Human Arm Imitation by a 7-DOF Serial Manipulator

Ayush Varshney, Ritesh Gautam, Dr. Amitabha Mukerjee

Indian Institute of Technology, Kanpur
CS365-Artificial Intelligence, April-2013

Abstract

This document is a report of the project done in partial fulfillment of the course of Artificial Intelligent (Spring-2013). It contains all the details of the algorithms and methods used in the project, as well as description of the robot used for experiments. It also contains some useful information on solving inverse kinematics problems. The project aims at developing a system for training industrial manipulator robots through a human controller. The imitation of the human arm by the robot arm can help remove the physical limitations of human working capability. Microsoft Kinect Sensor is used in the project, along with OpenNI and Nite libraries that provide the necessary functions to get 3D (x,y,z) coordinates of each human joint (20 in total). Amtec PowerCube Robot serves as the 7-DOF manipulator. The code was implemented using Microsoft Visual Studio and communication to the bot through Serial Communication using a RS232 Serial Cable. Satisfactory imitation was achieved which was mainly limited due to the speed limitations of the robot manipulator.

I. INTRODUCTION

Robotic arms are mechanically controlled devices designed to replicate the movement of a human arm. The devices are used for lifting heavy objects and carrying out tasks that require extreme environment and expert accuracy. The robotic arm most often is used for industrial and nonindustrial purposes. The imitation of the human arm by the robot arm can help remove the physical limitations of our working capability. We would just have to do as we want our robot to do and it would imitate our motion. Imitation is mainly useful when the robot has various different types of tasks which it has no prior experience with. It could be taught to perform those tasks by imitation and learning it once,then it can repeat the task any number of times. For example if we have to relocate a heavy object then we just need to imitate picking up the object in the air the arm would do the same. In this case when we only had to move one object making the robot imitate would be better than coding the arm to identify the object and then pick it up through some complicated path planning algorithm which would be preferred in multiple same type of tasks. So if we have a variety of task at our hands then we could use imitation which would complete our task with less overhead, with just a little loss in accuracy due to human error. The objective of this project was to make the Amtec PowerCube Manipulator imitate the human arm which is been processed through Microsoft Kinect. Our model can be further evolved to imitation learning. Here, a new approach has been employed to train robots where in our robot is made to learn how to perform a particular task by making it imitate a human arm. The Microsoft Kinect Sensor gives us data of four coordinates on the human arm - the shoulder joint, the elbow joint, the beginning of the palm, and the approximate center of it. Using the known DH Parameters of a human arm, we calculate the joint angles and transfer the angles through serial communication to the manipulator which then causes each of its motors to rotate to the required angle. Thus the human controls the robotic arm and uses it to perform the particular task.

Figure 1: The Amtec powercube robot

All the moves made by the arm from its initial position up to the completion are stored and can now be used to perform the task again any number of times. Thus we enable our robot to perform the repetitive task without using any actual sensory or path-planning system and the errors involved with it. Also when the task changes - we don't have to change the system but only train the robot once again.

II. THEORY

Inverse Kinematics

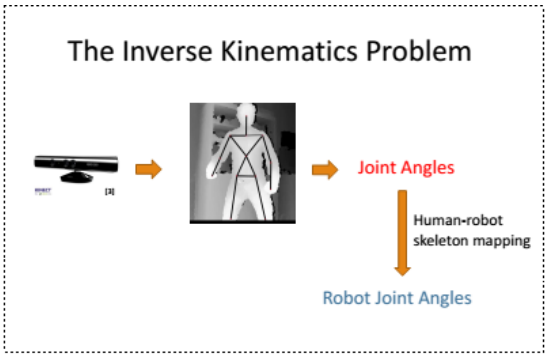

Figure 2: Our problem[1]

The inverse kinematics problem (IKP) for a robotic manipulator involves obtaining the required manipulator joint values for a given desired end-point position and orientation. It is usually complex due to lack of a unique solution and closed-form direct expression for the inverse kinematics mapping. [1]

Denavit-Hartenberg(DH)Parameters

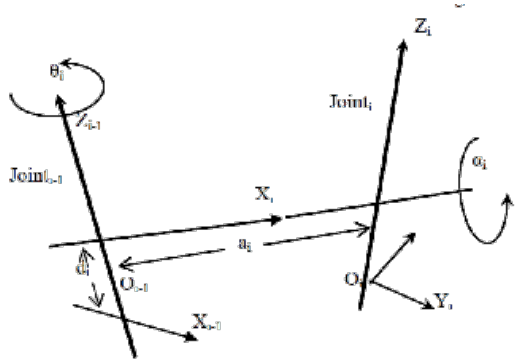

Figure 3: The relation between consecutive coordinates[2]

The Denavit-Hartenberg parameters (also called DH parameters) are the four parameters used to show the relation between consecutive links of a robot manipulator.Each link is given four values to describe its relation with the next link,these parameters are link offset(d_i), joint angle(θ_i), link length(a_i), link twist(α_i).

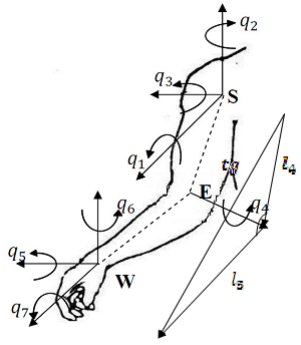

FIGURE 2. Kinematic chain of human arm

Figure 4: [2]

Frame	q_i	d_i	a_i	α_i
1	q_1	0	0	$\pi/2$
2	$q_2+\pi/2$	0	0	$\pi/2$
3	q_3	0	0	$-\pi/2$
4	q_4	0	L4	$\pi/2$
5	q_5	0	L5	$-\pi/2$
6	$q_6-\pi/2$	0	0	$-\pi/2$
7	q_7	0	0	$-\pi/2$

Table-I:Values of DH parameters of human arm [2]

Link	a_i	d_i	α_i	θ_i
1	0	d_1	-90	θ_1
2	0	0	90	θ_2
3	0	d_3	-90	θ_3
4	0	0	90	θ_4
5	0	d_5	-90	θ_5
6	0	0	90	θ_6
7	0	0	d_7	θ_7

Table-II:Values of DH parameters of Amtec powercube robot [3]

III. SOLVING THE IK PROBLEM

The transformation matrices of the human arm are:

$$A_1=\begin{bmatrix}c_1&-s_1&0&0\\0&0&-1&0\\s_1&c_1&0&0\\0&0&0&1\end{bmatrix}\ A_2=\begin{bmatrix}c_2&-s_2&0&0\\0&0&-1&0\\s_2&c_2&0&0\\0&0&0&1\end{bmatrix}$$

$$A_3=\begin{bmatrix}c_3&-s_3&0&0\\0&0&1&0\\-s_3&-c_3&0&0\\0&0&0&1\end{bmatrix}\ A_4=\begin{bmatrix}c_4&-s_4&0&l_4\\0&0&-1&0\\s_4&c_4&0&0\\0&0&0&1\end{bmatrix}$$

$$A_5=\begin{bmatrix}c_5&-s_5&0&l_5\\0&0&1&0\\-s_5&-c_5&0&0\\0&0&0&1\end{bmatrix}\ A_6=\begin{bmatrix}c_6&-s_6&0&0\\0&0&-1&0\\s_6&c_6&0&0\\0&0&0&1\end{bmatrix}$$

$$A_7=\begin{bmatrix}c_7&-s_7&0&0\\0&0&1&0\\-s_7&-c_7&0&0\\0&0&0&1\end{bmatrix}$$

Here c_i stands for cosθ_i and s_i stands for sinθ_i.
Now we calculate the joint angles:
$tg=\sqrt{(dx^2+dy^2+dz^2)}$
and $tg=l_4^2+l_5^2-2l_4l_5cos\theta_4$
So,
$\theta_4=cos^{-1}\frac{tg^2-l_4^2+l_5^2}{2l_4l_5}$
$-dx = s_1(l_5c_4s_3+l_4s_3)+c_1(l_5c_4c_3s_2+l_5c_2s_4+l_4c_3s_2)$
$dz=c_1(l_5c_4s_3+l_4s_3)-s_1(l_5c_4c_3s_2+l_5c_2s_4+l_4c_3s_2)$
Let $k_1=l_5c_4s_3+l_4s_3$
$k_2=l_5c_4c_3s_2+l_5c_2s_4+l_4c_3s_2$
Let $rcos\phi=k_1$
$rsin\phi=k_2$
Now,
$r(sin\theta_1cos\phi+cos\theta_1sin\phi=-dx)$
$r(cos\theta_1cos\phi+sin\theta_1sin\phi=-dz)$
$rsin(\theta_1+\phi)=-dx$
$cos(\theta_1+\phi)=dz$
$tan(\theta_1+\phi)=\frac{-dx}{dz}$

$$\theta_1=\begin{cases}tan^{-1}(\frac{-dx}{dz})-\phi & \text{if } dz\neq 0\\ -pi/2-\phi & \text{if } z=0\end{cases}$$

$r=\frac{-dx}{sin(\theta_1+\phi)}$
$l_5cos\theta_4sin\theta_3+l_4sin_3=rcoos\phi$
$\theta_3=sin^{-1}\frac{rcos\phi}{l_5cos\theta_4+l_4}$
$s_2(l_5c_4c_3+l_4c_3)+c_2(l_5s_4)=k_2$
$-c_2(l_5c_4c_3+l_4c_3)+s_2(l_5s_4)=dy$
Let $k_3=l_5c_4c_3+l_4c_3$
$k_4=l_5s_4$
$s_2k_2+c_2k_4=k_2$
$-c_2k_3+s_2k_4=dy$
Solving the above equations using matrix inversion we let $sin\theta_2=k_5$
$cos\theta_2=k_6$

$$\theta_2=\begin{cases}tan^{-1}\frac{-k_5}{k_6} & \text{if } k_6\neq 0\\ -pi/2 & \text{if } k_6=0\end{cases}$$

We now find $\theta_5,\theta_6,\theta_7$

$$A_5*A_6*A_7=\begin{bmatrix}r_{11}&r_{12}&r_{13}\\r_21&r_22&r_{23}\\r_{31}&r_{32}&r_{33}\end{bmatrix}$$

Using the DH parameters,we get equations for r_ij
$sin\theta_6=r_{22}$
$\theta_6=sin^{-1}r_{22}$
$r_{13}=c_5*c_6$
$r_{33}=-c_6*s_5$
$tan\theta_5=-\frac{r_{33}}{r_{13}}$

$$\theta_5=\begin{cases}tan^{-1}-\frac{r_{33}}{r_{13}} & \text{if } r_{13}\neq 0\\ pi/2 & \text{if } r_{13}=0\end{cases}$$

$-c_6*c_7=r_{21}$
$c_6*s_7=r_{22}$

$$\theta_7=\begin{cases}tan^{-1}-\frac{r_{22}}{r_{21}} & \text{if } r_{21}\neq 0\\ pi/2 & \text{if } r_{21}=0\end{cases}$$

Now we have all the $\theta's$ in the term of one variable ϕ

$$p(\theta)=p^d$$

where, p^d->target position in space
$p(\theta)$->calculated position as function of joint space

$$R(\theta)=R^d$$

where, R^d->target orientation in space
$R(\theta)$->Calculated orientation
The error function can be defined as

$$e(\theta) = \begin{cases} p^d - p(\theta) & \text{(positional)} \\ a(R^d * R(\theta)^T) & \text{orientational} \end{cases}$$

$$RT = R^d * R(\theta)^T(q) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_21 & r_22 & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

We can now get the value of ϕ by solving the following equation:

$$e(\theta) = 0$$

Now we can use the value of ϕ to get the values of $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7$.[2]

IV. TESTING THE ALGORITHM

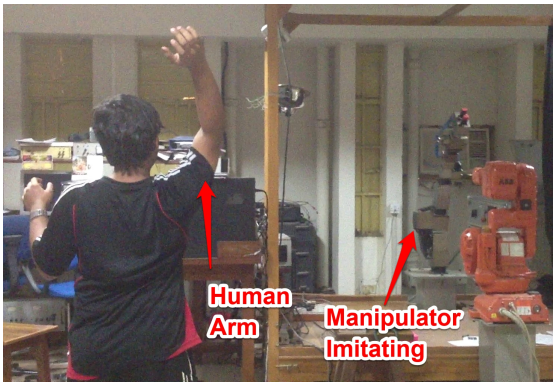

Figure 5: Robot imitating the human

Sending Control to the Robot

We have used serial input to send data to the robot arm. There is no level control available for the robot. But thanks to the previous work done on the robot, we were able to find a cpp library for operating the manipulator. To operate the code from inside a c++ program.

Include *RealCube.h* and *PCube.cpp* [4]
//create a robot object
RealCube $P(0)$;
//home the robot - bring to initial position
$P.check\_move\_home()$;
//Send the joint angles
$P.MovePosition(vTh)$; // vTh is an array of the joint angles

V. PERFORMANCE ANALYSIS

1. The robot lags as it cannot keep up with the speed of the human it is imitating because of the joint speed constraints.
2. The complete algorithm could not be tested because of some mechanical constraints but the sample outputs of joint angles show a good estimate.
3. The Kinect Sensor works only in the range 1.2m to 3.5m when reading the arm joints - which puts some constraint on the setup.
4. Some errors occur due to error in data from Kinect especially when the joints align in a straight line perpendicular to the kinect field of view.
5. One major problem in the current implementation is that we are not taking any feedback from the robot with regards to it's current position and thus are continuously sending data to the robot - which leads to a lot of vibrations and also some fast gestures are missed.

VI. CONCLUSION

Good imitation of the human arm by the robotic arm can be achieved only when we control the speed being sent to the robot and not just the final position, otherwise, it is a forced imitation. The inverse kinematics model presented here tries to solve the joint angle problem analytically but the solutions to the non-linear equation (invloving ϕ), sometimes cause problem due to singularities. Also in an IK problem the solution may not exist, thus in such a situation especial workarounds need to be defined to achieve good imitation. The Skeleton mapping causes a little problem especially due to the 3-DOF shoulder joint of the human arm. Thus, to achieve a good imitation, we not only need the current position feedback but also the velocity feedback from each motor. Also calculating the required speed to perform a gesture is a difficult task - a 2nd Order IK Problem.

VII. SUGGESTIONS FOR FURTHER RESEARCH

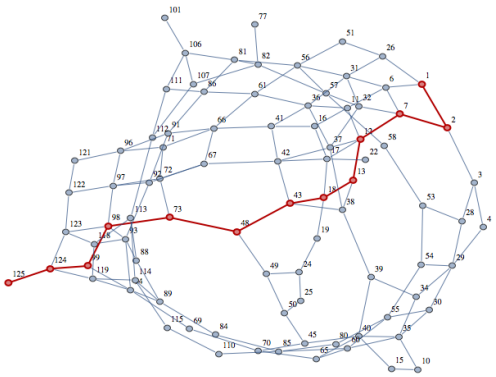

Figure 6: Local Optimization Problem

Our implementation can be futher used to train industrial robots using human trainers. As the human trainers could do some redundant task which is not needed for the completion of the final task to be performed by the robot. We have an idea of local optimization in which the redundant states of the robot can be removed to perform "local optimization" of the complete motion. For this, we can take the states of the robot after a time quantum to be a node, so our problem now reduces to finding the shortest path, such that we are given one of the paths. We could try to generate some more paths using the given nodes and then find the shortest path. The problem in this could be that the robot might now go through an obstacle, to avoid this we could mark some key nodes in our graph which should not be removed while optimizing.

VIII. REFERENCES

[1] From http://www.microsoft-careers.com /content/rebrand/hardware/ hardware-story-kinect/ [2013]

[2] From Human Arm Inverse Kinematic Solution Based Geometric Relations and Optimization Algorithm-Mohammed Z. Al-Faiz, Abduladhem A.Ali & Abbas H.Miry [2011]

[3] From Visual motor control of a 7DOF redundant manipulator using redundancy preserving learning network Swagat Kumar, Premkumar P., Ashish Dutta and Laxmidhar Behera - Robotica / Volume 28 / Issue 06 / October 2010, pp 795 810

[4] From home.iitk.ac.in/ vayush/cs365/project/ codes [2013]

[5] From http://mathematica.stackexchange.com /questions/4084/finding-a-not-shortest-path-between-two-vertices [2013]