Analysis and Application of 2 D.O.F Robotic Arm

Devvrat Arya
Department of Electrical and Electronics
Krishna Institute of Engineering and Technology
Ghaziabad, Uttar Pradesh-201206, India
Devvrat.1321064@kiet.edu

Prof. Alok Kumar Pandey
Department of Electrical and Electronics
Krishna Institute of Engineering and Technology
Ghaziabad, Uttar Pradesh-201206, India
Alok.pandey@kiet.edu

Deva Harsha Bolisetty
Department of Electrical and Electronics
Krishna Institute of Engineering and Technology
Ghaziabad, Uttar Pradesh-201206, India
Deva.1321063@kiet.edu

Abstract: **This paper presents an in-depth examination of 2 degree of freedom robotic arm which moves in a plane. Firstly, a simulation model of this arm is generated on MATLAB. Simulation analysis performed on the model is discussed later for range and accuracy calculations. A hardware model is rendered through it which can perform movements as per requirement. It is refined for real-time operation, moreover it offers high precision for correct positioning on the x-y plane. The robotic arm is controlled using an arduino microcontroller which is interfaced with MATLAB. This paper concludes with some possible applications of 2 D.O.F robotic arm mechanism based on type of end-effector attached to the robotic arm.**

Keywords: 2 D.O.F, arduino, end-effector and MATLAB

I. INTRODUCTION

Robotics is an emerging field nowadays, it is already playing vital role in industries for manufacturing or assembly work. It has profoundly reduced the human effort while it increases production by many folds. In view of consumer industry, it is a technology in developing phase and may exhibit huge applications in future. The robots which will become part of this consumer industry should be multipurpose and human-friendly.

Our project is meant to fulfill this purpose. It has a robotic arm with the replaceable end effector. This kind of robotic arm can place its end effector at any discrete point on a plane. Depending on the application, the end effector and points varies for performing the specific operation by an input of an image feed. Generally, small robotics arm are made using RC servo motors which can be easily controlled using PWM pulse [1] but the model presented here consists stepper motors providing increased precision that is necessary for correct positioning. This motor performance is strongly dependent on driver circuit to control the magnetic flux inside the motor. Arduino Uno-R3 having 8bit microcontroller onboard, controls these drivers. On the other hand, simulation is performed on MATLAB in which both forward kinematics and inverse kinematics is performed. Finally, hardware model is merged with MATLAB's image processing and inverse kinematics solution

II. SIMULATION MODEL

There are two joint variable here, namely rotations of motor (M1) and motor (M2): θ_1 and θ_2 respectively. These parameters will define the position of end point of arm (X, Y) hence it is necessary to derive a relation between these variables and end point position.

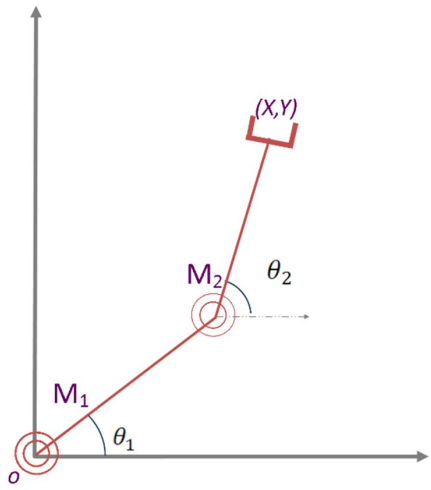

Fig.1 Joint variable and end position

A. *Forward kinematics*:

In forward kinematics analysis, we have both joint variables; the position of every link could be derived from them [2]. In order to perform this analysis the following mathematical derivation was realized

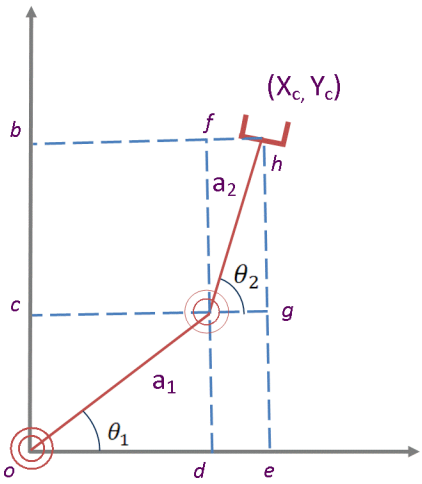

Fig.2 Robotic arm under forward kinematics analysis

In figure a robotic arm (in red) is shown,

The components of arm 1

$$od = a_1 \cos\theta_1 \quad (1)$$

978-1-4673-9080-4/16/$31.00 ©2016 IEEE

$$oc = a_1 \sin\theta_1 \quad (2)$$

The component of arm 2

$$de = a_2 \cos\theta_2 \quad (3)$$

$$bc = a_2 \sin\theta_2 \quad (4)$$

Hence

$$X_c = oe = od + de \quad (5)$$

$$X_c = a_1 \cos\theta_1 + a_2 \cos\theta_2 \quad (6)$$

In case $a_1 = a_2 = a$

$$X_c = a\,(\cos\theta_1 + \cos\theta_2) \quad (7)$$

Similarly,

$$Y_c = a\,(\sin\theta_1 + \sin\theta_2) \quad (8)$$

Using these equations, we need to obtain all the possible postion of end point under a step size of 3.72 ˚. This is only possible by giving every possible input which could be obtained by adding a factor of a step size to motor angle M1 and M2 one by one. Hence both motor angles are get incremented by a factor of 3.72, one after the other to brute force all possible points. Also, the simulation result is affeted by valid motor ranges. To illustrate, consider the following output.

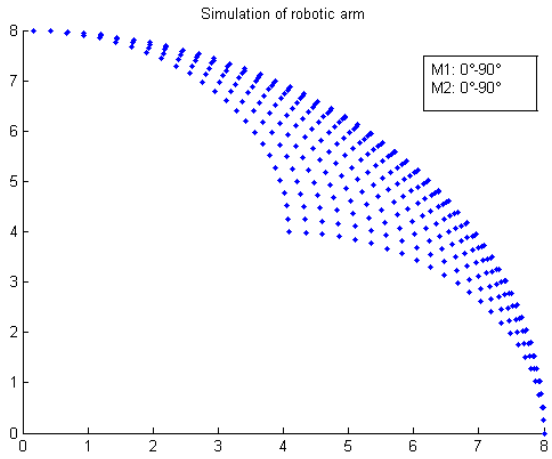

Fig.3 All points of contact at M1 range from 0º to 90º and M2 range from 0o to 90º

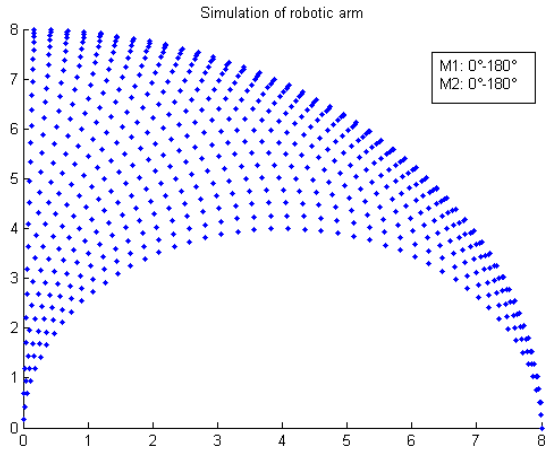

Fig.4 All points of contact at M1 range from- 0º to 180º and M2 range from 0º to 180º

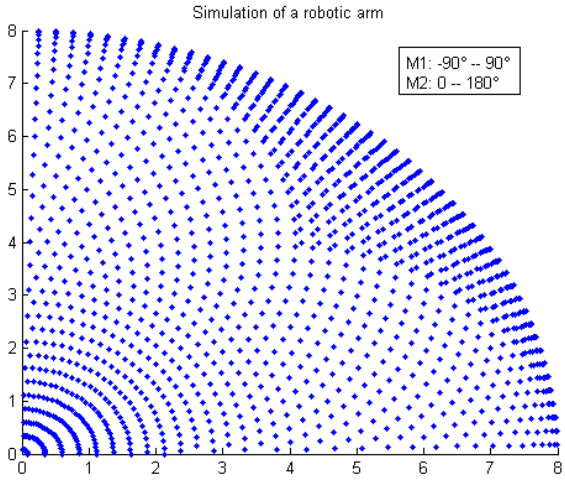

Fig.5 All points of contact at M1 range from -90º to 90º and M2 range from 0º to 180º

Analysis of these figures provides a clear idea about motor angular ranges that affects the number of end-points available in the first quadrant. Another factor that affects simulation result is step size. To illustrate this, consider an area of simulation result

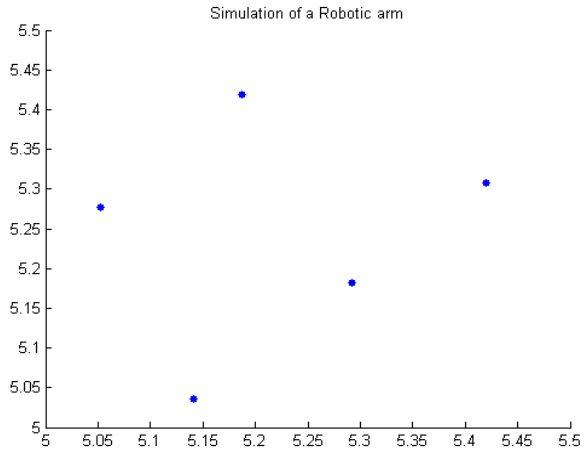

Fig.6 Points of contact at step size 3.72 degrees

Under the step size of 3.72^0 , the number of points of contact are very few in th depicted area. But, on reducing it by factor of 32 gives the step size of 0.117^0. A simulation is performed and given the following result. The points of contact increases exponentially which is neccessery for practical and precise applications. Furhter, it is also necessary to reduce the stepping angle even in hardware model.

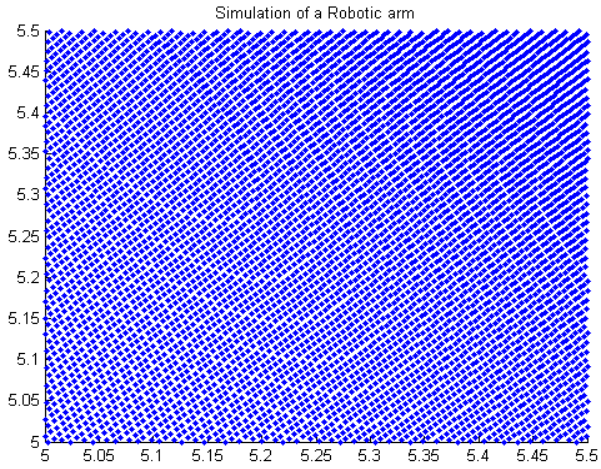

Fig.7 Points of contact at step size 0.117 degrees

This simulation develops necessary understanding about the dynamics but it cannot be used to operate the robotic arm. As forward kinematics equation takes input in terms of angular displacement of the actuator and gives output in the form of coordinate of end effector position. Hence, to operate a robotic arm, inverse kinematics approach is taken into account. In this approach we have taken input in terms of coordinate, process them and got output in terms of angular displacements. Hence, it is programmed like a black box which takes an input of two arrays of x and y coordinates of required points and gives an output of two arrays of angles (from the x-axis) of motor M1 and M2.

B. Inverse kinematics:

Inverse kinematics refers to the use of the kinematics equations of a robot to determine the joint parameters that provide a desired position of the end-effector [3]. As the figure shows, a robotic arm (in red) has an arm's length of a1 and a2 and its end at position (Xc, Yc) . Two triangles Δbcd and Δdef are drawn around the arms.

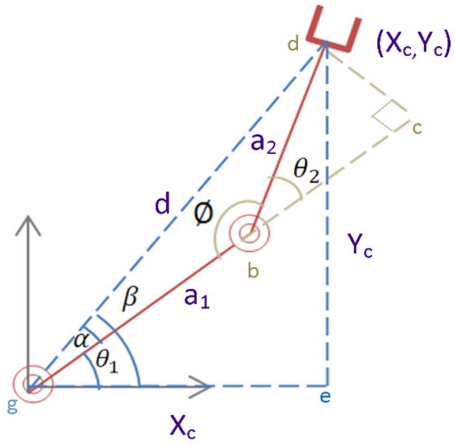

Fig. 8 Robotic arm under inverse kinematics analysis

According to the Pythagoras theorem,

$$d^2 = X_c^2 + Y_c^2 \quad (9)$$

At point b, it can be seen that,

$$\theta_2 = 180° - \emptyset \quad (10)$$

Applying cosine rule in Δdbc

$$d^2 = a_1^2 + a_2^2 - 2a_1a_2 \cos\emptyset \quad (11)$$

Here $a_1 = a_2 = a$

$$d^2 = 2a^2 - 2a^2 \cos\emptyset \quad (12)$$

From (9)

$$X_c^2 + Y_c^2 = 2a^2(1 - \cos\emptyset) \quad (13)$$

On rearranging

$$\cos\emptyset = 1 - \frac{X_c^2 + Y_c^2}{2a^2} \quad (14)$$

Let, $\cos\emptyset = k$

$$k = 1 - \frac{X_c^2 + Y_c^2}{2a^2} \quad (15)$$

$$\emptyset = \text{atan2}(k, \pm\sqrt{1 - k^2}) \quad (16)$$

From (10)

$$\theta_2 = 180° - \text{atan2}(k, \pm\sqrt{1 - k^2}) \quad (17)$$

For θ_1,

$$\theta_1 = \beta - \alpha \quad (18)$$

As $\beta = \text{atan2}(X_c, Y_C)$

And $\alpha = \text{atan2}(a + a\cos\emptyset, a\sin\emptyset)$

Hence,

$$\theta_1 = \text{atan2}(X_c, Y_C) - \text{atan2}(a\cos\emptyset, a\sin\emptyset) \quad (19)$$

After programming these equations in MATLAB, we tested it for an array of coordinates. The output is as follows:

Table 1: Conversion of Cartesian points to joint parameters

Xc	Yc	θ_1	θ_2
6.1	2.6	-10.9317	68.0336
3.2	5.6	23.9839	72.5424
1.3	3.4	6.1409	125.8692
5.3	4.8	15.5226	53.2866
6.8	1.2	-20.3217	60.6594
5.6	4.5	12.6809	52.2069

III. HARDWARE MODEL

A prototype of 2 D.O.F. robotic-arm is prepared which could be operated using simulation results. We have used Bipolar stepper motor as these motors offers high precision rotation in the form of angular steps. These motors revolve through a fixed angle for each pulse applied to the logic sequences. By controlling pulse rate motor's speed can be controlled [4]. Minebea 16pu-m202 bipolar stepper motor came out as suitable selection for this robotic arm as they offer steps of 3.75˚ and rated current of 0.6 amps [5]. There is very limited number of points under the step angle of 3.72 ˚ as depicted in Figure 6. A step size of 1.8 ˚ can be achieved through half stepping using a L298N driver. But, for further decrement of stepping angle, we substituted a microstepper driver DRV8825. This implies, at a maximum microstepping factor of 32, the motor can have a step of 0.117 degrees. This offers high precision, as the density of points increases significantly shown in Figure 7. These two stepper motors are arranged on a single aluminum frame. The shaft of motor M1 is mounted on a fixed wooden board. While the other motor (M2) shaft makes an elbow like joint with other arm. The whole setup is shown here

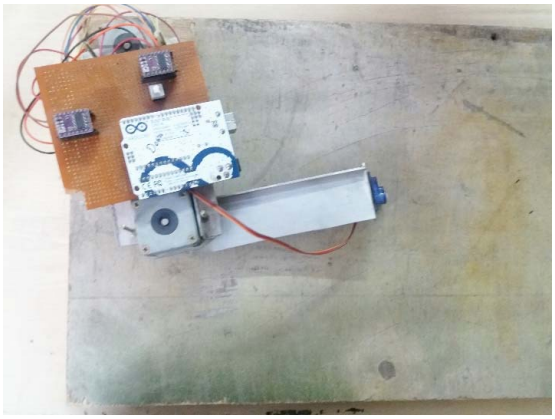

Fig. 9 Hardware model of robotic arm

A. *Motor driver:*

The stepper motor driver circuit serves two major tasks. Firstly, it changes the current and flux direction in the phase Windings. Second, it also enables a short current rise time and fall time for high speed performance. DRV8825 driver controls the motor with a similar technique. Here, if driver is given input of 'n' number of pulses then the the driver will rotate the stepper motor with that 'n' number of steps by varying the current through the phase windings. Another pin present on the driver determines its direction. The driver also has various modes to select which can reduce steps by a factor of 4,8,16 or 32 [6]. DRV8825 driver also has capability for active current limiting which could be set using trimmer potentiometer available on board [7]

B. *Microcontroller board*:

Microcontroller is one of the prerequisite for such robotic arm and Arduino UNO-R3 serves this purpose. It is used for quick prototyping as an arduino package available in MATLAB. The micro-controller board can drive motors as per angle required by producing pulses which is a controlling signal for motor. Now Matlab can instruct the microcontroller where to place the actuators through the serial port of microcontroller. Since any possible application of this project requires a point to be detected automatically rather than inserted manually. For that purpose image processing can be used as it provides us the capability to detect these points effortlessly. As we are already using Matlab for kinematics, It also has an image processing and computer vision toolbox that proved to be very useful. Computer vision toolbox has methods for acquiring, processing, analyzing, and understanding images and converts them into numerical or symbolic information [8]. The actual way for implementing image processing is discussed in application section as it requires a different approach to each application. The processing of data is shown as flowchart here.

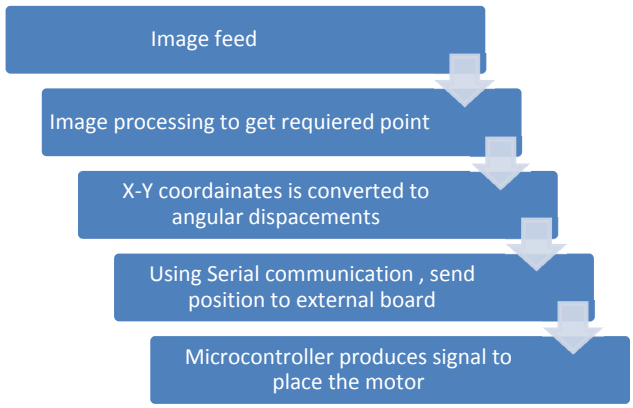

Fig. 10 Flow chart of data acquisition and processing

The whole setup can be shown in terms of block diagram

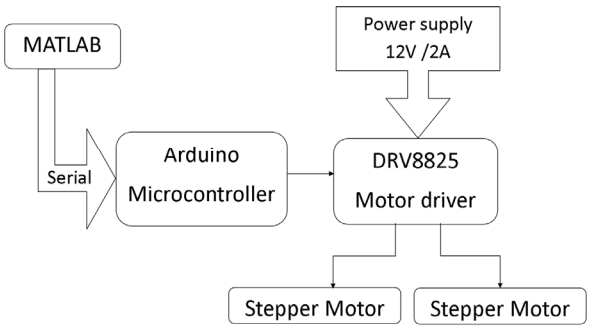

Fig. 11 Block diagram of project

IV. Application

A. *Drawing bot*

Here the end effector is replaced with sketch pen or a marker. The firmware is limited only to single-stroke drawings i.e. it does not have capabilities to pen up and pen down. So, it can make a drawing which has only one starting point and one end point. Firstly an image of drawing is used as image feed. It could be used directly to extract point. Further travelling salesman algorithm is used to join these points by finding the shortest path. Consequently, robotic arm joins these points using a marker and it is observed that drawings are limited for simple figures only. [9]

B. *Braille embosser*:

The braille language is a tactile writing system which has combination of six dots. The robotic arm can be used as braille embosser by replacing the end effector with a Braille printing head. A braille document is converted to a bitmap image which is provides an image feed to the MATLAB. As the arm reaches a desired point, the linear actuator present in braille head emboss dot on the paper.

C. *Autonomous Soldering machine:*

The robotic arm can be modified to a soldering machine by engaging a soldering iron as end-effecter. Gerber is one of the standard formats for PCB design. This Gerber file can converted into PNG or JPEG format to function as image feed. From here we can extract points which are required to solder. The arm places its end effector at required point and soldering tip can solder the component. Further this machine also needs a solder wire moving mechanism. [10]

[1] Patel, H.K., Verma, P. and Ranka, S. (2011), "Design and Development of Co-ordinate based Autonomous Robotic Arm", 2011 *Nirma University International Conference on Engineering*, Ahmedabad, Gujarat, 2011, pp. 1-6

[2] Prof. Jazar, Reza N. (2010), "Theory of Applied Robotics Kinematics, Dynamics and Control", *Springer,* ISBN 978-1-4419-1749-2, pp. 232–234.

[3] Ceccarelli, Marco (2013), "Fundamentals of Mechanics of Robotic Manipulation", *Springer*. ISBN 978-1-4020-2110-7, pp. 121–122.

[4] Scarpino, Matthew (2016), "Motors for Makers: A Guide to Steppers, Servos, and Other Electrical Machines", *Pearson Education*, ISBN : 0134032837, pp. 55–58.

[5] Minebea 16PU-M202 Bipolar Motor Datasheet.

[6] DRV8825 Stepper Motor Controller IC Datasheet.

[7] Description of DRV8825 Stepper Motor Driver www.pololu.com/product/2133

[8] Shapiro, Linda G. and Stockman, George C. (2001), *Computer Vision,* Prentice Hall, pp. 14–26.

[9] A Guide to DIY TRS Drawbot http://makezine.com/projects/trsdrawbot-2/

[10] Building a Laser Cut and Soldering Dobot Robot Arm http://www.instructables.com/id/Build-a-Laser-Cut-and-Soldering-Dobot-arm/