Modeling and Simulation of 6 DOF Robotic Arm Based on Gazebo

Zexin Huang
Institute of Robotics and Automatic Information Systems,
Nankai University
Tianjin, 300350, People's Republic of China
email: hzexin@mail.nankai.edu.cn

Fenglei Li
Institute of Robotics and Automatic Information Systems,
Nankai University
Tianjin, 300350, People's Republic of China
email: lifenglei_nk@163.com

Lin Xu*
Institute of Robotics and Automatic Information Systems, Nankai University
Tianjin, 300350, People's Republic of China
email: xul@nankai.edu.cn, telephone: 23505706-816

***Abstract*—This paper proposes a method for robotic arms in realistic physical environment based on Gazebo. Parallel-Axis Theorem is used to estimate the links' moment of inertia. Controllers of the joints and the path planning algorithm are the important parts of the system. ROS is chosen to organize task architecture considering its easier hardware abstraction. The simulation system is able to analysis and compare robotic arm's responses under different algorithms and different target pose. When several targets given at the same time, the system can analysis which is optimal concerning the path's length and velocity's change smoothness.**

Keywords-robotic arm; simulation; gazebo; joint position controller

I. INTRODUCTION

Robotic arm is applied in many different fields such as submarine task, space manipulator, industrial automation and medical applications. Robot programmers should visualize and test the behavior of the robotic arm in different circumstances and with different parameters [1]. Simulation of robotic arm is helpful for verification of the performance of planning algorithms and researches on dynamic characteristics and limit state. It has great advantages in cost, time and safety.

Recent related studies have been carried out in these areas. Lidong M et al. proposed a multidimensional space microgravity ground simulation method and designed a Microgravity Simulation System [2]. With the help of computer simulation, Zhijun Zhang et al. verified their neural networks and numerical methods effective to solve the repetitive motion planning scheme of redundant robot manipulators [3]. Simiao Yu et al. proposed a force and displacement compensation method towards the hardware-in-the-loop simulation system for manipulator docking [4].

As for the visualization tools of simulation, RVIZ, Adams, MATLAB and Gazebo are the most commonly used simulation platforms.

RVIZ is a commonly used simulation platform on Robot Operating System (ROS). Cunfeng L et al. proposed a method to realize the simulation system of service robot based on ROS [5]. Using SolidWorks, RVIZ and MoveIt, they designed a process to realize the simulation of service robot. Richard Tatum et al. made observations on the 7 degrees of freedom (DOF) Schunk arm to solve the inverse kinematics. They showed how to solve the high DOF system to obtain closed form solutions and developed a simulation system using RVIZ to visualize the moving of the arm [6].

MATLAB is a powerful and easy-to-use tool. With the help of support packages, MATLAB can take control of the manipulator actuators by connecting with raspberry pi board or Arduino board. J. Senthil Kumar et al. simulated with MATLAB and hardware in loop to prove PID controller satisfied joints' position tracking of the two-link robotic manipulator [7]. J.A. Velarde-Sanchez et al. proposed a method to plot manipulator path in 3D space using Simulink and MATLAB [8].

Providing better assessment of how loads and forces vary throughout a full range of motion and operating environment, Adams improves the accuracy of Finite Element Analysis. Yang C et al. simulated on a platform model and proved the novel controller superior to traditional PID controller. The platform mechanism was established in Adams and the controller was built in Simulink [9].

Among the tools mentioned above, Adams has physical engine but it is relatively difficult to connect Adams with the actual robot directly. MATLAB and RVIZ can be connected with hardware easily but has no physical engine to simulate collision or other motion. Gazebo has close relationship with ROS communication mechanism and hardware. Simulation can be easily transplanted to the actual manipulator. Gazebo is rather a simulation environment with physical engine to obtain data in the required environment [10]. Gazebo has advantages in robotic arm simulation. However, many research on simulation with Gazebo mainly uses official models like PR2 and UR5 [11, 12]. Most research focuses on building the simulation environment like underwater environment [13]. The application object of the existing method is limited. To solve this problem, this paper presents a general simulation process to establish control of 6 DOF robotic arm in Gazebo.

The paper is structured as follows: Section II describes the architecture of the simulation system, Section III describes the

978-1-7281-6139-6/20/$31.00 ©2020 IEEE

system design from three aspects. Experiment results are discussed in Section IV and conclusions and future work are discussed in Section V.

II. System Architecture

This robot simulation system consists of three parts: model attributes and configuration, data processing and path planner, graphical user interface (GUI) and simulation environment (Fig. 1).

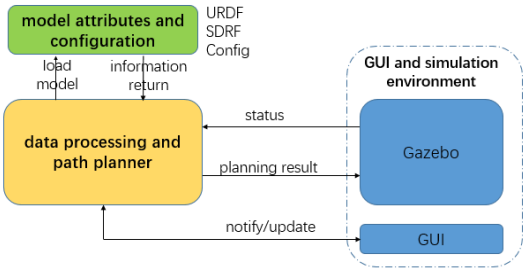


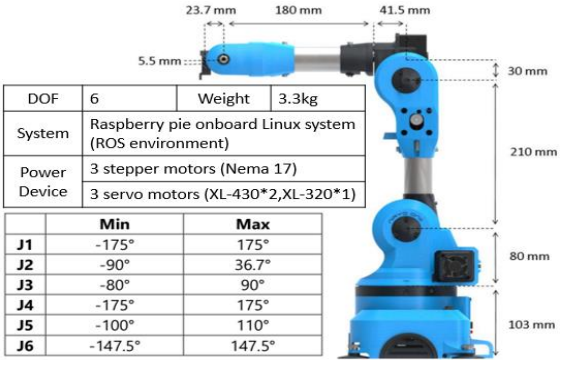
Figure 1. Simulation architecture.

Model attributes and configuration are described in Unified Robot Description Format (URDF), Semantic Robot Description Format (SRDF) and Config format. These files are mainly about physical attributes and assembly method. The second part receives parameters or commands, loads model files and gets the planning result. The last part is the input and output part of the system. Without this part, one can send commands using command terminal and analysis the planning result with data like acceleration and speed, but relatively abstract and inconvenient.

The system architecture is similar to Model-View-Presenter pattern [14]. In this pattern, the display part is separated from the models. Modifying input or output interface format, we can easily design and change a GUI or display planning result in other simulation environment.

III. System Design

A 6 DOF robotic arm, Niryo arm, is used (Fig. 2). It is controlled by raspberry pie onboard Linux system with ROS environment. In this section, the design of the system is presented.



DOF	6	Weight	3.3kg
System	Raspberry pie onboard Linux system (ROS environment)		
Power Device	3 stepper motors (Nema 17)		
	3 servo motors (XL-430*2,XL-320*1)		

	Min	Max
J1	-175°	175°
J2	-90°	36.7°
J3	-80°	90°
J4	-175°	175°
J5	-100°	110°
J6	-147.5°	147.5°

Figure 2. Geometric parameters of Niryo arm.

A. Model Attributes and Configuration

The model file consists of three parts: link, joint and transmission mechanism. Link attribute describes robotic arm's links from three aspects: visual, collision and inertial. More specifically, visual aspect describes geometric position, surface model, materials and other features. Collision aspect mainly describes collision entity feature. Inertial aspect describes inertial reference coordinate system, mass, moment of inertia (MOI) and other features. Joint attribute mainly describes range of rotation and speed limit. Transmission mechanism attribute is mainly about transmission mechanism and gear ratio.

Among these attributes, MOI plays an important role. Only when it is relatively accurate can the robotic arm be simulated correctly in the physical engine.

MOI of an object is defined as:

$$I = \sum_i m_i r_i^2 \tag{1}$$

A compound object's MOI is the sum of each part of the object:

$$I_{\text{total}} = \sum_i I_i \tag{2}$$

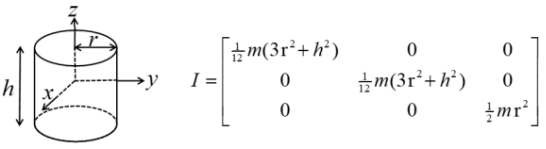


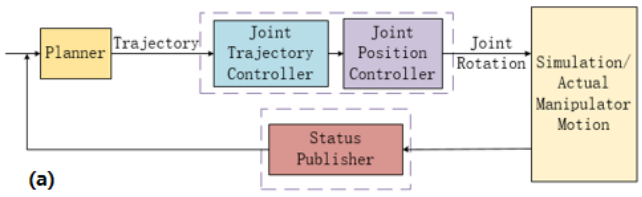
Figure 3. Moment of inertia of a cylinder.

MOI of simple body can be calculated using formula (Fig. 3). Each link of robotic arm is complex. In order to estimate their MOI, we simplify each part into several bodies. Using Parallel-Axis Theorem,

$$I_{\text{parallel-axis}} = I_{\text{center of mass}} + md^2 \tag{3}$$

in which represents the mass of an object and the distance from an axis through the object's center to a new axis, we can calculate MOI with the axis of the robot. For example, the base of robotic arm can be simplified into its shell and a stepper motor. The shell is regarded as a thin-walled square column and the stepper motor a cuboid. As a result, these two parts' MOI can be easily calculated separately.

In order to control the robot model, another important configuration is joints' controllers. Real robot uses a gravity compensation for better dynamic response and reduced power [15]. Correspondingly in simulation system, gravity of links is turned off in advance but influence of MOI retained.



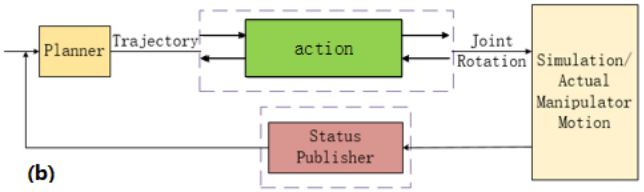


Figure 4. Joint controller's communication mechanism.

Niryo robot uses 3 stepper motors and 3 servo motors (Fig. 2). These motors have PID controllers inside them. For example, servo motor XL430 has velocity control mode and position control mode. Velocity mode uses PI control while position control mode uses feedforward and PID control.

In our simulation system, each joint has one PID controller whose parameters can be tuned. Joint trajectory controller gets and splits robotic arm's trajectory into each joint's trajectory. Joint position controller of each joint converts these data into joint rotation angles and sends them to the model in Gazebo or the real robot (Fig. 4(a)).

B. Data Processing and Path Planner

When receiving commands from the input part, data processing part performs corresponding operations. Take for examples, when receiving a starting up command, data processing part will run the scripts, load the model and wait for target pose to arrive (Fig. 5).

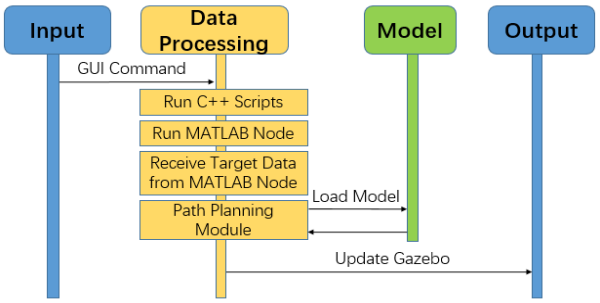


Figure 5. Process of simulation system.

After receiving target pose and other parameters, the planner plans a path according to the robotic arm model and obstacles in the environment. The path is described as a series of points' position, velocity and acceleration. Finally the points of path will be executed by position controller of each joint.

Through an action interface, a trajectory planning node called "move_group" is connected bidirectionally with the Gazebo (Fig. 4(b)). Unlike fire-and-forget mechanism, action client (/move_group) sends goal to action server (/gazebo) and gets result, feedback and status back (Fig. 6). Client can also cancel the goal.

In our system, path planner's algorithm can be customized. A custom algorithm is defined with header and source files, configured with YAML files. After building, the algorithm can be added to the method library. The algorithm now used is improved Rapidly-exploring Random Tree Star (RRT*) [16].

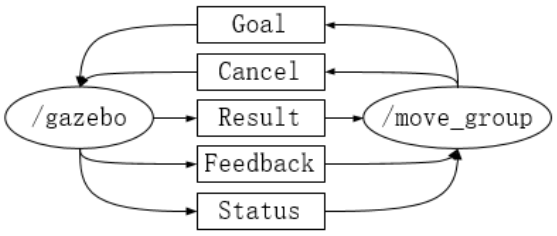


Figure 6. Mechanism of action interface.

Table I shows the process of RRT*. After generating a random point x_{rand}, RRT* finds the nearest point x_{nearest} on the tree and regards it as temporary parent of x_{rand}. Next, RRT* finds potential parent points X in a circular area with a radius of x_{rand}. Then RRT* will find the final parent point through which x_{rand} can attach the initial point in a collision free and shortest length path. Finally, the parent will be checked out and updated using the same process. The improved algorithm increases target-oriented feature on RRT* and considers about included angle between two adjacent points of the path to avoid sudden turns.

TABLE I. PSEUDO CODE OF RRT* ALGORITHM.

T.init(x_{start})
for $i = 1$ to n do
  x_{rand} = random_state(x)
  x_{nearest} = nearest_neighbor(T, x_{rand})
  x_{new} = extend(x_{nearest}, x_{rand})
  if not collision(x_{nearest}, x_{new}) then
    T.add_vertex(x_{new})
    T.add_edge(x_{nearest}, x_{new})
    X = near_neighbors(T, x_{new}, r)
    for x_{near} in X : rewire_RRT*(x_{near}, x_{new})
    for x_{near} in X : rewire_RRT*(x_{new}, x_{near})

C. GUI and Gazebo

A GUI is used to simplify system's operation. It mainly consists several parts (Fig. 7): selection of algorithm, commands to call for setting up environment and starting simulation process. When button pressed, commands or parameters would be sent to the data processing part as described in last section.

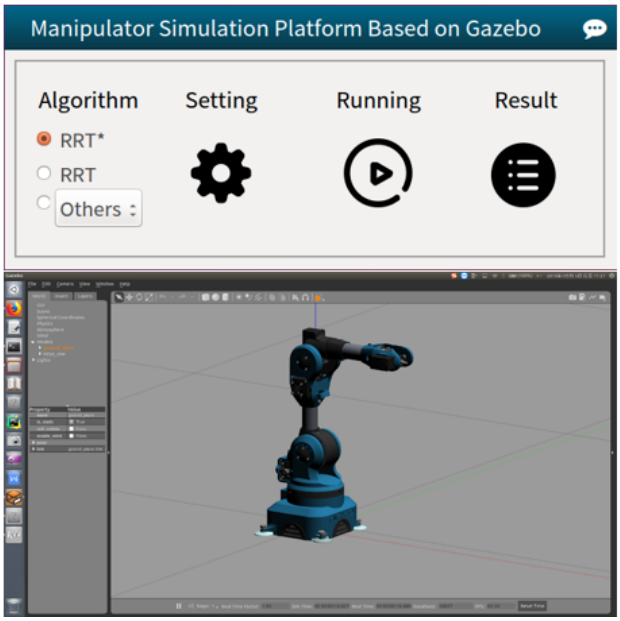


Figure 7. Input and output of the simulation system.

Gazebo (Fig. 7) supports custom plugins and provides sensor data and noise simulation. In this system, Gazebo is used mainly because of its physical engine. Input MOI together with other physical property and analysis robotic arm's motion characteristics in different motion plans.

IV. Experiment Results

As mentioned in Section III, PID parameters of joint controllers are very important. In Table II, proper parameters make each joint zero steady-state error with zero overshoot. Joints' rising time and adjusting time are all within 1 second. The table proves that this simulation system works well.

TABLE II. Response Performance of Joint Controller

Joint	Rise Time(s)	Adjustment Time(s)	Peak Time(s)	Overshoot	Steady-state Error
Joint1	0.50	0.58	0.61	0	0
Joint2	0.89	1.00	1.10	0	0
Joint3	0.71	0.78	0.80	0	0
Joint4	0.18	0.18	0.20	0	0
Joint5	0.32	0.38	0.40	0	0

Different parameters would lead to quite different response performance (Fig. 8). Improper parameters cause model shaking and poor response performance. Fig. 8(b) shows an example of slow response. Optimize the parameters by testing the influence of each joint's PID parameters on position control performance. Result in Fig. 8(a) shows that proper parameters can make the controllers feasible.

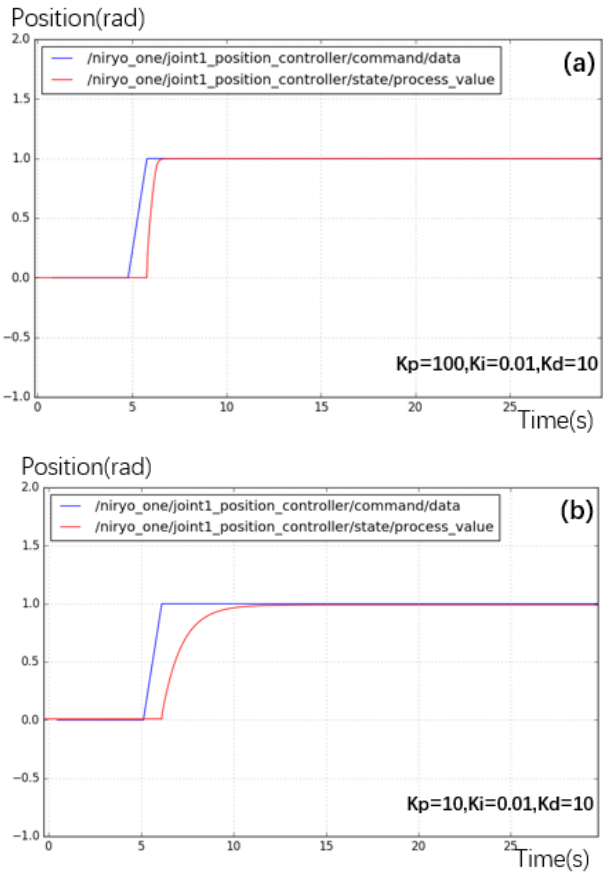


Figure 8. Different parameters leads to quite different response performance.

When different targets given, the simulation system would find corresponding feasible path consisting of several points , which represents point in the planned path of the joint . There are two basic indexes to evaluate the target points: path's length and change on acceleration. In Table III: $P_i = \sum_{j=1}^{n-1} \left| p_i^{j+1} - p_i^j \right|, A_i = D(a_i)$. Some application cases may pay more attention on path's length but others take precedence over smoothness of change on acceleration.

TABLE III. Evaluation of Two Different Target Points

Joint	Target1		Target2	
	P	*A*	*P*	*A*
Joint1	0.063	0.20	0.086	0.0016
Joint2	0.13	0.39	0.15	0.44
Joint3	0.34	0.35	0.32	0.37
Joint4	0.33	0.33	0.51	0.35
Joint5	0.43	0.29	0.70	0.34

V. Conclusions and Future Work

Many research on simulation with Gazebo mainly uses official models and the application object of the existing method is limited. This paper presents a general simulation process to establish control of six joint manipulator in Gazebo. Parallel-Axis Theorem is used to estimate each link's MOI. Joint controller with good response performance is used to establish control of joints. With the same process, the application of other robotic arm models can be realized quickly in Gazebo.

Our future study will focus on how to train a decision-making system that can decide which algorithm should be adopted when a target pose coming. Some machine learning algorithms may be used to achieve high precision and low response time.

Acknowledgment

This work is supported by the Fundamental Research Funds for the Central Universities and the National Natural Science Foundation of China (NSFC Grant No. 61873135).

References

[1] Velarde-Sanchez J A, Rodriguez-Gutierrez S A, Garcia-Valdovinos L G, Pedraza-Ortega J C. 5DOF manipulator simulation based on MATLAB-Simulink methodology[C]// International Conference on Electronics Communications & Computers. 2010.

[2] Lidong M , Yukun C , Chi G X , Zheyao X, Naiming Q. Experimental study on the multi-dimensional microgravity simulation system for manipulators[C]// 2015 International Conference on Fluid Power and Mechatronics (FPM). IEEE, 2015.

[3] Zhang Z , Zheng L , Yu J , Li Y, Yu Z. Three Recurrent Neural Networks and Three Numerical Methods for Solving Repetitive Motion Planning Scheme of Redundant Robot Manipulators[J]. IEEE/ASME Transactions on Mechatronics, 2017:1-1.

[4] Simiao Y , Junwei H , Zhiyong Q , Yu Y. A Force and Displacement Compensation Method Toward Divergence and Accuracy of Hardware-in-the-Loop Simulation System for Manipulator Docking[J]. IEEE Access, 2018, 6:35091-35104.

[5] Cunfeng L, Lina H, Hongtai C, Xin N. Research on Motion Planning System of Service Robot Based on ROS[C]// CYBER, 2017.

[6] Tatum R , Lucas D , Weaver J , Perkins J. Geometrically motivated inverse kinematics for an arm with 7 degrees of freedom[C]// Oceans. MTS, 2016.

[7] Kumar J S , Amutha E K . Control and tracking of robotic manipulator using PID controller and hardware in Loop Simulation[C]// International Conference on Communication & Network Technologies. IEEE, 2015.

[8] Velarde-Sanchez J A, Rodriguez-Gutierrez S A, Garcia-Valdovinos L G, Pedraza-Ortega J C. 5DOF manipulator simulation based on MATLAB-Simulink methodology[C]// International Conference on Electronics Communications & Computers. 2010.

[9] Yang C, He J, Jiang H, Han J. Modeling and Simulation of 6-DOF Parallel Manipulator Based on PID Control with Gravity Compensation in Simulink/ADAMS[C]// International Workshop on Modelling. IEEE, 2009.

[10] Qian W , Xia Z , Xiong J , Gan Y, Guo Y, Weng S, et al. Manipulation Task Simulation using ROS and Gazebo[C]// 2014 IEEE International Conference on Robotics and Biomimetics. IEEE, 2014.

[11] Ramirez-Alpizar I G , Harada K , Yoshida E . Motion Planning for Dual-arm Assembly of Ring-Shaped Elastic Objects[C]// IEEE-RAS International Conference on Humanoid Robots. IEEE, 2014.

[12] Rockel S , Konecny S , Stock S , Hertzberg J. Integrating physics-based Prediction with Semantic Plan Execution Monitoring[C]// IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2015.

[13] Manhaes M M M , Scherer S A , Voss M , Douat L. UUV Simulator: A Gazebo-based package for underwater intervention and multi-robot simulation[C]// Oceans. IEEE, 2016:1-8.

[14] Syromiatnikov A , Weyns D . A Journey through the Land of Model-View-Design Patterns[C]// Software Architecture. IEEE, 2014.

[15] Boisclair J , Richard P L , Laliberte T , Gosselin C. Gravity Compensation of Robotic Manipulators Using Cylindrical Halbach Arrays[J]. IEEE/ASME Transactions on Mechatronics, 2017, 22(1):457-464.

[16] Noreen I , Khan A , Ryu H , Doh N J. Optimal path planning in cluttered environment using RRT*-AB[J]. Intelligent Service Robotics, 2017