform, we set the position coordinates of the central point of the planned circle as (900mm, 0, 700mm), and the radius is set to 100mm; thus, the trajectory equations of the TCP in plane 1 are as follows.

\[
X = 100 \cos(\theta) \cos(JY) + 900 \\
Y = 100 (\sin(\theta) \sin(JX) \sin(JY) + \cos(\theta) \cos(JX)) + 700 \\
Z = 100 (\sin(\theta) \sin(JX) \sin(JY) - \cos(\theta) \sin(JX))
\]

\((0 \leq \theta \leq 2\pi)\) (15)

By collecting the planned and actual position information of the TCP, the results are displayed in Figures 17 and 18.
In the first stage of this experiment, the platform remains still; thus, the first circle is drawn in the horizontal plane, i.e., plane 1. Then, the platform rotates 0 degrees along the X axis and 34.6 degrees along the Y axis. As a result, the manipulator adjusts the attitude of the marking pen and draws a circle in plane 2. In stage 3, the platform is rotated 41 degrees along the X axis and 0 degrees along the Y axis. And the last circle is drawn in plane 3, as illustrated in Figure 19.

According to test results in Figure 19, the TCP of the manipulator moves in a well accordance with the planned trajectory when drawing the circles in 3D space, except for the moments when the platform changes its attitude suddenly. At those transient instants, the tracking process is a little delayed. To take a further analysis of the reasons, we can make use of the collected joint angles information.

Since the variation range of joint 5 is the biggest one and joint 1 is the smallest, then, we chose to take the angles of joint 5 and joint 1 as an example; a time plot of these joint angles is shown in Figure 20, and their tracking errors are illustrated in Figures 21 and 22.

As presented in Figures 19 and 20, the joint angles of the actual trajectory are almost overlapped with the planned trajectory. And the joint angles can be fast and stable track to complete the entire trajectory. As it can be observed in Figures 21 and 22, the tracking error in three interim instants is bigger than the others.
Figure 20: Joint angles of the planned trajectory and the actual trajectory.

Figure 21: The tracking error of the joint angles of joint 1.
It is worth noting, from these two figures, that at the instant we change the attitude of the platform, the joint angles of the planned trajectory change quickly, while the actual ones are slightly delayed. Concerning the control cycle of the control software and the velocity of the TCP, the attitude changing speed of the platform is much faster than the moving speed of the manipulator. At the moment when the platform changes its attitude, the inertial navigation sensor detects that change information; as a result, a new trajectory in a new plane will be generated. Meanwhile, the manipulator is still following the orders from the previous control cycle and moving in the previous plane; therefore, the joint angles of the planned trajectory are very different from those of the actual trajectory at this instant. However, except for these three moments, an average value of the tracking error of the joint angles is mainly less than 0.7°, which is acceptable when taking into account the measuring errors. The obtained experimental results show clearly that the proposed equipment hardware of VSFOS is working well in 3D space manipulation tasks.

6. Conclusion and Future Work

This paper mainly introduces a new virtual submerged floating operational system (VSFOS), which makes it faster and much safer to conduct underwater operational simulations. The hardware framework and software structure of VSFOS are introduced in detail. And two critical questions regarding the manipulator kinematics and the communication with the manipulator are highlighted. In the end, two experiments have been conducted to demonstrate the effectiveness of the proposed VSFOS.
The developed control software and the RobotStudio modeling software are combined together to test the developed system hardware and software framework. The test results have shown that the end-effector of the manipulator can follow the desired trajectory with developed system framework. An underwater salvage simulation test is then conducted. The end-effector of the manipulator can move according to specified trajectory on the mobile operational interface.

In the future, some follow-up works demand further research, namely, improving the operational precision and reducing the response time of VSFOS. Some operational tools will be designed and installed on the end-effector of the manipulator to perform difficult subject.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request with guarantee for self-use and not sharing with other parties.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (Grant 91748113 and Grant 51579111), in part by the National Key Research and Development Program of China (Grant 2017YFB1302302), in part by the Shenzhen Science and Technology Plan Project (Grant JCYJ20170413131304568), and in part by the Research Fund from Science and Technology on Underwater Vehicle Technology (Grant SXJQR2017KFJJO6).

References


