Operating and Visualizing Robotic Manipulator by an OS independent Web Based Application

MM802 Final Project Report

April 18, 2019

Kumar Halder
dept. of Computer Science, Multimedia
University of Alberta
Edmonton, Canada
khalder@ualberta.ca

Gauravi Chavan
dept. of Computer Science, Multimedia
University of Albert*a*
Edmonton, Canada
gauravi@ualberta.ca

Abdul Malek
dept. of Computer Science, Multimedia
University of Albert*a*
Edmonton, Canada
amalek1@ualberta.ca

Abstract— **Robot Operating System (ROS) is widely used in several industries for automating their systems with robotic manipulators. Although, ROS is open source, it is currently being supported only by Ubuntu owing to its compatibility with ROS's design specifications. Efforts initiated towards operating the arm using any other operating system besides Ubuntu have not been successful. Therefore, in spite of its usability this system cannot reach out to a huge section of clients who use Windows and MacOS. Therefore, to refrain themselves from losing such clients, many industries are compelled to hire a robot engineer just to operate this system in Ubuntu. However, the industries end up spending a lot of money on this additional manual resource's remuneration. Through this project, we are trying to find a solution for this highly relevant and practical issue.**

We have proposed a system that computes the orientation and position of each of the 6 joints of our robotic manipulator -'Aubo i5', wired to the server's system while it picks up and drops an object using Inverse Kinematics. This data is then communicated to the client's side to visualize it in a 3D web application through TCP connection.

Keywords—Inverse Kinematics, Aubo i5, Mesh Streaming, 3D Visualization using JavaScript

I. INTRODUCTION

The current solutions that the industries have come up with, certainly pave way for a healthy debate in the technical world. Is it wise and cost efficient for the industries to recruit manual resources over mere incompatibility issues? Does the existing researchers' work conserve and optimally utilize network resources? Have the researchers come up with any deliverables wherein they try to eliminate the dependency of an expert to operate the system? A research enthusiast would find it interesting to search answers to these questions. Given the attention required for a problem of this calibre, makes it necessary to conduct thorough research for accomplishing its objectives.

An interesting amalgamation of two engineering sectors, as this, will certainly appeal to the folks interested in networking and robotics alike. Moreover, our work will certainly engage a math enthusiast who likes solving trigonometry and algebraic equations.

The report is divided into fourteen sections wherein the first three sections give you an overview of our research work, Section IV will describe the Literature Review we conducted and the approach that we chalked out through them, Section V talks about our Development Environment, Section VI talks about our Implementation Details, Section VII describes the Novelty of our project, section VIII talks about Results Evaluation, section IX includes Challenges Faced, section X outlines Roles and Responsibilities of each group member, section XI describes our Deliverables, section XII talks about setting up the environment and configuration details, Section XIII states our Projects Status , Section XIV includes Future Scope, this is then followed by Acknowledgement, References and Appendix.

II. PROBLEM STATEMENT

We intend to deploy an OS independent web interface for controlling and visualizing the state of robotic arm - "Aubo i5" while it lifts and drops an object by optimally utilizing the network resources. Our primary objective lies in finding the orientation described by each joint such that the end detector of Aubo i5 reaches the object. This would be done using Inverse Kinematics.

III. PROJECT SCOPE

Our proposed system deals with identical objects of fixed shape and size (as we focus on how the object is being lifted and not on what is being lifted) and fixed position of the arm. Our project is divided into three sections.

1. Computing Inverse Kinematics
2. Mesh Streaming through TCP Connection between Server and Client
3. 3D Visualization in Web Application

IV. LITERATURE REVIEW

In [1] the authors demonstrate the use of Multipath TCP in robotics communication. MPTCP uses all available communication interfaces concurrently for communication. Its use in robotics improves the connection reliability as well as the throughput.

In [2] the authors presented a software module designed for efficient and convenient visualization of 3D models inside the web browser environment. It is written purely in JavaScript and takes advantages of the new HTML5 standard. Their proposed solution based on progressive mesh streaming that is compared with server-side rendering approach.

In [3] the researchers' work addresses the inverse kinematics problem for the 7 Degrees of Freedom Barrett Whole Arm Manipulator with link offsets. A parametric solution for all possible geometric poses is generated by them for a desired end-effector pose (position and orientation). The set of possible geometric poses are completely defined by three circles in the Cartesian space by the authors of [3]. An analytical method of identifying a set of feasible poses for some joint-angle constraints is also addressed in this paper.

In [4] the authors propose an analytical methodology of inverse kinematic computation for 7 DOF redundant manipulators with joint limits. Here, a closed-form inverse kinematic solution is derived based on a parameterization method.

The authors of [6] used a combination of analytical and numerical methods to solve generalized inverse kinematics problems including position, orientation, and aiming constraints. Their proposed methods result in faster and more reliable algorithms than conventional optimization-based techniques.

Researchers in the past [7] have come up with a methodology to solve Inverse Kinematics which is based on a combination of two nonlinear programming techniques and the forward recursion formulas, with the joint limitations of the robot being handled implicitly as simple boundary constraints. This method [7] is numerically stable since it converges to the correct answer with virtually any initial approximation, and it is not sensitive to the singular configuration of the manipulator. In addition, this method is computationally efficient and can be applied to serial manipulators having any number of degrees of freedom.

In [8] the inverse kinematic problem associated with manipulators of arbitrary architecture is discussed. Its formulation is based on invariants in the rotational part of the closure equations, which produces a formally overdetermined nonlinear algebraic system. This leads to a reduced number of computations, which allows an efficient numerical solution of the problem.

In [9] a geometric approach for deriving a consistent joint solution of a six-point PUMA1 robot is being presented. This approach calls for the definition of various possible arm configurations based on the link coordinate systems and human arm geometry. These arm configurations are then expressed in an exact mathematical way to allow the construction of arm configuration indicators and their corresponding decision equations. The arm configuration indicators are prespecified by a user for finding the joint solution. These indicators enable one to find a solution from the possible four solutions for the first three joints and a solution from the possible two solutions for the last three joints.

We planned out what approach would be suitable for us by referring to these research papers.

V. Development Environment

1. We used an Ubuntu system on server side and Windows system on client side.
2. "Aubo i5" - robotic arm was physically wired to the server Ubuntu system.
3. ROS was used for controlling the robotic arm.
4. We used Matlab for developing and testing our Inverse Kinematics and Forward Kinematics programs.
5. We used Unity 3D for creating a virtual arm that was used for testing our calculations during the initial phase of the project.
6. Three.js for 3D visualization embedded in the web application for visualizing the arm at the client's side.
7. For socket communication we used Ros-bridge for a full duplex channel over a single TCP IP connection between client & server.

VI. Implementation Details

Our implementation is divided into three sections:

1. Inverse and Forward Kinematics
2. TCP connection using Socket Programming
3. Visualization in Web Application

1. Inverse and Forward Kinematics

We discuss in detail how we identify our problem, how we implement it and on what basis we choose our performance metrics to demonstrate our results in graphical representation as under:

A. We first send two 3D coordinates for pickup and drop-off locations from the client side.

Pick up coordinates: p1 (a1, b1, c1) and drop-off coordinates: p2 (a2, b2, c2)) are sent from client's side

B. On obtaining the coordinates at the server side, the end detector of Aubo i5 reaches out to each of the two coordinates separately

For joint 'i':

The state of pickup object is $\overline{\nabla}i = pi\ .\ \Theta i$

The state of drop-off object is $\overline{\nabla}i1 = pi1.\ \Theta i1$

Therefore, say $\mu i = |\ \overline{\nabla}i1 - \overline{\nabla}i\ |$ is the transform vector for joint 'i'.

Each of the six joints describe a specific orientation for reaching out to the coordinates, first pickup followed by drop off. The state (described angle and position) of each joint will be provided by the arm that is physically wired to the server. (Whether the arm uses sensors or some other elements to determine the state is a confidential information within the robotic arm’s manufacturer circle).

The transform vector after all 6 joints displace and orient themselves:

$$\mu 1 = \sum \mu i \text{ for } i=1 \text{ to } 6$$

The time function of the transform vector from pickup to drop off is:

$$\frac{d\,(\mu 1)}{dt} = \frac{d}{dt}\left[(\overline{\nabla} i1/ |\hat{u}|) - (\overline{\nabla} i / |\hat{u}|)\right] \ldots\ldots\ldots\ldots\ldots..(1)$$

Where a, b, c are the joints’ positions with respect to its previous positions a = | a1 - a0 |, b = | b1 - b0 | and c = | c1 - c0 |. Ɵ values are its orientation values. You would notice that there are nine different functions of Ɵ1 to Ɵ6 in their corresponding matrices.

C. After receiving the orientation and using the coordinates that it already had; “Inverse Kinematics” is performed to evaluate whether the angle computed at the server side is correct or not.

It already has pick up coordinates: p1 (a1, b1, c1) and dropoff coordinates: p2 (a2, b2, c2)). So, using the resultant state of arm, inverse kinematics is performed and it is verified whether it gets the same theta values as set in forward kinematics.

For joint ‘j’:

The state of pickup object is $\overline{\nabla} j = pj.\ \Theta j$

The state of dropoff object is $\overline{\nabla} j1 = pj1.\ \Theta j1$

Therefore, say $\mu j = |\overline{\nabla} j1 - \overline{\nabla} j\,|$ is the transform vector for joint ‘j’

The transform vector after all 6 joints displace and orient themselves:

$$\mu 2 = \sum \mu j \text{ for } j=1 \text{ to } 6$$

The time function of the transform vector from pickup to drop off is:

$$\frac{d\,(\mu 2)}{dt} = \frac{d}{dt}\left[(\overline{\nabla} j1/ |\hat{u}|) - (\overline{\nabla} j / |\hat{u}|) \right) \ldots\ldots\ldots\ldots\ldots(2)$$

For determining the time lag occurring between the start time of complete state in server and client side, we note the time for the pickup $\Sigma\overline{\nabla} j$ for j=1 to 6 at client’s side and the drop-off time at server’s side $\Sigma\overline{\nabla} i$ for j=1 to 6. This is how we would know what the time lag between real and virtual sides is. (System is not set exactly in real-time due to resultant state completion lag ‘k’)
The state of drop off object is
$\Sigma\overline{\nabla} i = \Sigma pi.\ \Sigma\Theta i$……………..……j = 1 to 6

The state of pickup object is

$\Sigma\overline{\nabla} j = k.\ \Sigma pj.\ \Sigma\Theta j$…………..….j = 1 to 6

Hypothesis H0= | t(μ2) - t(μ2) | = 0……….. Then we conclude time taken for complete state change from pickup to drop-off in both client and server side is same.

Hypothesis H1= | t(μ2) - t(μ1) | ≠ 0……….. Then we conclude time taken for complete state change from pickup to drop-off in both client and server side is different.
Case H1 could happen due to various network conditions; which we would investigate while testing.

To sum up, for result evaluation we check the following 2 events which we use as our performance metrics for testing results:
Note: Complete state indicates one complete cycle from pickup to drop off time.

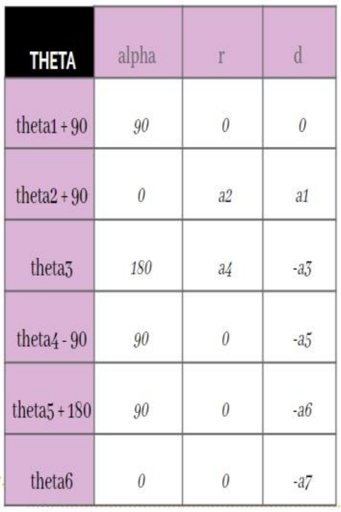
(1.) Time lag occurring between the start time of complete state in server and client side.

(2.) Time to complete the state at both sides since the temporal factor is not passed.

(3.) We compare between the network resources used by us and other methods.

(4.) We observe the response of 3D scene in web browser

D. We will set values of Ɵ here so that we can verify them through inverse kinematics. We then compute transform matrices for six joints using DH Parameters. These values are shown as under:



THETA	alpha	r	d
theta1 + 90	90	0	0
theta2 + 90	0	a2	a1
theta3	180	a4	-a3
theta4 - 90	90	0	-a5
theta5 + 180	90	0	-a6
theta6	0	0	-a7

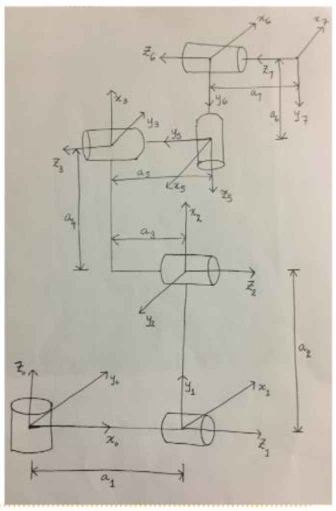


Fig. DH Parameters

The values between the centers of joints in Aubo i5 are fixed. These values comprise {a1, a2, a3, a4, a5, a6}. ‘THETA’ represents the phase angle between the x-axis while ‘alpha’ represents the phase angle between the z-axis of the successive joints. ‘r’ represents the distance between the x-axis whereas ‘d’ represents the distance between the z-axis of the joint in question with respect to its previous joint. It should be noted here that this model demands that each time the z-axis of the joint and the previous joint in question should intersect the x-axis of the joint that is being considered. Only then we can calculate ‘r’ and ‘d’. Using trigonometry, we filled up this table by referring to the figure of Aubo-i5 on the right.

We substitute the values from the above table for each of the 6 joints in the matrix ‘W’ below:

W=

[(cos(THETA)), (-sin(THETA))*(cos(alpha)) , (sin(THETA))*(sin(alpha)), r* (cos(THETA)) ;

(sin(THETA)), ((cos(THETA) * cos(alpha))) , ((-cos(THETA) * sin(alpha))), r * (sin(THETA));

0, (sin(alpha)) ,(cos(alpha)), d ;

0, 0 , 0 , 1 ;]

The above matrix ‘W’ has been pre-formulated by the past researchers and experts in this field. It should be noted that all these angles are in degrees. Using “Forward Kinematics” we shall now compute 4*4 homogeneous transform matrix for each of the six joints.

E. We now substitute the pH values from the table to solve the equations using ‘Algebraic Approach’. We have the 6 transform matrices in the following representation for each of the six joints:

Σi,n=1 to 6 **Ti = [fn(Θi) fn(Θi) fn(Θi) a**

fn(Θi) fn(Θi) fn(Θi) b

fn(Θi) fn(Θi) fn(Θi) c

0 0 0 1]

After substituting DH Parameters in W, we get the matrices in the following format:

T1= [-sin(Θ1) 0 cos(Θ1) 0;

cos(Θ1) 0 sin(Θ1) 0;

0 1 0 0;

0 0 0 1]

T2= [-sin(Θ2) -cos(Θ2) 0 -a2*sin(Θ2);

cos(Θ2) -sin(Θ2) 0 a2* cos(Θ2);

0 0 1 a1;

0 0 0 1]

T3= [-sin(Θ2) -cos(Θ2) 0 -a2*sin(Θ2);

cos(Θ2) -sin(Θ2) 0 a2* cos(Θ2);

0 0 1 a1;

0 0 0 1]

T4= [sin(Θ4) 0 -cos(Θ4) 0;

-cos(Θ4) 0 -sin(Θ4) 0;

0 1 0 -a5;

0 0 0 1]

T5= [-cos(Θ5) 0 -sin(Θ5) 0;

-sin(Θ5) 0 cos(Θ5) 0;

0 1 0 -a6;

0 0 0 1]

T6= [cos(Θ6) -sin(Θ6) 0 0;

sin(Θ6) cos(Θ6) 0 0;

0 0 1 -a7;

0 0 0 1]

The resultant matrix is as follows:

T_final = [mx nx ox px

my ny oy py

mz nz oz pz

0 0 0 1]

The first 3*3 matrix in T_final representation describes the rotation of the end effector, and last column p, py and pz represent the position of the end effector.

We have, T_final = T1 * T2 * T3 * T4 * T5 * T6

These, however, are long tedious equations. We, therefore, pre-multiply and post-multiply the inverse of a specific matrix to check if the resultant equations are simpler. We thereby choose the appropriate equations that could be solved to generate 6 Θ values.

Pre-multiplying by inverse transform of a certain matrix say T1

inv(T1) * T_final = inv(T1) * T1 * T2 * T3 * T4 * T5* T6 ;.........(1)

inv(T1) * T_final = T2 * T3 * T4 * T5* T6 ;..............(2)

inv(T2)* inv(T1) * T_final = T3 * T4 * T5* T6 ………...(3)

……………..

……………...

We find Θ1to Θ6 values based upon the equations that we derive and check whether these values match with ones that we had pre-set in forward kinematics.

F. Once evaluation is done, we use both the parameters: ‘orientation’ and ‘position of coordinates’ to visualize the state of arm (in 3D) at the client’s side through a web interface.

We send the resultant state only after the end defector reaches the coordinates after all its 6 joints describe a specific orientation and position for pickup and drop off. It should be noted that unless a state is complete the state information will not be passed to the client's side as it helps in handling state abortion events (No intermediate state information would be sent).

2. *TCP connection using Socket Programming*

For communication between server and client, we used TCP connection through ROS-bridge on the server side and roslibjs on the client side. We used TCP because it provides a full duplex stream service with automatic error handling, retransmission and guarantee of reliable delivery. Performance was a big issue in our client side, so instead of calculating everything on the client side, we calculate the minimum and only send raw data from client to server. In the server side we calculated the inverse kinematics for this robotic manipulator and have sent this information of result again on the client side, to visualize that on web interface.

3. *Visualization in Web Application*

We have created a 3D interface for the user to define location and orientation for the robot to pick up the object. For the report, we have used skeleton representation of Aubo i5 arm, in the scale of centimeters. We used three.js and used its primitive object types with groups to represent the skeleton of the robotic manipulator. For defining the object position and location, we used html buttons for movements as well as input field to directly manipulate position and orientation. Given Rx, Ry and Rz the three individual rotation components of the object of theta1, theta2 and theta3 each along X, Y and Z axis respectively; the final rotation matrix of the object can be defined as,

R = [Rx*Ry*Rz]
Where Rx = [[1 0 0]; [0 cos(theta1) -sin(theta1)]; [0 sin(theta1) cos(theta1)];]
Ry = [[cos(theta2) 0 sin(theta2)]; [0 1 0]; [-sin(theta2) 0 cos(theta2)];]
Rz = [[cos(theta3) -sin(theta3) 0]; [sin(theta3) cos(theta3) 0]; [0 0 1];]

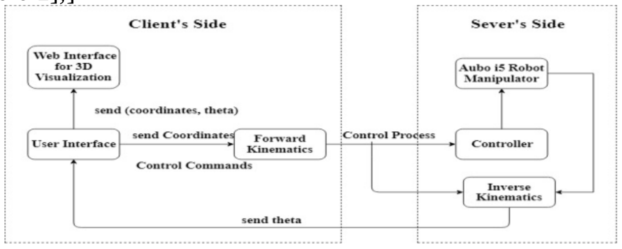


Fig: Our Proposed System

VII. NOVELTY FACTOR

1. We believe that our novelty lies in our application domain wherein a layman or a person with even minimal knowledge in this domain can operate this system from the client side with much ease. Although, computing the orientation and position of each joint in the robotic arm using "Inverse Kinematics" has been tried by a number of researchers, no one has implemented it for Aubo i5 till now.

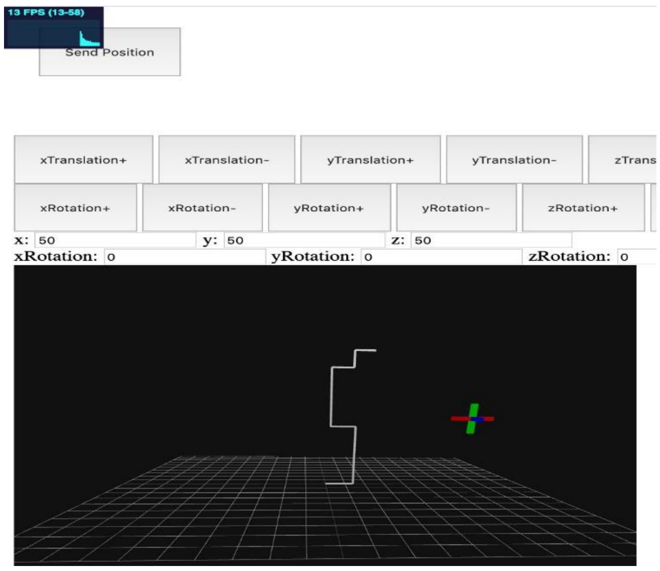
2. We have computed these values using pure math - "Algebraic Approach", which was not used to its truest potential until the time of writing this report. Previously, no Inverse kinematics has been published in any publication for the robotic manipulator "Aubo i5", since it is relatively new in market.

3. Most researchers in the past[5] have used camera for sending phase and position information from the server's side for evaluating if the computed angle is correct. This utilizes considerable network resources. On the contrary, we pass the mesh information (position and angle) through TCP IP between client and server and verify whether the computed angle is correct using "Forward Kinematics technique" in the development phase that uses our designed math functions for verification.

4. A few other robotic manipulators have magnets attached to their end detectors, that attract objects within its magnetic locus even before the last joint has taken its position and angle. We eliminate such ambiguous events by using "Aubo i5" which straight away grabs the objects. (No magnets here)

VIII. RESULT EVALUATION

We demonstrate our results in the 3D interface of web application as below. This interface would be provided at the client's side where he can choose a position to the server's side:



x: 50 y: 50 z: 50
xRotation: 0 yRotation: 0 zRotation: 0

Fig. 3D State of arm before providing the position In Web Application

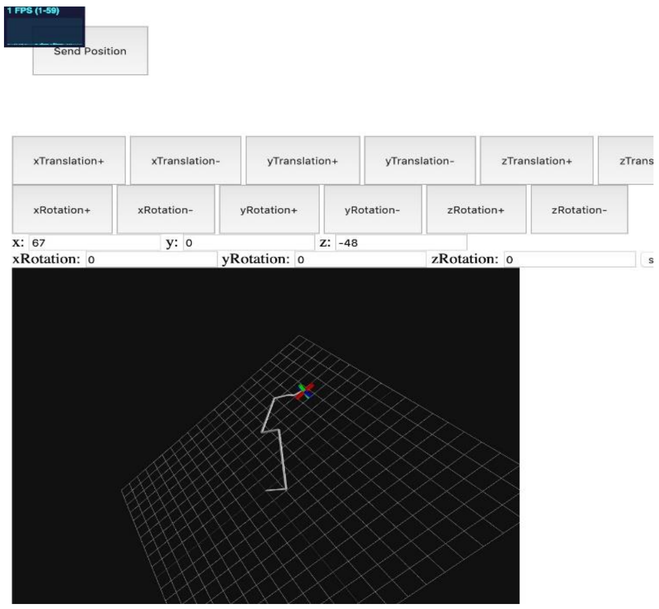


Fig. Orientation and position of the arm when it grabs the object

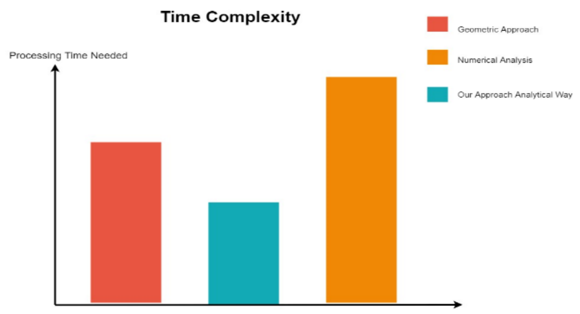


Fig. Comparison between time complexities of different methods for solving equations

On reviewing the papers of the research work conducted in the past, we figured out that other approaches like 'Geometric Approach' and 'Numerical Analysis' take longer to execute. We indicate in the figure how our approach demonstrates optimal time complexity. Through this, we contribute towards constructing an algorithm that promises optimal time complexity.

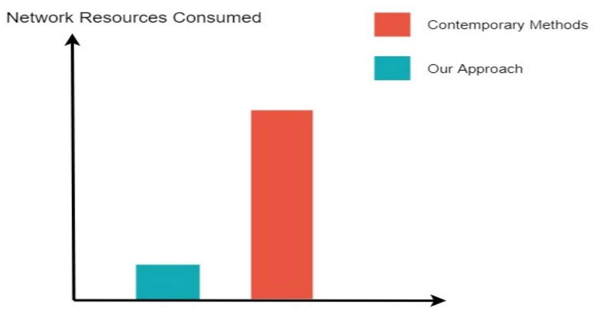
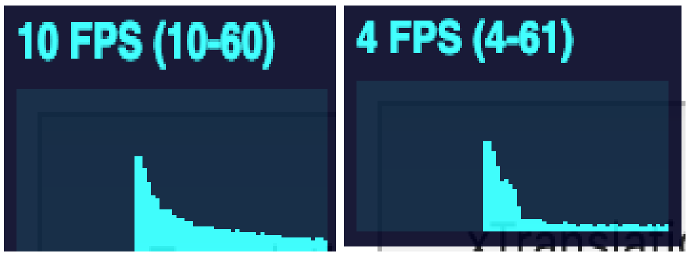


Fig. Comparison between the Network Resources Utilization between our proposed approach and [5]

It should be noted that, since we use purely math functions (unlike other models that uses camera [5]) for computing orientation and position of joints, we hardly consume any network resources and thus contribute towards reducing considerable network traffic.

Moreover, since we are automatizing the visualization system, we are contributing towards optimizing the profit margin and financial decision making of the industrial sector.

One of the biggest issues that should be noted is the response of 3D scene in a web browser. We used three.js library for rendering the 3D scene, that uses WebGL. The following graphs represent the frames per second (FPS) rendered in webGL. Both are recorded for first 30 seconds after refreshing the page in the browser, where the first graph shows FPS without any sort of scene manipulation and the second graph shows FPS when there was some user interfere along with the scene. They are recorded using stats.js, which is another javascript library for analysing the performance of the scene, wherein the camera view range of three.js library was between 0.1 to 1000. The platform used during this experiment is Macbook Pro with specification 2.7 GHz Intel Core i5, 8GB DDR3 memory and Intel Iris Graphics 1536 MB.



The performance is quite poor, when compared to the ideal FPS which is 60 FPS. Over time, the FPS tends towards 1-2 FPS, making it unreliable for commercial use. The performance is improved when the camera range of near to far plane is in between the range of 0 to 300, but still with time the performance degrades. Further improvements in the performance can be done by scaling the 3D scene to a factor of decimals, however this will compromise the representation of the precision of the robotic arm while it is performing.

IX. CHALLENGES FACED

1. The equations we get in step E of section 'Implementation Details' are really long and time consuming to solve. Also as the orientation values are interdependent, one incorrect value breaks the entire chain. We could figure out only at the end whether or not our computed angles are correct after verifying them with forward kinematics. We have briefly explained in the Appendix Section below about the equations that we choose to solve and finally how we came up with the orientation solution that we achieved.
2. There isn't any suitable library that animates the arm representation in javascript.
3. Researching on the suitable approach that utilizes less time complexity and network resources.

X. ROLES AND RESPONSIBILITIES

Sr No.	Role	Completed by Member
1.	Inverse and Forward Kinematics	All team mates

2.	Socket Programming using TCP IP	All team mates
3.	3D Visualization in Web Application	Kumar Halder
4.	Literature Review	Kumar Halder
5.	Report	Gauravi Chavan
6.	Documentation of Code and README	Abdul Malek

XI. DELIVERABLES

A web interface on a Windows or MacOS at the client’s side that controls the robotic arm and visualizes its state in 3D.

XII. SETTING UP THE ENVIRONMENT AND CONFIGURATION DETAILS

Despite several dependencies and configuration issues, we were able to set this system up and running. The objective of this project is to enable MacOS and Windows clients to operate the robotic manipulator which is connected to the Linux system at the server’s side. Given the broad and cross testing system as this, it is liable to have many module dependencies and to fix up the environment is not as easy as one would like it to be.

Please Note: Since this project’s environment has a lot c Linux dependencies and it is difficult to configure the system. Despite clear instructions provided on the README; we would encourage you to contact us if you are not able to set it up on your system. We would show you a demo on our system in such a case.

XIII. PROJECT STATUS

Our project work is divided into three sections. The first part comprises of sending data for communication between server and client using TCP through ROS-bridge on the server side and roslibjs on the client side. This section has been completed. The second section comprises of visualization and animation of the robotic manipulator in web application using javascript. We created a 3D interface for the user to define location and orientation for the robot to pick up the object. For this we have used skeleton representation of Aubo i5 arm. This section too has been completed. The final section comprises of Inverse Kinematics, wherein although we have computed and got the values of ϴ1 to ϴ6, we encounter a few issues due to more than one roots of the ϴs. We have almost completed fixing up this issue and therefore are heading towards the finishing steps of this section as well. Currently the final orientation for the positive roots of all ϴs is correct.

XIV. FUTURE SCOPE

1. Apart from ‘Numerical Analysis’, ‘Geometric Approach’ and ‘Analytical Approach’; researchers should come up with other procedures or libraries to solve equations and find orientation and position parameters.
2. Our current system demands a lot of computational power for rendering 3D visualization while animating the arm in javascript at the client’s side. Researchers may use other approaches/libraries in future that can work efficiently even with low computational power.
3. The arm could be trained using different shapes of objects for serving other purposes beyond networking and visualization. Moreover, researchers can also evaluate how accurate the arm gets with time.

ACKNOWLEDGEMENT

This research work is being encouraged and conducted at the company - “Rational Robotics”, Canada.

REFERENCES

[1] Aniruddh Rao, Visali M, Samar Shailendra, Anantha Simha; “Reliable Robotic Communication using Multi-Path TCP”; 2017.

[2] Bartosz Sawicki, Bartosz Chaber; “Efficient Visualisation of 3D models by Web Browser”; 2013

[3] Giresh K. Singh, Jonathan Claassens; “An analytical solution for the inverse kinematics of a redundant 7 DoF Manipulator with link offsets”; 2010

[4] Masayuki Shimizu, Hiromu Kakuya, Woo-Keun Yoon, Kosei Kitagaki , Kazuhiro Kosuge; “ Analytical Inverse Kinematic Computation for 7-DOF Redundant Manipulators with Joint Limits and Its Application to Redundancy Resolution”; 2008.

[5] Xiaoli Yang, Qing Chen, Dorina C. Petriu, Emil M. Petriu; “Internet Based Teleoperation of a Robot Manipulator for Education”; 2004.

[6] Deepak Tolani, Ambarish Goswami, Norman I.Badler; “ Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs”; 2000.

[7] L.-C.T. Wang, C.C. Chen; “A combined optimization method for solving the inverse kinematics problems of mechanical manipulators”; 1991.

[8] Jorge Angeles“On the Numerical Solution of the Inverse Kinematic Problem”; 1985.

[9] C.s.g. Lee, M. Ziegler; “Geometric Approach in Solving Inverse Kinematics of PUMA Robots”; 1984.

APPENDIX

Let’s consider an example where we pre-set the 6 theta values for the robotic manipulator in Forward Kinematics:

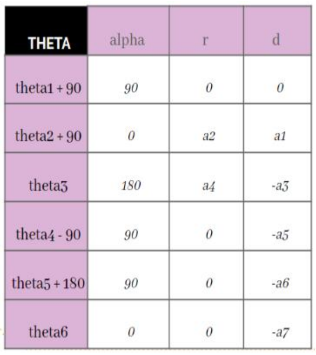
ϴ1= 55;	ϴ2= 37;	ϴ3= 81;
ϴ4= 28;	ϴ5= 90;	ϴ6= -55;

We have fixed values (perpendicular distances) between the c enters of each successive adjacent joints which are as follows and in the scale of millimeter which is provided by Aubo i5 manufacturer:

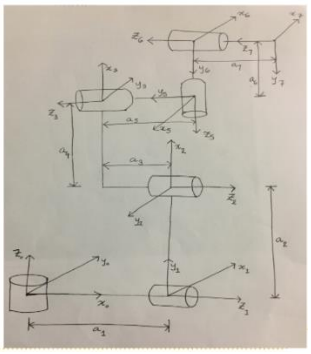
a1=140.5;	a2=408;	a3= 121.5;
a4 = 376;	a5 = 102.5;	a6= 102.5;
a7 = 94;		

You may see these values from the diagram below.

We may now refer to the following table to generate homogeneous transform matrix:



THETA	alpha	r	d
theta1 + 90	90	0	0
theta2 + 90	0	a2	a1
theta3	180	a4	-a3
theta4 - 90	90	0	-a5
theta5 + 180	90	0	-a6
theta6	0	0	-a7



We substitute the values 'THETA', 'alpha', 'r' and 'd' in the matrix below:

W=
[(cos(THETA)), (-sin(THETA))*(cos(alpha)), (sin(THETA))*(sin(alpha)), r* (cos(THETA)) ;

(sin(THETA)), ((cos(THETA) * cos(alpha))), ((-cos(THETA) * sin(alpha))), r * (sin(THETA)) ;

0, (sin(alpha)), (cos(alpha)), d ;

0, 0 , 0 , 1 ;]

After substituting DH Parameters in W, we get the matrices in the following format:

T1= [-sin(ϴ1) 0 cos(ϴ1) 0;

cos(ϴ1) 0 sin(ϴ1) 0;

0 1 0 0;

0 0 0 1]

T2= [-sin(ϴ2) -cos(ϴ2) 0 -a2*sin(ϴ2);

cos(ϴ2) -sin(ϴ2) 0 a2* cos(ϴ2);

0 0 1 a1;

0 0 0 1]

T3= [-sin(ϴ2) -cos(ϴ2) 0 -a2*sin(ϴ2);

cos(ϴ2) -sin(ϴ2) 0 a2* cos(ϴ2);

0 0 1 a1;

0 0 0 1]

T4= [sin(ϴ4) 0 -cos(ϴ4) 0;

-cos(ϴ4) 0 -sin(ϴ4) 0;

0 1 0 -a5;

0 0 0 1]

T5= [-cos(ϴ5) 0 -sin(ϴ5) 0 ;
-sin(ϴ5) 0 cos(ϴ5) 0 ;
0 1 0 -a6 ;
0 0 0 1];

T6= [cos(ϴ6) -sin(ϴ6) 0 0 ;
sin(ϴ6) cos(ϴ6) 0 0 ;
0 0 1 -a7 ;
0 0 0 1];

The resultant matrix is as follows:

T_final =[mx nx ox px
my ny oy py
mz nz oz pz
0 0 0 1]

We have, T_final = T1 * T2 * T3 * T4 * T5 * T6
And inverse of T1 matrix which is: T_inv = inv(T1)
For simplifying these equation we pre-multiply inverse of T1 on both sides

T_inv * T_final = T_inv * T1 * T2 * T3 * T4 * T5 * T6

Therefore, we get **LHS = T_inv * T_final ………..(I)**
RHS = T2 * T3 * T4 * T5 * T6……….(II)
We wisely choose the first equation from row 3 and column 3 of the LHS and RHS of matrix (**I**) and **(II)**:

(cos(ϴ1)*ox) / (cos(ϴ1)^2 + sin(ϴ1)^2) +

(oy*sin(ϴ1)) / (cos(ϴ1)^2 + sin(ϴ1)^2) =

- cos(ϴ5)…………..(1)

We choose the second equation from row 3 and column 4 of matrix **(I)** and **(II)** :

(cos(ϴ1)*px) / (cos(ϴ1)^2 + sin(ϴ1)^2)

+ (py*sin(ϴ1)) / (cos(ϴ1)^2 + sin(ϴ1)^2) = a1 - a3 +

a5 + a7 * cos(ϴ5)…………(2)

After solving (1) and (2), we get,

sin(ϴ1) = ((A*C)+ square_root(sq(A*C)

-(sq(B)+sq(C))*(sq(A)-sq(B)))) /(sq(B)+sq(C))

ϴ1 =sin_inverse(sin(ϴ1));

Where, A = a1-a3+a5

B = px + a7 * ox

C = py + a7 * oy

sq = square of the value

Consider equation (1) now:

(cos(ϴ1)*ox) / (cos(ϴ1)^2 + sin(ϴ1)^2) + (oy*sin(ϴ1)) / (cos(ϴ1)^2 + sin(ϴ1)^2) = - cos(ϴ5)

And as cos(ϴ1)^2 + sin(ϴ1)^2 = 1 and cos(ϴ1)^2 + sin(ϴ1)^2 = 1

Therefore, we get (cos(ϴ1)*ox)+ (oy*sin(ϴ1)) = - cos(ϴ5)........(3)

Substituting ϴ1 in (3) we get,

cos(ϴ5) = - (cos(ϴ1) * ox) + (oy * sin(ϴ1))

ϴ5 = cos_inverse(-cos(ϴ1) * ox - oy * sin(ϴ1));

From equation **(I)** and **(II)** matrices we choose a different set of 2 equations(from left and right hand side matrices) to solve ϴ6. We choose equations from row 3 and column 1 of matrix **(I)** and **(II).**

(cos(ϴ1)*mx)/(cos(ϴ1)^2 + sin(ϴ1)^2) + (my* sin(ϴ1)) /(cos(ϴ1)^2 + sin(ϴ1)^2) = cos(ϴ6)* sin(ϴ5) ………….(4)

From row 3 and column 2 of matrices (I) and (II) we have the following equation:

(cos(ϴ1)*nx)/(cos(ϴ1)^2 + sin(ϴ1)^2) + (ny*sin(ϴ1))/ (cos(ϴ1)^2 + sin(ϴ1)^2) = -sin(ϴ5) * sin(ϴ6) ………….(5)

Dividing (5) by (4) we get;

ϴ6 = tan_inverse(-(cos(ϴ1) * nx + ny * sin(ϴ1)) / (cos(ϴ1) * mx + my * sin(ϴ1)));

Now we have, T_final = T1 * T2 * T3 * T4 * T5 * T6;

And inverse of T1 matrix which is: T_inv = inv(T1), T_inv5 = inv(T1), T_inv6 = inv(T6);

For simplification we premultiply by T_inv:

Therefore, we get T_inv * T_final = T_inv * T1 * T2 * T3 * T4 * T5 * T6;

T_inv * T_final = T2 * T3 * T4 * T5 * T6;

Now we post multiply by T_inv6 to finally get

T_inv * T_final*T_inv6= T2 * T3 * T4 * T5;

Now we post multiply by T_inv5:

T_inv * T_final*T_inv6*T_inv5 = T2 * T3 * T4 * T5*T_inv5;

Finally we get,

LHS = T_inv * T_final * T_inv6 * T_inv5 ….(III)

RHS = T2 * T3 * T4;...........(IV)

Where LHS is the left hand side and RHS is the right hand side of the equation.

From these resultant matrix of **(III)** and **(IV)** we wisely choose the equation from row 1 column 4 and,

- a2*sin(ϴ2) - a4*cos(ϴ2)*sin(ϴ3) - a4*cos(ϴ3)* sin(ϴ2) = a7*((cos(ϴ1)*oy)/(cos(ϴ1)^2 + sin(ϴ1)^2) - (ox*sin(ϴ1))/(cos(ϴ1)^2 + sin(ϴ1)^2)) + a6*((cos(ϴ6)*((cos(ϴ1)*ny) /(cos(ϴ1)^2 + sin(ϴ1)^2) - (nx*sin(ϴ1))/(cos(ϴ1)^2 + sin(ϴ1)^2))) /(cos(ϴ6)^2 + sin(ϴ6)^2) + (sin(ϴ1)* ((cos(ϴ1)*my)/(cos(ϴ1)^2 + sin(ϴ1)^2) - (mx*sin(ϴ1))/(cos(ϴ1)^2 + sin(ϴ1)^2))) /(cos(ϴ6)^2 + cos(ϴ6)^2)) + (cos(ϴ1)*py)/ (cos(ϴ1)^2 + sin(ϴ1)^2) - (px*sin(ϴ1))/(cos(ϴ1)^2 + sin(ϴ1)^2)

We observe that the right hand side of the above equation is known to us by now therefore we replace it by constant 'K'

Therefore, now we have **a2*sin(ϴ2) + a4*cos(ϴ2) *sin(ϴ3) + a4*cos(ϴ3)*sin(ϴ2) = K……(6)**

Now we choose the equation from row 2 and column 4 of matrix **(III)** and **(IV)** as below:

a2*cos(ϴ2) + a4*cos(ϴ2)*cos(ϴ3) - a4*sin(ϴ2) *sin(ϴ3) = pz + a7*oz + a6 * ((cos(ϴ6)*nz) / (cos(ϴ6)^2 + sin(ϴ6)^2) + (mz*sin(ϴ6))/(cos(ϴ6)^2 + sin(ϴ6)^2))

We observe that the right-hand side of the above equation is known to us by now therefore we replace it by constant 'L'

a2*cos(Θ2) + a4*cos(Θ2)*cos(Θ3) - a4*sin(Θ2) *sin(Θ3) = L……..(7)

By solving (6) and (7) we get,

Θ3 = cos_inverse((sq(K)+sq(L)-sq(a2)-sq(a4))/ (2*a2*a4))

Where sq = square of the given value.

Modifying (6) anf (7) in order to get Θ2 is as under,

a2*sin(Θ2) + a4*sin(Θ2+Θ3) = -K

a2*cos(Θ2) + a4*cos(Θ2+Θ3) = L

We calculate T from some math:

T = (sq(a4) - sq(K) -sq(L)-sq(a2))/(2*a2);

Where sq = square of the specific value.

Θ2 = cos_inverse(((-2 * L * T + square_root(4*sq(L * T) - 4 * (sq(K)+sq(L)) * (sq(T)-sq(K)))) / (2 * (sq(K)+sq(L)))))

Now we consider, row 2 and column 1 of resultant matrix (**III) and (IV)**

sin(Θ4) * (cos(Θ2) * cos(Θ3) - sin(Θ2) * sin(Θ2)) - cos(Θ4)*(cos(Θ2)*sin(Θ3) + cos(Θ3)* sin(Θ2)) = - (cos(Θ5)*((cos(Θ6)*mz) / (cos(Θ6)^2 + sin(Θ6)^2) - (nz*sin(Θ6)) / (cos(Θ6)^2 + sin(Θ6)^2))) / (sin(Θ5)^2 + sin(Θ5)^2) - (oz*s5) / (cos(Θ5)^2 + sin(Θ5)^2)

We observe that the right-hand side of the equation is known to us, we substitute it with a constant 'R'

Now, we have

sin(Θ4) * (cos(Θ2) * cos(Θ3) - sin(Θ2) * sin(Θ2)) - cos(Θ4) * (cos(Θ2) * sin(Θ3) + cos(Θ3) * sin(Θ2)) = R;

Therefore, sin(Θ4 + Θ2 + Θ3) = R;

Θ4 = sin_inverse(R) - Θ2 - Θ3;

Similarly, we compute the values of Θ for finding the negative roots and finally we compare the manipulator's state and verify whether the arm reaches the solution (position), with the different combinations of the Θ values.