Proceedings of the 33rd Chinese Control Conference
July 28-30, 2014, Nanjing, China

Kinematics Modeling and Experimental Verification of Baxter Robot

Zhangfeng JU[1,2], Chenguang YANG[2,3]*, Hongbin MA[3,4]
1. North Automatic Control Technology Institute, Taiyuan, China
2. Centre for Robotics and Neural Systems, School of Computing and Mathematics, Plymouth University, Plymouth, UK
3. School of Automation, Beijing Institute of Technology, Beijing, China
4. State Key Lab of Intelligent Control and Decision of Complex Systems, Beijing Institute of Technology, Beijing, China

Abstract: The Baxter® humanoid robot made by Rethink Robotics™ offers users an affordable platform with guaranteed safety for both academic and industrial applications. The platform provides the users a good opportunity to carry out research on dual-arm robot manipulation and vision based control. For simulation of the Baxter® robot, a proper kinematic and dynamic model should be built. The Baxter robot uses URDF file to describe the robot kinematics, i.e., the relationship between adjacent joint and link. Consider the large difference between the structure of URDF and the conventional Denavit-Hartenburg(DH) notations which are widely used in robotics literature, we perform rigorous theoretic analysis of kinematics of Baxter® robot including all limbs. A kinematics model has been built following DH convention, and has been implemented in the MATLAB Robotics Toolbox. Extensive comparison between simulation and experimental results has verified the validity of the kinematic model.

Key Words: Baxter® robot, Simulink model, DH convention

1 Introduction

Automated robots play an important role in modern manufacturing, especially on the assembly line. The increasing pace of introduction of new products, particularly those with many variants and short lifetimes, brings considerable uncertainty to assembly line design. It is therefore very desirable to be able to develop flexible and re-configurable manufacturing systems. Sometimes, in order to reduce initial investment, manual assembly may be more preferable, but the cost exponentially increases with less automation. In this instance, the best solution is to exploit human-robot physical cooperation to close the gap between fully manual assembly and fully automated manufacturing lines [1, 2]. In this context, the humanoid dual-arm Baxter® robot with intrinsic safety (e.g., physical compliance created by spring in between driving motors and robot joints) has been developed to collaborate with human users (Fig. 1).

Baxter® robot includes a torso based on a movable pedestal and two 7DOF (degree of freedom) arms installed on left/right arm mounts respectively. Each arm has 7 rotational joints and 8 links, as well as an interchangeable gripper (such as electric gripper or vacuum cup) which can be installed at the end of the arm. A head-pan with a screen, located on the top of torso, can rotate in the horizontal plane [3].

When the robot is interacting with the external environment, the interactive force between robot and surrounding environment should be well controlled to prevent possible damaging impact which is harmful to the robot. A traditional force controller tends to increase the stiffness to obtain high bandwidth and accuracy of position [4], but this control strategy may result in poor compliance and even instability. Consequently, it is not a desired control method to implement for human friendly interaction.

* Corresponding Author. Chenguang Yang. Email: cyang@ieee.org
This work is supported in part by EPSRC grants EP/L026856/1 and EP/J004561/1, EU grant PIIFR-GA-2010-910078, Royal Society grant RG130244, NSFC-RS Joint Project grant (61211130359 and IE111175) and Beijing Outstanding Talents Programme (2012D009011000003).

Fig. 1: Human robot interaction with Baxter robot in a factory. (Captured from the video [5])

To enable compliance on the dual arms, the Baxter® robot is equipped with an SEA (Series-Elastic Actuator) instead of connecting the motor shaft directly to the joint (usually through a gear box). The motor in a SEA is coupled to the joint through a spring, so that the torque generated by twist of spring, rather than the torque from the motor directly drives the link. This enables the robot to behave in a human-like elastic manner. Due to the elastic effect of the spring, the SEA will lead to improved shock tolerance and reduced danger in cases of collision. In addition, the Baxter® robot is able to sense a collision at a very early time instant, before it hits badly onto a subject. Therefore, Baxter® robot is a good choice to fulfill safe human-robot interaction.

In order to accomplish the objective of control design based on the Baxter® robot, the first important thing is to create a model of the robot, on which simulation studies can be performed to verify any proposed control strategies. MATLAB/Simulink® software package [6] provides an efficient simulation environment, including a large numbers of mathematical functions, and a variety of specialized toolboxes. Therefore, MATLAB/Simulink® has been used to implement many kinds of real-time interface for motion control of many robots such as KUKA manipulators [7]. Robotics Toolbox developed by Peter Corke is a very useful robotic simulation software [8] built under MATLAB/Simulink® . It enables us to create and manipulate coordinate transformations and trajectories planning in a straightforward manner. There are useful built-in functions of forward/inverse kinematics and dynamics for serial-link type manipulators. Kinematics is fundamental for creating a simulation model. A robot usually consists of a series of link manipulator, which comprises of a chain of links. These links are connected with 1DOF joints [9, 10]. The pose of the end effector can be represented by a complex function of states of joints and calculated if all joints angles are known.

Denavit-Hartenberg (DH) notations are widely used to describe the kinematic model of a robot. The DH notation for describing serial-link robot mechanism geometry is a principal method for a roboticist [11]. If the DH description of a robot is known, we can easily use Robotics Toolbox mentioned above to simulate its motion and calculate its kinematics, e.g., joints configuration for a certain pose, the Jacobian matrix, and so on [12]. We have developed and tested many new control algorithms using this Robotics Toolbox in our previous work [13, 14]

However, Baxter® robot software does not provide the DH parameters directly. Instead, it uses a different coordinate system presentation, i.e., URDF (Unified Robot Description Format) file, to describe the frame transform between joints and links. To re-use many control and planning algorithms built in the Robotics Toolbox above mentioned, it is desired to obtain the DH parameters from URDF file and then build a robot model using the above mentioned toolbox.

The contribution of this paper is to build and verify kinematics model of Baxter® robot for simulation of motion control. In this paper, we also illustrate how to develop and build a robot kinematic simulation model with DH notation method by using URDF file. To test the accuracy of the model, we have designed and performed two experiments.

2 Procedure of Modelling

The study of the kinematics of the Baxter robot includes the following steps. First, we perform analysis of mechanical structure of the Baxter® robot, and the main elements in URDF file and a 3D visual model will be described. Secondly, the method to obtain the DH parameters from a link in the left arm is presented. Last, we will tabulate the DH parameters and create the kinematic model of dual arms with correspondent DH parameters. This completes the kinematic modelling of Baxter® robot.

2.1 Structural analysis

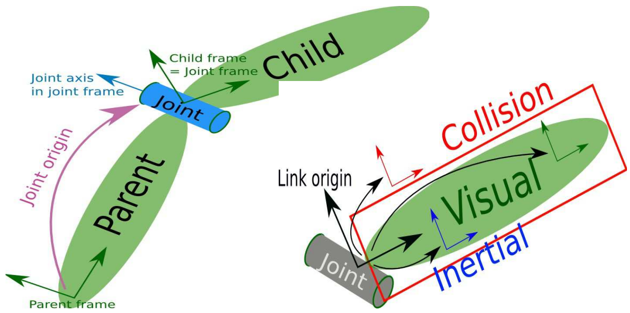
Baxter® robot has two arms and a rotational head on its torso. The arm is shown in Fig. 2. The arm of Baxter robot comprises a set of bodies, called links, in a chain and connected by revolute joints. There are seven rotational joints on the arm, named s0, s1, e0, e1, w0, w1, w2 respectively. Each joint has 1DOF.

The *"arm mount"* assembly, is fixed on the *"torso"* assembly. Joint s0 is connected to the *"arm mount"* assembly. Between joint s0 and s1, there is *"upper shoulder"* assembly. *"lower shoulder"* assembly is located between joint s1 and joint e0, and so on. This structural connection is illustrated in Fig. 2.

Fig. 2: The arms of Baxter robot, on which there are 7 joints. (Modified from [15])

2.2 URDF description

URDF (Unified Robot Description Format) is a file in XML format which describes a robot, detailing its parts, joints, dimensions, and so on. It is a fundamental robot model file in the Robot Operating System (ROS), which is a flexible framework for developing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms [16, 17]. The way ROS uses a 3D model of a robot or its parts, to simulate them or to simply help the developers in their daily work, is by the means of the URDF files. In ROS, a 3D robot such as the Baxter is always associated with a URDF file which describes the kinematic information of the robot.



(a) Joint (b) Link

Fig. 3: Joint and link in URDF file [18, 19].

The Baxter® robot URDF file consists of a series of frames and transform description between frames. A package about Baxter® robot mechanics construction can be found in [20]. This package includes all files we need to analyse key factors of Baxter® robot, such as URDF file and other necessary files.

There are two major elements in URDF file that describe the geometry of a robot: "link" and "joint", as shown in Fig. 3. These two elements are not the same as joints and links normally found in DH notation system. In fact, a "link" in URDF represents a frame. The "link" code section means a frame predefined in the robot: it describes a rigid body with an *inertia*, *visual* features, and so on. Below is an example of a link element named *left_upper_shoulder*:

```
<link name="left_upper_shoulder">
 <visual>
  <origin rpy="0,0,0" xyz="0,0,0"/></visual>
  <inertial>
  <origin rpy="0,0,0"
```

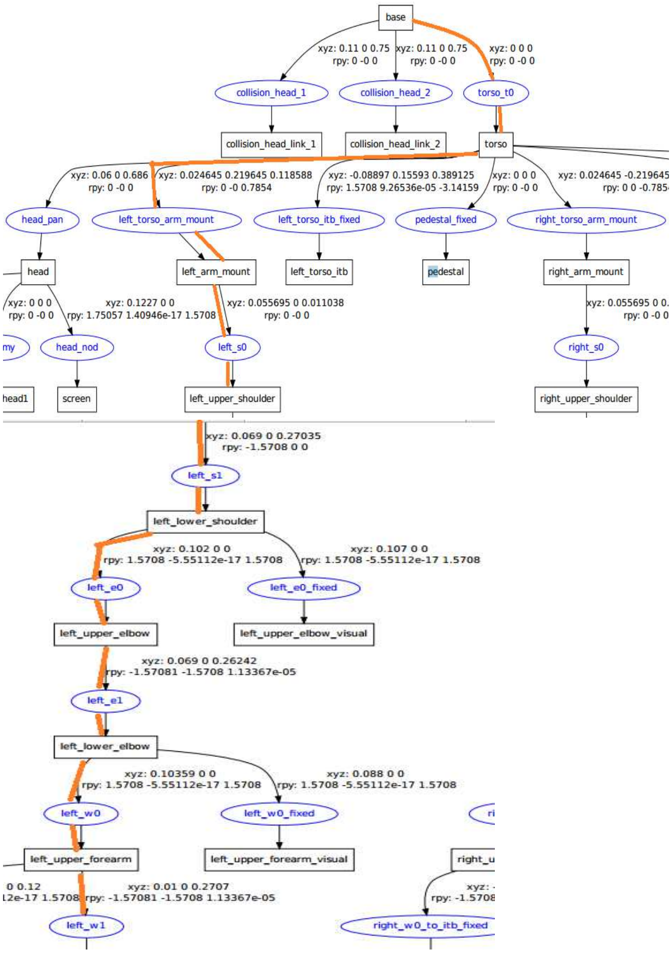


Fig. 4: Parts of frames and transform relationship between frames in Baxter® robot, generated from ROS instruction *$rosrun urdfdom urdf_to_graphiz baxter.urdf* [17]. Rectangle("link") represents a frame and ellipse("joint") represents a translation/rotation between frames.

```
    xyz="0.01783,0.00086,0.19127"/>
  <mass value="5.70044"/>
  <inertia
    ixx="0.04709102262"... izz="0.03595988478"/>
 </inertial>
</link>
```

The *visual* property of the link specifies the shape of the body (such as box, cylinder) for visualization purposes. *Origin* section in *visual* indicates the reference frame of the visual element with respect to the reference frame of the link, as shown in Fig. 3(b).

A "joint" in URDF represents a transform between two frames. It does not necessarily mean a motional (translational or rotational) joint. In fact, most of these "joints" are fixed. For example, the "joint" named *torso_s0* and *left_torso_arm_mount* are both fixed, as described above. Below is an example of a "joint" element.

```
<joint name="left_s0" type="revolute">
 <origin rpy="0,0,0" xyz="0.055695,0,0.011038"/>
 <axis xyz="0,0,1"/>
 <parent link="left_arm_mount"/>
 <child link="left_upper_shoulder"/>
 <limit lower="-1.7" upper="1.7" />
</joint>
```

The *type* property specifies the type of "joint": revolute, prismatic or fixed, where the former two types are motional joint. The *origin* property is the transform from the parent link to the child link. The "joint" is located at the origin of the child link, as shown in the Fig. 3(a) above. *rpy* represents the rotation around the axis of the parent frame: first roll around x, then pitch around y and finally yaw around z. *xyz* represents the position offset from the origin. *lower* and *upper* describe the minimum and maximum joint angles in radians. In Fig. 4, the first link has name *base* which is represented by a rectangle, also representing a coordinate frame named *"base"*. The ellipse represents a coordinate transformation between two frames, for example, through *torso_t0*, frame *base* can be translated to frame *torso*. In order to get the transform between frames, we will use the *visual* properties of link, and we can get a transform path of left arm, which is depicted in Fig. 4 in a bold line.

2.3 DH (Denavit-Hartenberg) notation

Standard DH notation is used to create a kinematics model of robot. The fundamentals of serial-link robot kinematics and the DH notation are well explained in [21, 22].

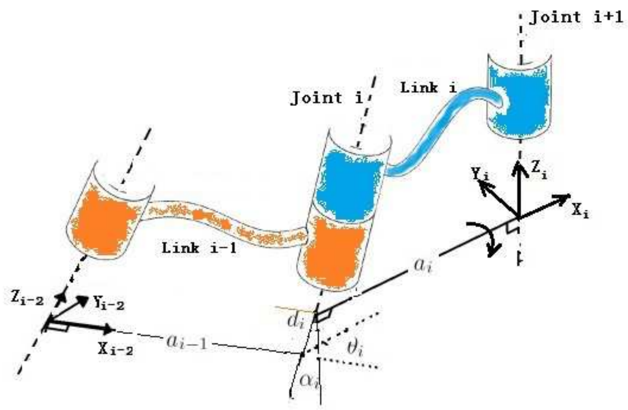


Fig. 5: Definition of standard DH link parameters. (modified from [23]).

In DH notation system, each link can be represented with two parameters: the link length a_i and the link twist α_i. The link twist α_i indicates the axis twist angle of two adjacent joints i and i-1. Joints are also described by two parameters: the link offset d_i, which indicates the distance from a link to the next link along the axis of joint i, and the joint revolute angle θ_i, which is the rotation of one link with respect to the next about the joint axis [12]. Usually, three of these four are fixed link parameters while one is a joint variable. For a revolute joint, the joint variable is considered to be θ_i, while for a prismatic joint, it will be d_i. This principle is depicted in Fig. 5.

The four parameters of each link can be specified using DH notation method. With these parameters, link homogeneous transform matrix which transforms link coordinate frame $i-1$ to frame i is given as formula (1).

$$
\begin{aligned}
{}^{i-1}A_i(\theta_i, d_i, a_i, \alpha_i) &= R_z(\theta_i)T_z(d_i)T_x(a_i)R_x(\alpha_i) \\
&= \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ c\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{1}
$$

where $\sin(\theta)$ is abbreviated as $s\theta$ and $\cos(\theta)$ as $c\theta$, R_j denotes rotation about axis j, and T_j denotes translation along axis j. For an n-link robot arm, the overall arm transform namely forward kinematics, in terms of the individual link transforms can be expressed in formula (2). The Cartesian position of the end effector can be calculated from formula (3).

$$^{0}A_n = {}^{0}A_1 {}^{1}A_2 \cdots {}^{n-1}A_n \quad (2)$$

$$^{n}X_0 = {}^{0}A_n X_n \quad (3)$$

where $X = [x, y, z, 1]^T$ is an augmented vector of Cartesian coordinate.

2.4 Kinematic model of Baxter

Now we are ready to create forward kinematics model of Baxter® robot. We will take left arm as an example, and try to create the left arm simulation model use DH notation.

From the frame of *base*, the "joint" *torso_s0* and *left_torso_arm_mount* are fixed, the first motional joint we can find is *left_s0*, so we regard the *left_s0* as the first joint and *left_upper_shoulder* as the first link, *left_s1* and *left_lower_shoulder* as second joint and link respectively, one by one until the seventh joint *left_w2*, the last rotational joint, and link *left_wrist*, are calculated.

The first four frame transformation can be found in URDF file (refer to Fig .4). Here is the description of *left_s0*, *left_s1*, *left_e0*, and *left_e1* in URDF:

```
<joint name="left_s0" type="revolute">
 <origin rpy="0 0 0" xyz="0.056 0 0.011"/>
<joint name="left_s1" type="revolute">
 <origin rpy="-1.57 0 0" xyz="0.069 0 0.27"/>
<joint name="left_e0" type="revolute">
 <origin rpy="1.57 0 1.57" xyz="0.102 0 0"/>
<joint name="left_e1" type="revolute">
 <origin rpy="-1.57 -1.57 0"xyz="0.069 0 0.262"/>
```

These frame transforms are shown in Fig. 6.

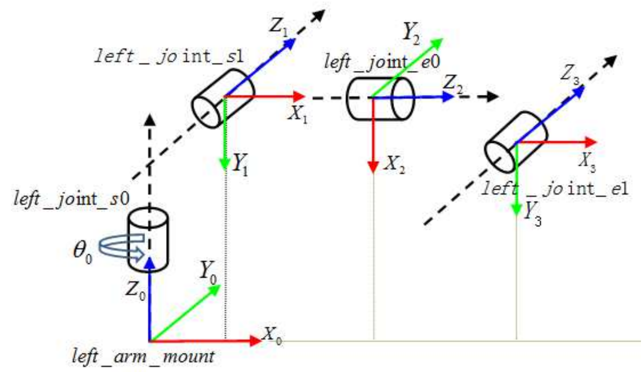


Fig. 6: First three joint frame transformational relationship.

From *left_s0* to *left_s1*, the axis of joint rotates $-\frac{\pi}{2}$ along X_0 axis (all revolution abides by right-hand law), so link twist $\alpha_1 = -\frac{\pi}{2}$ .The translation of *left_s1* in URDF file is x=0.069 m and z=0.27 m , so d_1=0.27 m and a_1=0.069 m, let θ_1=0 just because this is a revolute joint.

From *left_s1* to *left_e0*, the axis of joint rotates $\frac{\pi}{2}$ along axis Z_1 first and then rotates $\frac{\pi}{2}$ along axis X_1. So, we should let $\theta_2=\frac{\pi}{2}$ and let link twist $\alpha_2=\frac{\pi}{2}$. Although the translation of *left_e0* is $x=0.102$ m , we have to assign $d_2=0$ m and a_2=0 m because the x=0.102 m is not the offset along the axis of joint *left_s1*. This is the key in this step, to make rotation occur only at the end of a link.

From *left_e0* to *left_e1*, the axis of joint rotates $-\frac{\pi}{2}$ along X_2 axis, so link twist $\alpha_3=-\frac{\pi}{2}$. The translation of *left_e0* is $x=0.069$ m and $z=0.262$ m, so $a_3=0.069$ m. The offset along the revolution axis should be d_3=0.262 m+0.102m= 0.364 m, includes the x=0.102 m of joint *left_s1*. Repeating these steps, DH parameter table can be developed as shown in Table 1.

Table 1: DH Notation table of the left arm

Link i	θ_i(deg)	d_i(m)	a_i(m)	α_i(rad)
1	q_1	0.27	0.069	$-\frac{\pi}{2}$
2	$q_2+\frac{\pi}{2}$	0	0	$\frac{\pi}{2}$
3	q_3	0.102+0.262	0.069	$-\frac{\pi}{2}$
4	q_4	0	0	$\frac{\pi}{2}$
5	q_5	0.104+0.271	0.01	$-\frac{\pi}{2}$
6	q_6	0	0	$\frac{\pi}{2}$
7	q_7	0.28	0	0

From DH notation table, we can get the transform matrix from the frame of *left_arm_mount* to frame of *left_gripper* (end point). The model of the right arm of Baxter® robot can be created using the same procedures above. DH notation table of right arm is the same as the left arm except $d_7 = 0.275$ m , because in our configuration the gripper on right arm is an electric gripper, and a vacuum cup is installed on the left arm. According to different type of gripper mounted on each arm, the value of d_i will be different. In fact, each Baxter® robot has a unique URDF parameter configuration. Before creating the simulation model, the actual parameters should be obtained from frame transformation topic in ROS, rather than using ideal URDF file from the website. The transform from frame of *base* to *left_arm_mount* given in formula (4).

$$^{base}T_{left\_arm\_mount} = R_z(\frac{\pi}{4})T_x(0.056\,\mathrm{m})T_z(0.011\,\mathrm{m}) \quad (4)$$

Then, a complete kinematics model of Baxter® robot can be created by using MATLAB commands developed in MATLAB Robotics toolbox.

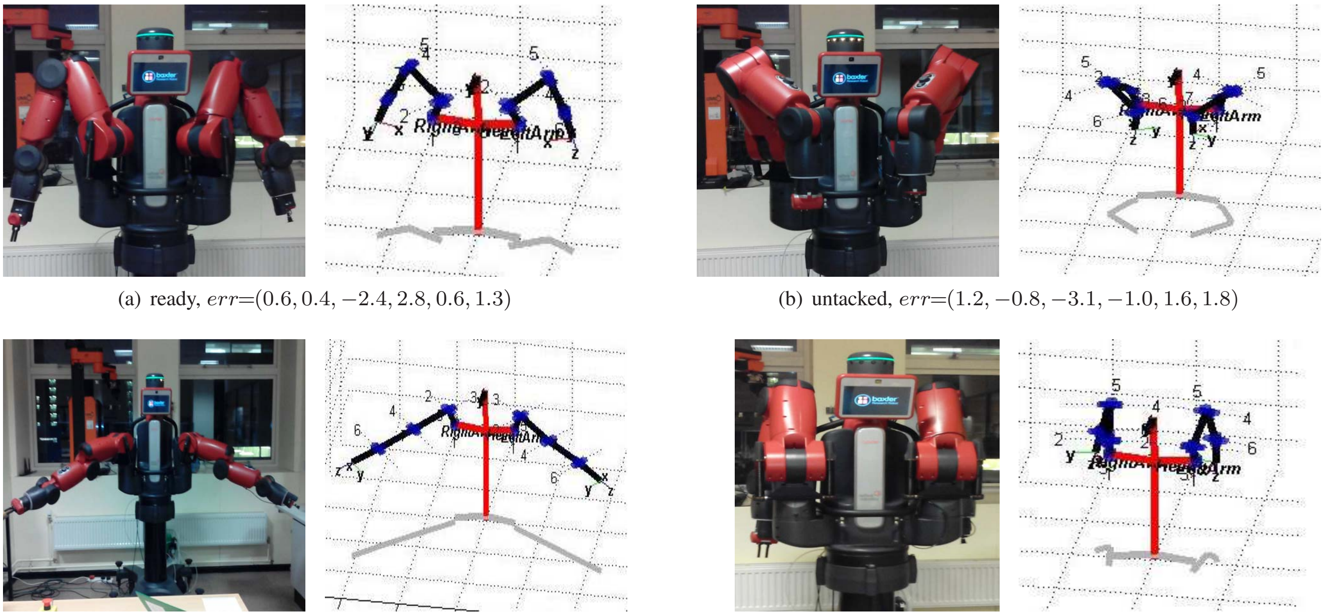
3 Experiments and Tests

3.1 Experiment 1

When the 7 joint angles $(\theta_1, \theta_2, \cdots \theta_7)$ are known, we can calculate the Cartesian position and orientation of the robot. To test the accuracy of the kinematic model we created, we put the Baxter® robot arm to several postures, retrieve the joint angular value $(\theta_1, \theta_2, \cdots \theta_7)$ and the position and orientation in Cartesian space (x_r, y_r, z_r), then input joint angles to our mathematical model and calculate Cartesian position (x_m, y_m, z_m) using forward kinematics. The postures of the robot and generated by model are shown in the Fig. 7. It can be seen that when same joint angles are given, the model has the same posture as the robot.

3.2 Experiment 2

The pose that Baxter is shipping in or in power-off state is referred to as 'shipping pose' [24] (Fig. 7(d)). Baxter's arms should be un-tucked(Fig. 7(b)) before subsequent movements. During the period when robot's arms moves from tucked pose to un-tucked pose, and when it moves back to tucked pose, we retrieved a series of joint angles and Cartesian positions (x_r, y_r, z_r) and orientations $(\eta_{Rr}, \eta_{Pr}, \eta_{Yr})$, and calculate our Cartesian position (x_m, y_m, z_m) and orientations $(\eta_{Rm}, \eta_{Pm}, \eta_{Ym})$ using formula (1)- (4) based on the model built. The deviation between them can be calcu-



(a) ready, err=$(0.6, 0.4, -2.4, 2.8, 0.6, 1.3)$ (b) untacked, err=$(1.2, -0.8, -3.1, -1.0, 1.6, 1.8)$

(c) extended, err=$(0.6, 0.7, -3.9, -2.8, 0.2, -1.8)$ (d) tacked, err=$(0.5, 0.4, -1.6, -0.9, 2.3, 0.2)$

Fig. 7: Comparison of the posture of robot and generated from model when same joint angles are given. The *'err'* is $(e_x, e_y, e_z, e_R, e_P, e_Y)$. They are calculated from formula (5) with units[1] as same as those in section 3.2.

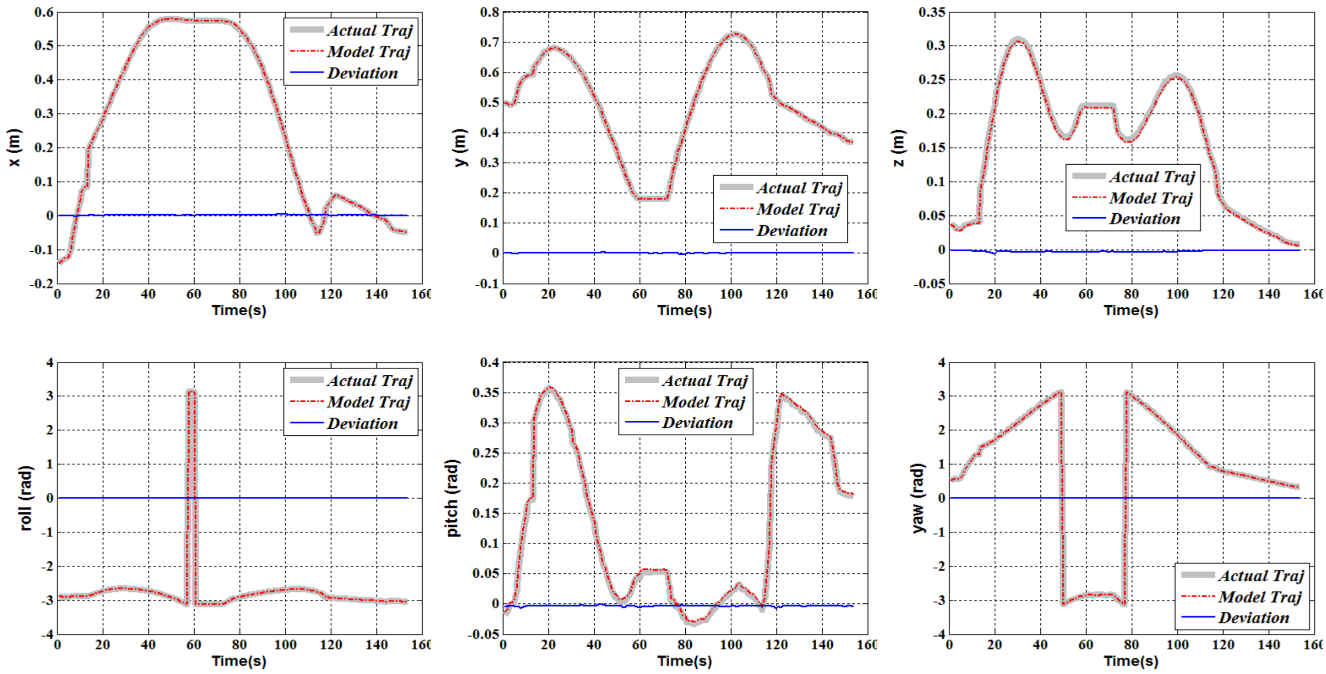


Fig. 8: Comparison of the trajectories of robot and those generated from model during the robot's left arms is tucked/un-tucked.

lated from formula (5).

$$\begin{aligned} \Delta X &= (\Delta x, \Delta y, \Delta z) = (x_m, y_m, z_m) - (x_r, y_r, z_r) \\ \Delta\eta &= (\Delta\eta_R, \Delta\eta_P, \Delta\eta_Y) = (\eta_{Rm}, \eta_{Pm}, \eta_{Ym}) - (\eta_{Rr}, \eta_{Pr}, \eta_{Yr}) \end{aligned} \tag{5}$$

Fig. 8 shows the position and orientation of left end point during the period of arm is tucking and untacking. Position and orientation of the end point of model (red curve) follows the trajectory of the actual robot (light gray curve) very closely.

The maximum error[1] of position between our model and the real robot is sufficiently small and the maximum of errors (absolute values) for right arm is listed as below

$$[e_x, e_y, e_z, e_R, e_P, e_Y]_{max} = [4.2, 3.3, 6.8, 5.6, 7.6, 7.3]$$

for left arm is

$$[e_x, e_y, e_z, e_R, e_P, e_Y]_{max} = [4.4, 3.8, 6.1, 5.1, 7.2, 7.6]$$

The average error calculated from the data for right arm is

$$[e_x, e_y, e_z, e_R, e_P, e_Y]_{avg} = [0.7, -0.6, -3.8, 2.9, -3.2, -1.2]$$

for left arm is

$$[e_x, e_y, e_z, e_R, e_P, e_Y]_{avg} = [1.1, 0.6, -2.8, -1.6, -3.5, 1.1]$$

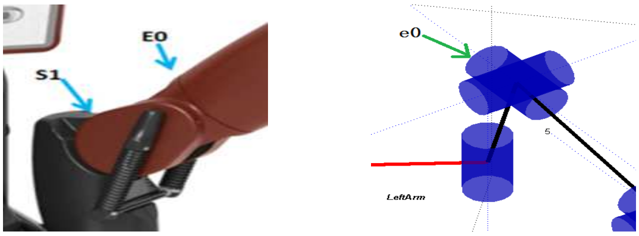
The accuracy is satisfactory for simulation purpose.

4 Summary and Discussion

In this paper, we built a kinematics model of a Baxter® robot by extracting the parameters from a URDF

[1]Unit of x/y/z is mm and unit of roll/pitch/yaw is mrad.

file and designed two experiments to verify the model. The experiments presented above verified that the kinematics model created here matches the real robot well, and can be used to perform motion simulation.



(a) Joint e0 is in the middle of upper arm in actual robot

(b) Joint e0 is at the end of the upper arm

Fig. 9: Joint rotates can only be represented at the end of link in model.

In general, the key steps to extract DH parameters from URDF file are summarised as below:

1) Fig. 4 describes the transform relationship of joints and links in the URDF file and can be considered as an inverted tree. To identify DH parameters for Baxter robot, it is necessary to find the first rotational joint ($rpy \neq 0$) from the root of this tree. Mark it as joint 1, and recognize four elements θ_1, d_1, a_1, α_1 and fill them in the DH table. Denote the next link as link 1, and assign properties includes *mass*,*inertia* and *origin* to the model. Then continue the above procedure for joint 2, joint 3, and so on.

2) When frame rotates not only about X axis (r in *rpy*), but also Z axis(y in *rpy*), an additional rotation ($\frac{\pi}{2}/-\frac{\pi}{2}$) needs to be added to the joint revolution angle θ_i.

3) The location of a joint seen on the model built in MATLAB Robotics Toolbox may be different from its location on the physical robot. For example, Baxter robot's joint e0 is located towards the middle of the the upper arm link as shown in Fig. 9(a), while on the model developed by the MATLAB Robotics Toolbox, it is located in such a manner that it shares origin with the previous joint s1, as shown in Fig. 9(b). This is because that the origin of the coordinate frame associated with joint e0 is same as the origin of coordinate frame associated with joint s1, according to DH convention. Consequently, in the MATLAB Robotics Toolbox model the link offset should be modified.

Now we see that the kinematics parameters of the robot, as well as dynamics parameters including mass, inertia and origin of gravity center are collected. Using them we could build the dynamics model as well, but further experiments and analysis need to be carried out to verify the dynamics model in our future work.

Acknowledgments

The authors thank Alex Smith for his comments and proof reading of the paper, and Peidong Liang for his support in set-up of the experiments.

References

[1] J. Krüger, T. Lien, and A. Verl, "Cooperation of human and machines in assembly lines," *CIRP Annals-Manufacturing Technology*, vol. 58, no. 2, pp. 628–646, 2009.

[2] "EU FP-7 Rosetta project." http://www.fp7rosetta.org/.

[3] "Baxter Product Datasheet." http://rr-web.s3.amazonaws.com/assets/Baxter_datasheet_5.13.pdf.

[4] G. A. Pratt, M. M. Williamson, P. Dillworth, J. Pratt, and A. Wright, "Stiffness isn't everything," in *Experimental Robotics IV*, pp. 253–262, Springer, 1997.

[5] "Baxter University: Retrain Baxter." http://www.rethinkrobotics.com/resources/videos/.

[6] "MATLAB and Simulink for technical computing." http://www.mathworks.com/.

[7] F. Chinello, S. Scheggi, F. Morbidi, and D. Prattichizzo, "KCT: a MATLAB toolbox for motion control of kuka robot manipulators," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 4603–4608, IEEE, 2010.

[8] P. Corke *et al.*, "A computer tool for simulation and analysis: the robotics toolbox for MATLAB," in *Proc. National Conf. Australian Robot Association*, pp. 319–330, 1995.

[9] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. John Wiley & Sons New York, 2006.

[10] R. P. Paul, *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*. Richard Paul, 1981.

[11] J. Denavit, "A kinematic notation for lower-pair mechanisms based on matrices," *Trans. of the ASME. Journal of Applied Mechanics*, vol. 22, pp. 215–221, 1955.

[12] P. Corke, "A simple and systematic approach to assigning denavit-hartenberg parameters," *IEEE transactions on robotics*, vol. 23, no. 3, pp. 590–594, 2007.

[13] A. Smith, C. Yang, H. Ma, P. Culverhouse, A. Cangelosi, and B. E, "Bimanual robotic manipulation with biomimetic joint/task space hybrid adaptation of force and impedance," in *IEEE International Conference on Control & Automation (ICCA),in press*, IEEE, 2014.

[14] C. Yang, Z. Li, and E. Burdet, "Human like learning algorithm for simultaneous force control and haptic identification," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 710–715, IEEE, 2013.

[15] E. Guizzo and E. Ackerman, "How rethink robotics built its new baxter robot worker," *IEEE Spectrum*, 2012.

[16] "About ROS." http://www.ros.org/about-ros/.

[17] A. Martinez and E. Fernández, *Learning ROS for Robotics Programming*. Packt Publishing Ltd, 2013.

[18] "Urdf/XML/Link." http://wiki.ros.org/urdf/XML/Link.

[19] "Urdf/XML/Joint." http://wiki.ros.org/urdf/XML/joint.

[20] "RethinkRobotics baxter common." https://github.com/RethinkRobotics/baxter_common/tree/master/baxter_description.

[21] J. F. Nethery and M. W. Spong, "Robotica: a mathematica package for robot analysis," *Robotics & Automation Magazine, IEEE*, vol. 1, no. 1, pp. 13–20, 1994.

[22] P. Corke, "A robotics toolbox for MATLAB," *Robotics & Automation Magazine, IEEE*, vol. 3, no. 1, pp. 24–32, 1996.

[23] M. M. Wickham, "A study of haptic interactions with an under actuated robot in three dimensions," 2010.

[24] "Tuck Arms Example." https://github.com/RethinkRobotics/sdk-docs/wiki/Tuck-Arms-Example.