Proceedings of The 2016 IEEE International
Conference on Real-time Computing and Robotics
June 6-9, 2016, Angkor Wat, Cambodia

# Motion Sensing Based Framework for Robot Manipulation

Hao Deng[1,2], Zeyang Xia[1,2,*], Shaokui Weng[1,2], Yangzhou Gan[1,2], Peng Fang[1,2] and Jing Xiong[1]

1. Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
2. Laboratory of Human-Machine Intelligence-Synergy Systems, CAS, Shenzhen, China
* Author to whom correspondence should be addressed
Email: zy.xia@siat.ac.cn; Phone: +86-755-8639-2218

***Abstract*— To data, outside of controlled environments, robots have normally performed manipulation tasks when operated with human. This pattern requires operators with high technical skills training and targeted knowledge acquiring for varied teach-pendant operating system. Motion sensing technology, enabling human-machine interaction in a novel and natural user interface using gestures, crucial inspires us to adopt a user-friendly and straight-forward interaction mode on robot manipulation. Thus, in this paper, we presented a motion sensing based framework for robot manipulation, which recognizes gesture commands captured by motion sensing input device and drives the action of robots. For compatibility, a general hardware interface layer was also developed in the framework. Simulation and physical experiments have been conducted for preliminary validation. The results have shown that the proposed framework is an effective approach for general robot manipulation with motion sensing control.**

***Index Terms*— Human-machine interaction, motion sensing, gesture recognition, robot manipulation**

## I. INTRODUCTION

Robots have found an increasingly wide utilization in all fields, they lift massive objects, move with blurring speed, repeat complex performances with unerring precision and manipulate with high dexterity. Robots have long been imaged as mechanical workers, cooperating with or even replacing people [1]. Yet the situation is, to data, robots have been very successful at manipulation in controlled environments such as in a factory or a laboratory. When outside of the controlled environments, they have normally performed manipulation tasks when operated with human [2]. Although the latest robotic researches would handle the situations with a sophisticated sensing and intelligent system [3], most robots are still under the typical teach-pendant operating pattern to get knowledge about current environment configurations.

Typical teach-pendant is a hand-held control unit equipped with buttons or joysticks to manually send the robot to desired positions, a screen to display the robot states, and a large red emergency stop button. In teach-pendant process, operators need to stand in certain nearby area, hold the control unit and move robot with buttons or joysticks, meanwhile, special focus should be paid on to avoid collision and identify the arrival of targeted position, shown in Fig.1-(a). As a result, this pattern is not only time consuming when manipulation task changes, but also requires operators with high technical skills training, and hand-eye coordination, otherwise, it would be so extremely easy to lead serious collision damage to the robots or equipment, and even heavy casualties. Differing from the teach-pendant operation in an active mode, lead-by-the-nose is a passive technique. In which, controller will de-energize the robot joints and allow users to drag and move the robot by hand to the desired positions or even paths while logging these position data [6], shown in Fig.1-(b). However, both techniques are hardware-dependent and controller-specific, which definitely increases the learning and training costs when adopting new types of robot system. In the final analysis, all these issues come down to a matter of user-friendly but general human machine interaction interface.


(a) (b)

Fig. 1. Typical patterns to get knowledge about current environment configurations. (a) teach-pendant operation with hand-eye coordination and fully focus [4], (b) "lead-by-the-nose" operation for paint spraying [5].

Recent researches have shown that the motion sensing technology enables human-machine interaction in a novel and natural interface using gestures or spoken commands. And the latest revolutionary motion sensing devices, with the represent of Kinect [7] and Leap Motion [8], are boosting wider applications of motion sensing technology from gaming to robotics[9, 10].

Some approaches are put forward that use the motion sensing data to control the action of a robot where the robot will initiate the human active movements, or even learn actions from human [11]. In practice, this motion sensing

*This work was supported by the National Science Foundation of China (51305436), Guangdong Natural Science Foundation for Distinguished Young Scholars (2015A030306020), Major Project of Guangdong Province Science and Technology Department (2014B090919002), Shenzhen High-level Oversea Talent Program (KQCX20130628112914284) and Fundamental Research Program of Shenzhen (JCYJ20140901003939038).

978-1-4673-8959-4/16/$31.00 © 2016 IEEE

is usually at the crossroad between gesture recognition and skeleton tracking. Sometimes, we only interested in the pose of hand to directly drive end effector of the robot, or we may also need to command the robot in a jointed mode which requires the joint angles date captured from skeleton tracking. In previous studies, numbers of works have been proposed on use of motion sensing input devices for robotic applications [12]. However, most of them were focusing on specific cases, which usually integrated a certain type of motion sensing input device, Kinect, Leap Motion, or et al, for a targeted robot system configuration. In this paper, we proposed a motion sensing based framework to connect and work between any available motion sensing input devices and robotic systems. Our study is aimed to contribute a modular designed, friendly plugging and configuring framework for general robot manipulation.


Fig. 2. Most wildly used motion sensing input devices. Microsoft Kinect (left) and Leap Motion (Right).

The reminder of this paper is organized as follows. Section II describe the architecture design of our proposed framework. Section III shows how to create the motion sensing commands for robot manipulation. Section IV depicts the control core for robot manipulation. Section V depicts realization of hardware interface for hardware abstraction. Section VI demonstrates implementation and experiments of the proposed framework. And section VII summarizes our study and works out the future work.

## II. FRAMEWORK ARCHITECTURE DESIGN

For motion sensing based robot manipulation, the proposed framework should be characteristic of: (1) Skeleton tracking and gesture capture, especially the poses and joint angles for the chosen sensing area. (2) Conversion from human movements to robot actions. (3) Compatibility with varied hardware, including motion sensing devices and robot. (4) Controller for accurate, robust and safe manipulation.

To meet these requirements, the design philosophy of our proposed framework is to develop as distributed and modular as possible, so that it could be hardware independent and target customized. The foundation of our framework is laid on the powerful Robot Operation System (ROS) [13], which adds value to the development of robotics projects and applications with packaged libraries and tools.

We have designed the framework in a three-layer structure, which comprises of the motion sensing, ROS foundation and hardware interface, as given in Fig. 3. The three-layer structured framework is targeted to provide an effective approach for general robot manipulation.

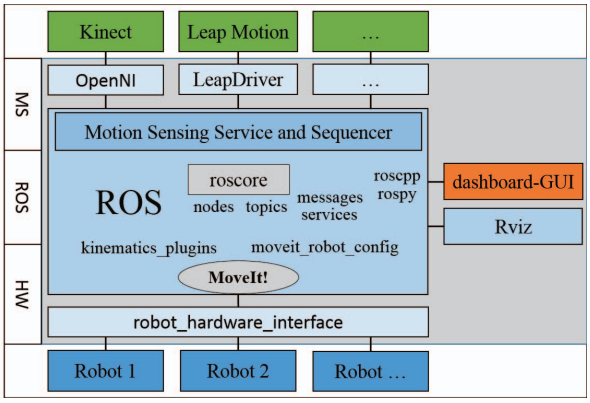


Fig. 3. Design architecture of the proposed framework for motion sensing based robot manipulation with the three main designed layers, the MS-motion sensing layer, ROS layer and the HW-hardware layer.

The first Motion Sensing (MS) layer provides drivers for supported devices, and manages the inputted commands from skeleton tracking and gesture capture. Note that this layer is heavily based on ROS communities for open device-drivers, while our main contribution is the distributed and modular designed Motion Sensing Service and Sequencer, which will allows the end-users to request poses or joint angles commands from the motion sensing devices friendly. Existing services can be modified, or new services can be added very easily in ROS layer to adapt for customized manipulation task.

The ROS foundation layer is at the heart of the proposed framework, called the Framework Core. On functional level, the layer can be divided into four modules, which are the information management module, model management module, manipulation control module and visualization module. Developed on ROS platform, this layer creates the middleware between motion sensing devices and the real robots.

The hardware interface (HW) layer is a robot-targeted communication interface that talks and listens to the controller of physical robots. With communication protocols, it can build a service-request mode between the framework and the actual robots.

## III. MOTION SENSING COMMANDS

In this section, we will focus on how to understand the intent of the operator using motion sensing input devices and create motion sensing commands for robot manipulating.

### A. 3D Tracking of Hand Articulations

3D tracking of articulated objects is an interesting and hot research problem. And the latest revolutionary motion sensing devices, with the represent of Kinect and Leap Motion, are boosting wider applications of this technology from gaming to robotics. Among them, 3D tracking of human hands is the most efficient and straight-forward interaction mode. Using these motion sensing input devices, we can easily acquire the 3D position, orientation and full articulation of a human hand. In ROS communities, the MIT Kinect demos [14] provided with purely hand and finger detection, and the

leap motion SDK [15] also established a native supported for hands skeleton tracking, as shown in Fig. 4.

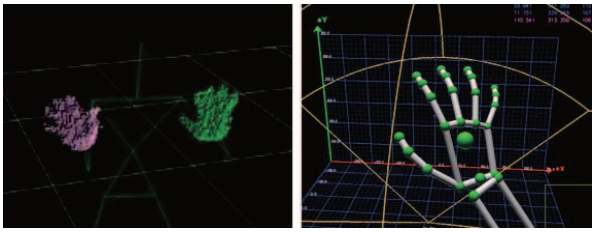

Fig. 4. 3D tracking of human hands with position, orientation and full articulation. (a) Two hands tracking from Kinect, using point cloud to display in Rviz, (b) hand skeleton tracking from Leap Motion, using native visualization tool in LeapCommandCenter.

With the open device-drivers from ROS communities, we modified the original drivers and integrated into a multi-sensor driver package for most wildly used motion sensing input devices. The ROS package does not have all the functions that the devices are official capable, but it is enough for robot manipulating, and also can be easily extended. The package currently is capable of tracking the hand palm, given in Fig.5, and provides the following information: hands numbers, hand types, each hand palm position, and each hand palm direction. The information is defined as *motion_sensor/hand.msg* in the custom message, and can be published into ROSMASTER in Node */motion_sensor/hand*.

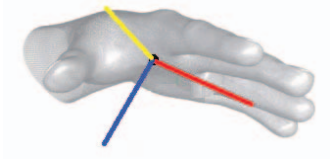

Fig. 5. Hand palm tracking of the current motion sensing driver package.

| **User-defined Message 1** | |
|---|---|
| **motion_sensor/hand.msg** | |
| std_msgs/Header | header |
| uint32 | hands_num |
| string[] | hand_typer |
| geometry_msgs/Point | position |
| geometry_msgs/Quaternion | orientation |

### B. Motion Sensor Data Processing

Even though numerous palm pose can be captured and acquired from motion sensing devices, there are still some theoretically interesting problems, the space registration, and sensor data filtering.

To satisfy the real-time and accurate tracking, the hand space {G}, motion sensing device space {O} and the robot space {R} need to be mapped to realize registration. Since we have no additional optical measuring device to build this transformation relationship during operation, we make full use of dynamic time warping (DTW) to calculate every pose distance between two hand gestures to measure the minimal motion similarity [16] between end effector of the robot and the hand. Thus, we avoid building the transformation relationship between coordinate system, but try to map the dynamic distance matrix of hand palm in motion sensing device space {O} to distance matrix of end effector in world space {W}, as shown in Fig. 6. Source motion created by human hand will be imitated by the robot through motion sensing devices.

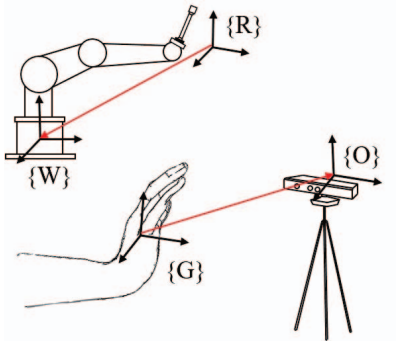


Fig. 6. The coordinate systems in motion sensing based manipulating operation.

Motion sensing devices are so acute that manipulation process may easily fail due to sensor noise or natural hand shaking noise. In Fig. 7, the plotted out hand palm position in z from sensing data without filer shows a slight fluctuation, which will definitely lead robot to tremble in corresponding axial direction. Additionally, uncontrolled hand movement, especially when hand moves into and departs from the sensing scene, may also cause abnormal action of the robot. When designing the filter, we consider the fast response performance. The simplest way is to do a moving average of the sensor data. The red line in Fig.7 is the same sensor data with average of array as large as 100. And for uncontrolled hand movement, we set an optimal threshold to ignore huge pose changes in a short time.

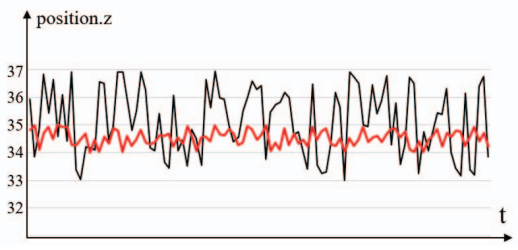


Fig. 7. Hand plam position's Z location plotted in the graph for a certain time when holding hand in one fixed pose. Where the black line is the sensor data without sensing data filer, and red line is the sensor data with filter.

### C. Motion Sensing Commands Publish

In order to publish the final motion sensing commands which is as specific as the 6D-pose of the hand, we use the *geometry_msgs/Pose* message. The Pose consists of one *Point* with three float64 (x, y, z) coordinates and one *Quaternion*

with four float64 (x, y, z, w) values. The rqt_graph tool provides by ROS can create the graph of the node relations as shown in Fig. 8.

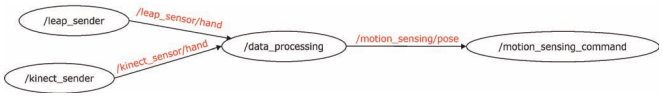


Fig. 8. ROS Nodes and Topics created by rqt_graph for motion sensing layer. Where Node /data_processing subscribed /leap_sensor/hand or /kinect_sensor/hand and published a Topic /motion_sensor/pose for Node /motion_sensing_command.

## IV. FRAMEWORK CORE

As the framework core, the ROS foundation layer consists of four modules to play as the middleware combining software and hardware together. On functional level, those modules are responsible for information management, model management, manipulating control and visualization

### A. Information Management

In a typical ROS system, the roscore process acts as the central manager of all nodes and establish point-to-point communication between nodes in the topics or services paradigms. A ROS topic can publish messages and subscribe to certain topics to build a unidirectional communication channel between one or many publishers and an arbitrary number of subscribers. While ROS service implements the request-response paradigm for communication between clients and a single server. The messages to be exchanged can be standard ROS defined or user-defined. In this framework, information includes two kinds, the sensor information and the geometry information. For sensor information, currently we have:

*/motion_sensing_command* as *geometry_msgs/Pose*
*/joint_states* as *sensor_msgs/JointState*

All geometry information in ROS are managed by *tf* transformation library, including every robot parts, markers (marker array), sensors and hands.

### B. Model Management

The MoveIt [17] is used in our framework to replace the original *robot_description* stack. Using the MoveIt Setup Assistant, we can easily build configuration files for any given robot.

In this framework, we integrated one type of industrial robot, the Staubli TX90, with MoveIt. Meanwhile, for better kinematics performance, we use the Kinematics/IKFast [18] to build a kinematics plugin. Therefore, the model management will consist of as two packages, the *moveit_robot_config* and *kinematics_plugins*.

### C. Manipulation Control

To drive robot with motion sensing commands, the manipulation control module need to request cartesian motion and start the corresponding arm motion. In this framework, we build an object oriented task manager with roscpp, which can get the motion sensing commands from Motion Sensing Service and Sequencer in ROS service paradigm and send setPose() command to robot controller, but also requires joint_values and pose_values returns for collision-aware planning. And the task manager provides the following API for motion sensing control:

*bool motion_setPose (geometry_msgs::Pose & pose);*
*geometry_msgs::Pose info_getPose ();*
*std::vector⟨double⟩ info_get_joints ();*

### D. Visualization

The ROS rviz visualization tool provides a 3D viewer for robot manipulating scene and information display. Additional, with Gazebo simulator [19], we can simulate robot in a real physical world with realistic sensor feedback and physically plausible interactions between objects. This simulation parts have been established in our previous work [20]. For friendly use, the ROS node */dashboard_gui* established a simple dashboard-style dashboard user interface for operation and settings.

Therefore, the whole framework core realized the process from subscribing the */motion_sensing_command*, planning and generating executable motion for targeted robot configuration, and finally publishing the desired */motion_command* to the robot controller.

## V. GENERAL HARDWARE INTERFACE

The hardware interface layer is acting as translator between hardware devices and ROS foundation. This layer is a combination of low level communication protocols and also provides software interfaces to ROS, enabling applications to access and operate hardware devices. And in our framework, this layer can communicate with controllers through Ethernet.

The robot driver communicates with ROS through a general robot message type motion_command/command.msg in Table. 2. Trajectories, from /motion_command, are streamed to the controller using this message type through supported communication protocol. After that, robot controller buffers these points and interpolates between them to drive motors of the robot to desired pose.

In practice, the communication protocol is relayed to the supported communication types of the targeted robotic controller. Generally, the Ethernet and CAN bus are wildly used. Currently, in our framework, we realized the driver with SOAP protocol, and provides the following API for motion command:

*bool* ***GetRobots****(std::vector⟨int⟩ robots);*
*bool* ***Login****(const std::string& url, const std::string userName, const std::string& password);*
*bool* ***GetRobotJoints****(std::vector⟨double⟩ joints);*
*bool* ***GetRobotCartesianPosition****(std::vector⟨double⟩ position);*
*bool* ***SetJoints****(const std::vector⟨double⟩ joints);*
*bool* ***MoveL****(std::vector⟨double⟩ pos);*
*bool* ***MoveJ****(std::vector⟨double⟩ pos);*

These classes are particularly useful on command robot controllers to drive the robot to the targeted pose as desired as the motion sensing commands.

| User-defined Message 2 | |
|---|---|
| **motion_command/command.msg** | |
| uint8 | command_index |
| std_msgs/Header | header |
| float32[] | position |
| float32[] | orientation |
| float32[] | joints |
| - - - | |
| uint8 | result |
| std_msgs/Header | header |
| float32[] | position |
| float32[] | orientation |
| float32[] | joints |

After obtaining the connections between ROS with the targeted robot controller, we can easily build two nodes, motion_controller_server and motion_controller_client with defined messages types to establish the request-response paradigm for motion commands publishing.

## VI. IMPLEMENTATION AND EXPERIMENTS

### A. Experiment Configuration

To preliminarily validate the proposed framework, we conduct the motion sensing robot manipulating experiments both in simulation and physical robot. The motion sensing input devices are Kinect Xbox 360 and leap motion the SDK v2. The targeted robot is the 6 DOF Staubli TX90 with SOAP communocation supported. We can easily generate the *staubli_moveit_config* and *staubli_tx90_ikfast* packages. After that, we can modify the launch file and start to check the scene in Rviz.

### B. Experiment Result and Analysis

The result of the motion sensing based robot manipulation experiment in simulation is shown in Fig.9. Both Kinect and Leap Motion are tested under the same environment.

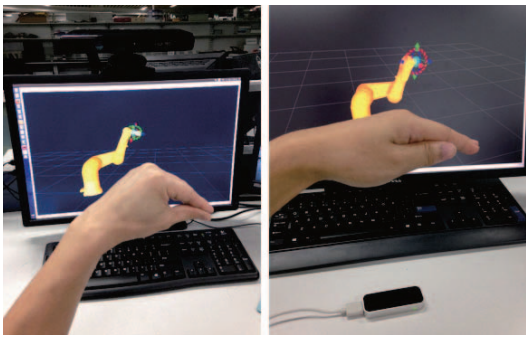

Fig. 9. Motion sensing robot manipulating using the proposed framework. With Microsoft Kinect (left) and with Leap Motion (Right).

And a very simple pick & place manipulation experiment was also conducted on the physical robot with Kinect, as setup in Fig. 10. And Fig. 11 gives the snapshots of robot manipulation from initial pose to picking pose and final picking position.

The results from simulation and physical experiments have shown that the proposed framework is an effective approach for robot manipulation with motion sensing control.

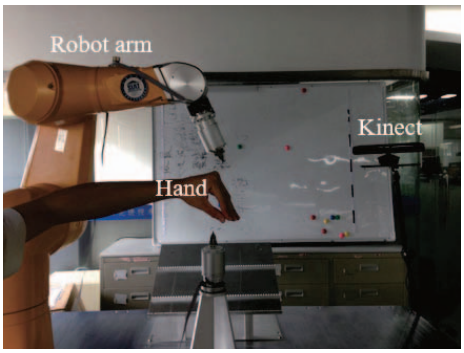


Fig. 10. Motion sensing robot manipulating on the physical robot with Kinect and in which, the robot is moving from initial place to the first picking pose as the hand commands.

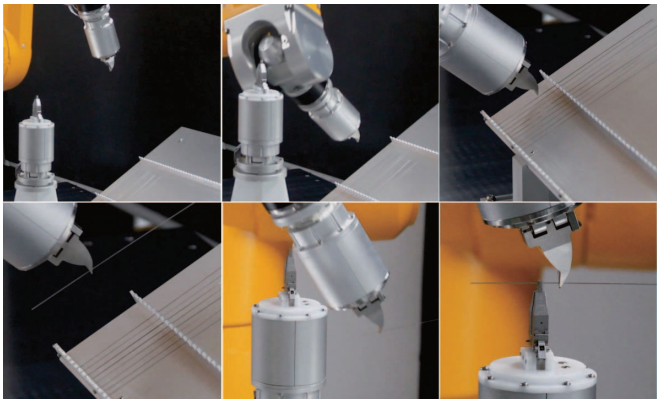

Fig. 11. Snapshots of motion sensing robot manipulating, moves from initial place to the first picking pose, picks the object and places at the desire position.

## VII. CONCLUSION AND FUTURE WORKS

This paper proposed a motion sensing based robot manipulation framework. The framework contains an integrated motion sensing devices driver for gesture commands recognition, a ROS-based framework core to transform motion sensing commands to robot motion commands, and a general hardware interface for compatibility with varies robot manipulators. The validation in simulation and physical robot have shown that the proposed framework is an effective approach for general robot manipulation with motion sensing control.

Our future works include: (1) application on more wide brands of robots with complex manipulating task, (2) two hands motion sensing manipulation with dual arms, (3) force sensor applying to avoid collision in very near area, and (4) accuracy and robustly improvement.

## REFERENCES

[1] C. C. Kemp, A. Edsinger and E. T. Jara, "Challenges for Robot Manipulation in Human Environment" in *Proc. 2007 IEEE Int. Conf. on, Robotics and Automation*, vol. 14, no. 1, pp. 20-29, March 2007.

[2] C. Kemp, L. Aryananda, A. Edsinger, P. Fitzpatrick, L. Natale, and E. T. Jara, eds. in *Proc. RSS Workshop: Manipulation for Human Environments*, http://www.archive.org/details/, manipulation_for_human_environments_2006, Philadelphia, PA, Aug. 2006.
[3] N. Sian, T. Sakaguchi, K. Yokoi, Y. Kawai, and K. Maruyama, *Operating humanoid robots in human environments*. in *Proc. RSS Workshop: Manipulation for Human Environments*, Philadelphia, PA, Aug. 2006.
[4] Online, "Official RobotWorks Homepage". Available at: http://www.bluetechnik.com/robotworks.html, 2015.
[5] Todd, Robert H., Dell K. Allen, Leo Alting. "Manufacturing Processes Reference Guide ". Industrial Press Inc, New York City, 1994.
[6] K. H. Low. "Industrial Robotics: Programming, Simulation and Applications ". pIV pro literature Verlag Robert Mayer-Scholz, 2007.
[7] Online, "Kinect for Windows ". Available at: https://www.microsoft.com/en-us/kinectforwindows/, 2015.
[8] Online, "Leap Motion ". Available at: https://www.leapmotion.com/, 2015.
[9] K. L. Ou, W. H. Tarng, Y. C. Yao and G. D. Chen. "The Influence of a Motion-sensing and Game-based Mobile Learning System on Learning Achievement and Learning Retention ". *in Conf. 2000 IEEE Int. Conf. Advanced Learning Technologies*, pp. 511-515, 2011.
[10] T. Lu, C. To, C. Lin and Y. Lin, "Applying Motion Sensing Technology to Interact with 3D Virtual Characters " in *Proc. 2011 Science & Research Conferencee*, Aug. 3-5, 2011, Singapore.
[11] I. J. Ding, C. W. Chang and C. J. He, "A Kinect-Based Gesture Command Control Method for Human Action Imitations of Humanoid Robots" *in Conf. 2014 IEEE Int. Conf. Fuzzy Theory and Its Applications*, pp. 108-211, 2014.
[12] [12] R. A. El, J. Huang and M. T. Yeh, "Study on the Use of Microsoft Kinect for Robotics Applications" *in Conf. 2012 IEEE Int. Conf. Position Location and Navigation Symposium*, pp.1280-1288, 2012.
[13] Online, A. Ioan, and L. Sucan, "ROS ". Available at: http://www.ros.org, 2015.
[14] Online, Docs.ros.org, "MIT Kinect Demos ". Available at: http://wiki.ros.org/mit-ros-pkg/KinectDemos, 2015.
[15] Online, Docs.ros.org, "leap_motion ". Available at: http://wiki.ros.org/leap_motion, 2015.
[16] A. Senin, "Dynamic Time Warping Algorithm Review" Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA, 2008.
[17] Online, Docs.ros.org, "MoveIt! ". Available at: http://moveit.ros.org/, 2015.
[18] Online, Docs.ros.org, "Kinematics/IKFast-MoveIt ". Available at: http://moveit.ros.org/wiki/Kinematics/IKFast, 2015.
[19] W. Q, Z. Y. Xia, J. Xiong, Y. Z. Gan, Y. C. Guo, S. K. Weng. H. Deng, Y. Hu and J. W. Zhang, "Manipulation Task Simulation using ROS and Gazebo" *in Conf. 2014 IEEE Int. Conf. Robotics and Biomimetics*, 2014.