Geometric Shape Drawing Using a 3 Link Planar Manipulator

Anil Kumar, *Student Member, IEEE* and Rahul Kala, *Member, IEEE*
Robotics and Artificial Intelligence Laboratory,
Indian Institute of Information Technology, Allahabd, India
anilkumar6079@gmail.com, rkala001@gmail.com

Citation: A. Kumar and R. Kala, "Geometric shape drawing using a 3 link planar manipulator," 2015 Eighth International Conference on Contemporary Computing (IC3), Noida, 2015, pp. 404-409.
Full text available at: A. Kumar and R. Kala, "Geometric shape drawing using a 3 link planar manipulator," 2015 Eighth International Conference on Contemporary Computing (IC3), Noida, 2015, pp. 404-409.

***Abstract*—Using manipulators to draw artistic figures is a benchmark problem in robotics. In this paper we describe a 3-link planar manipulator which draws artistic shapes or geometric shapes on a canvas. We first take a snapshot of the shape to draw from the camera and process it for finding the outer boundary or edges. The edges are mapped onto the drawing canvas. The edges are converted from Cartesian coordinate space to the configuration space of the manipulator. The joint trajectories so generated are passed to the controller as a trajectory input, which draws the shape using the end-effector or a marker in our case. Our manipulator is able to draw many shapes with high precision and speed.**

***Index Terms*—3 link manipulator, Forward Kinematics, Inverse Kinematics, Artistic Shape.**

I. Introduction

Humans are the best creation of God and are the most intelligent creations having many responsibilities leading to stress and they avoid to do repeated tasks. Fortunately robots are build to do those kinds of tasks and save humans from those dirty works. Undoubtedly many robots are used in industries and for personal assistance i.e. automobiles, spacecraft, health services, education, etc. These robots improve productivity and save million of dollars and human efforts for all. For these reasons robots are widely accepted in human environments. The robots are generally designed for specific tasks and these tasks should be performed by the robot precisely.

Robots are slowly finding their way in the artistic industry, wherein they are being used to draw artistic images much like professional painters. It is widely regarded as a benchmark problem in robotics and involves significant levels of planning and control. Even though it may be argued that pictures may easily be printed, the pictures painted have their own appeal. It is widely believed that the future robots would be able to match professional painters in this business, be it to design creations, to take an idea from the human and complete the design, to assist humans in design, to physically draw the designs, or to create drawings of micro size or overly large drawings.

Our research focuses to develop a 3-link manipulator to draw artistic shapes and geomagnetic shapes i.e. circle, oval, human face, etc. For our problem planar manipulator is able to follow the trajectory on the workspace which resembles the shape to draw. Complicated drawings can be easily decomposed into simple shapes. Drawing such simple shapes gives the basic qualification to the robot, which is also the most challenging problem. The current focus is to draw simple shapes only. Drawing complicated pictures requires complex procedures of finding simple shapes, planning the sequence of edges, giving fine artistic touches, etc., and these issues are of future interest to us[4]. The ability is similar to a child learning drawing, wherein the child first learns how to draw simple shapes, and later learns how to draw more artistic and complex drawings.

The basic kinematics and mathematical formulation of the problem in terms of position, velocity, rotation and workspace decomposition is best discussed in the books by Craig[1] and Fu et al. [2]. The object to be drawn is supplied as an image to the algorithm. By image processing we can get the edge of the shape, which can be converted into a trajectory in the Cartesian space. The problem of inverse kinematics converts any point in the Cartesian space to the joint space, specifying each of the joint angles necessary for the end-effector to reach the desired position. The solution relies on the problem of forward kinematics which converts the joint angle positions into the position of the end-effector in the workspace. A similar transformation in the speeds is given by the Jacobian. For drawing any geometric shape we need to follow the trajectory with very less error, so that more accurate path is followed by the manipulator.

Numerous attempts have been made in the past to make drawings using robotic manipulators. Ata and Myo [6] purposed using the sum of position errors of all intermediate points as the error function, which was optimized using a Genetic Algorithm and pattern search for a 3 Degrees of Freedom (DOF) manipulator. A motivational work is done by Calinon et al. [3] where the robot has 4 DOF arm and an interactive process is done by speech synthesis. For each scenario, the arm manipulator is capable of drawing artistic portraits. Jaafar and Shauri [8] discussed how the robot can perform repeated tasks, improve productivity and reduce cost. Berger and Shamir [5] used a data-driven technique for sketching human face from a photograph to be used in portrait sketching applications. The use of a mobile robot to draw human hand position in air, where the hand position is taken as an input and drawn using a 6 DOF arm, was shown by Osaki and Kanek [9]

This paper is organized as follows. *Section* 2 presents the kinematics and inverse kinematics of the manipulator and workspace for our problem specification. *Section* 3 details the image trajectory generation for the geometric shapes. In *section* 4 we talk about the hardware specifications, whereas

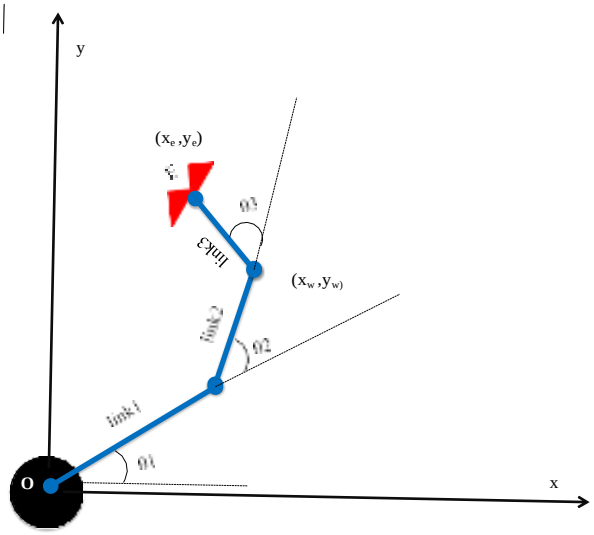

Fig. 1. Three Link Planar Manipulator

in *section* 5 we give the results. Finally *section* 6 gives the conclusion remarks.

II. Kinematics of 3-Link Planar Manipulator

Kinematics describes the mathematical formulation of serially connected manipulators, which is simple to analyze. Our planar manipulator has three links as shown in *fig* 1 which can revolve on the 3 joints and are free to move in a workspace of X-Y plane only. All 3-links (l_1, l_2, l_3) are known and joint angles θ_1, θ_2, θ_3 are variables that we need to find for the end-effector to be at the desired position. So we need to construct forward kinematics transformation matrix ${}^{i-1}_{i}T$ from the base link to the end-effector and correspondingly the inverse kinematics, with the known specifications.

- The `ith` common normal is at 90 degree in between joint axes `i` and `i+1`
- The `ith` link is also the distance between the joints `i` and `i+1`.

A. Forward Kinematics

Forward Kinematics returns the position and orientation of the end-effector, given the values of the joint angles. The length of each link is known. The joint angles are given as an input. We can also thus compute the orientation matrices between all adjacent links from the DH convention. The forward kinematics is represented by Compound Homogeneous Transformation Matrix(CHTM)[15] given by *equation* 1.

$$\underset{init}{final}\,T = {}^{final}_{0}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\cdots{}^{k-1}_{k}T \tag{1}$$

For our problem we use a three-link planar manipulator, therefore 3×3 rotation matrix is used for the figure 1. This gives the general forward kinematics equation given by *equations* 2 – 7.

$$T_1(\theta_1) = \begin{bmatrix} cos(\theta_1) & -sin(\theta_1) & 0 \\ sin(\theta_1) & cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$T_2(\theta_2) = \begin{bmatrix} cos(\theta_2) & -sin(\theta_2) & l_1 \\ sin(\theta_2) & cos(\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$T_3(\theta_3) = \begin{bmatrix} cos(\theta_3) & -sin(\theta_3) & l_2 \\ sin(\theta_3) & cos(\theta_3) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$T_4 = \begin{bmatrix} 1 & 0 & l_3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$${}^{4}_{0}T(\theta_1, \theta_2, \theta_3) = \begin{bmatrix} cos\theta_{123} & -sin\theta_{123} & l_1cos\theta_1 + l_2cos\theta_{12} + l_3cos\theta_{123} \\ sin\theta_{123} & cos\theta_{123} & l_1cos\theta_1 + l_2cos\theta_{12} + l_3cos\theta_{123} \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$\begin{aligned} x &= l_1cos\theta_1 + l_2cos(\theta_1 + \theta_2) + l_3cos(\theta_1 + \theta_2 + \theta_3) \\ y &= l_1sin\theta_1 + l_2sin(\theta_1 + \theta_2) + l_3sin(\theta_1 + \theta_2 + \theta_3) \end{aligned} \tag{7}$$

Here θ_{123} is $\theta_1 + \theta_2 + \theta_3$ and θ_{12} is $\theta_1 + \theta_2$.

In *Fig* 1 our robot arm manipulator is serially connected, while revolute joint axes are in perpendicular direction to the links. Cartesian space origin at O–XY is the base frame and (x_e, y_e) is the location of the end-effector. The end-effector calculation is done by forward kinematics which is given in *equations* 2– 7, taking as inputs the joint angles, where joint angles are in anticlockwise direction from X axis. For our planar manipulator we need to calculate the end-effector orientation, which is given as the sum of the joints angles given by *equation* 8.

$$\varphi_e = \theta_1 + \theta_2 + \theta_3 \tag{8}$$

By using *equations* 2– 8 we can get the relationship between the end-effector position and orientation, and the joints angles $(\theta_1, \theta_2, \theta_3)$. The workspace of the manipulator is defined as all the points which the end-effector can reach. We assume that our workspace is obstacle free and the workspace is planar, so that the manipulator can move freely in the workspace.

B. Inverse Kinematics

Inverse Kinematics takes as inputs the position and orientation of the end-effector and gives the joint angles which make the end-effector reach the given position. This is inverse of the problem of forward kinematics. While forward kinematics always gives a unique solution, the solution to inverse kinematics may not always be unique.

We need to calculate the joint angles$(\theta_1, \theta_2, \theta_3)$ for some known position and orientation of the end-effector. This is given by *equations* 9 – 11.

$$\theta_1 = \tan^{-1}\left(\frac{y_w}{x_w}\right) - \cos^{-1}\left(\frac{x_w^2 + y_w^2 + l_1^2 + l_2^2}{2 \times l_1 \times \sqrt{x_w^2 + y_w^2}}\right) \tag{9}$$

$$\theta_2 = \pi - \cos^{-1}\left(\frac{l_1^2 + l_2^2 - x_w^2 - y_w^2}{2 \times l_1 \times l_2}\right) \tag{10}$$

$$\theta_3 = \varphi_e - (\theta_1 + \theta_2) \tag{11}$$

Given any position of the end-effector, we can calculate all joint angles from *equations* 9 11. Hence we can transform any point in the Cartesian coordinate space to the joint coordinate space.

C. Via Points and workspace

For any artistic shape drawn on paper we have a trajectory having some initial to final point, which represents the shape of the object to be drawn. The procedure of getting this trajectory from the image will be discussed in *sectionIII*. The intermediate points are known as via points. The via points should be connected so that more accurate and smooth shape can be drawn by the manipulator. Each of these via points is in the Cartesian space which can be transformed into the joint space using Inverse Kinematics. The manipulator tries to reach every via point. While doing so the speed of the end-effector is different from the speed of the joint angles. The relation between the Cartesian speed of the end-effector to the speed of the joint angles is given by the Jacobian. Sharp edges in the object trajectory result in sudden changes of speeds in the Cartesian and joint space, which may not be realizable by the manipulator. The sharper be the trajectory, the larger will be the change in speed, and more will be the errors in tracking the shape. Hence the object should preferably not have too sharp edges. Currently we have not optimized the joint and Cartesian speeds and accelerations to efficiently take such sharp turns.

For the trajectory to be traced, it is necessary that all points in the trajectory are within the workspace and free of singularities. A singularity occurs when the determinant of the Jacobian is 0, or a small Cartesian speed produces a very large joint speed. For a 3-link manipulator, singularities are only found at the workspace boundary. It can be ascertained that all points are within the workspace by computing the reachable workspace first, and then scaling down the entire trajectory to ensure that all points lie within the workspace. This restricts our manipulator to move within workspace only.

III. Image Trajectory Generation

In this problem, the input image that the user wants the robot to draw is supplied as an image. The image may be taken using a camera or may be digitally drawn by the user. The image is in the form of an array of pixels, while the drawing manipulator requires a trajectory in the Cartesian space, convertible into a joint space. This section deals with a problem of getting a Cartesian space trajectory.

We first convert the color image into grayscale image where every pixel has a range between 0 to 255. It is assumed that the image consists of an object drawn with a dark ink over a white background, for which case the color information is not needed and the task is trivial. Let the original image be of dimensions $m \times n$. Let the image be given by the function $img : R^2 \rightarrow [0, 255]$, where $img(x, y)$ denotes the grayscale intensity of the pixel at location (x, y), $1 \leq x \leq m$, $1 \leq y \leq n$.

The next issue that comes is resolution. If we try to implement the algorithm on the original image shape, then the trajectory has too many points within a very small gap. A trajectory rich in the number of points is very difficult to trace at respectable speeds. So we need to reduce the resolution of the original image into smaller sized image by using an averaging window. Hence the dimensions get reduced with the same specification. If the original map has a size of $m \times n$ and the resolution factor is a with a window size $a \times a$, then the new image is given by *equation* 12 [13]. The resultant image has a resolution of $m/a \times n/a$

$$img^t(x, y) = \frac{\sum_{i=0}^{a} \sum_{j=1}^{a} img(xa+i, ya+j)}{a^2} \quad (12)$$

Moreover we need to convert *equation* 12 into a binary image. Therefore we need to set some threshold value for the lower resolution image. The binary image is given by *equation* 13. Here Th is the threshold used. Since the original image is drawn with a dark color over a light background, the threshold is easy to set.

$$img^{tt}(x, y) = \begin{cases} 0 & img^t(x, y) \leq Th \\ 1 & img^t(x, y) > Th \end{cases} \quad (13)$$

Fundamentally the binary image contains as black all such points which must be drawn by the end-effector of the robot, which is a pen; and as white all points over which must not be drawn by the robot. The Input image however may be thick and therefore the trajectory path may not be unique and multiple via point will be generated. The current approach is for line drawings and sketching and not for filling colours or paintings. Hence we need edges to draw rather than coloured areas which is the initial representation of the image. We apply Morphological Reconstruction Algorithm (MRA) [12]. MRA depends on the relative arrangement of pixels and uses 4-connected pixel for our 2-D input image shown in fig 2. We apply canny Edge detection technique to get the edges and invert the image.

Fig. 2. Left: input image, Center: After morphological operation, Right: output image

It is assumed that there is a single object having a single boundary. In such a case the Canny edge detector produces a simple edge. The edge is converted into a trajectory by traversing through the edge and noting down all the points. From the *Algorithm*1 we can get the sequence of pixels or the trajectory in the Cartesian coordinate frame.

These points are however in the image coordinate axis system and need to be transferred to the real-world coordinate axis system which was used in the computation of the forward and inverse kinematics. The real-world coordinate axis system is fixed with origin coinciding with the base of the manipulator and XY axis aligned to the axis of the drawing canvas. Inside

Algorithm 1 Image Trajectory Generation

Input 2D image

1) Convert color image into garyscale image.
2) Resize the image and generate new image.
3) Apply morphological operation.
4) Apply Canny Edge Detection for finding edges of the shape.
5) Compute the trajectory

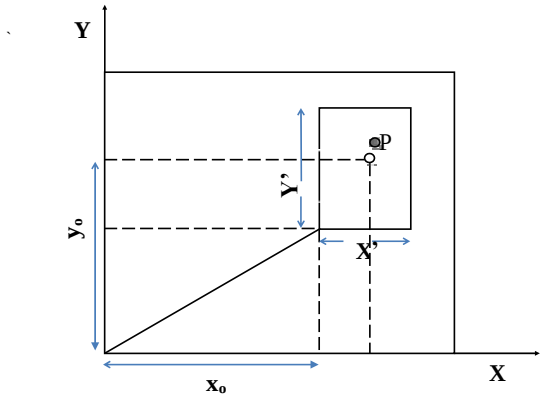

Fig. 3. Mapping of image point into workspace

the reachable workspace of the manipulator, we first define the area within which the image needs to be drawn. This is shown in Fig. 3

We define 3 coordinate frames. The first is the image coordinate frame $X_I - Y_I$, the frame in which the object is supplied as an image by the user. The second is the drawing coordinate frame wherein the image is to be drawn denoted by $X^J - Y^J$. The 3rd coordinate frame is the real world coordinate frame in which the manipulator operates, denoted by $X - Y$. We first map image coordinates into drawing coordinate frame and then the drawing coordinate frame into the real-world coordinate frame.

The size of the input image is $m/a \times n/a$, which needs to be mapped into an area of size $a \times b$. We assume here that any point in this area must be within the workspace i.e. reachable by the end-effctor. Any point $P_I(x_I, y_I)$ in the image coordinate frame hence maps to the point $P^J(x^J, y^J)$ in the drawing coordinate frame with the mapping given by *equation* 14.

$$x^J = \frac{x_I \times a}{m/a}$$
$$y^J = \frac{y_I \times b}{n/a} \tag{14}$$

The scaling factors of a and b can be used to compress or expand the drawing area using *equation* 14. After several experiments we found that if we compress the size of the image on workspace then we get more accurate results i.e. less error; whereas after expanding the workspace area we get more error. Now we need to find the end-effctor position or the point

Fig. 4. Hardware Setup

Fig. 5. Dynamixel Servo Motors used for the experiments

$P(x_e, y_e)$ at the workspace from the base of manipulator. Let the origin of the drawing axis system be at the point (x_o, y_o). The point in the real-world coordinate system is hence given by *equation* 15.

$$x_e = x_o + x^J$$
$$y_e = y_o + y^J \tag{15}$$

IV. Hardware Specifications

All the experiments are performed on a robotic 3 link serial connected manipulator. The system arrangement is shown in *Fig* 4. We use the actuator as three servo motors Dynamixel AX-12+[7]. The servo motors are connected to the computer using a USB2Dynamixel serial to USB converter. The main functionality of USB2Dynamixel is to transform the serial port input into USB port input so that we can easily connect it with a PC or Laptop. However it has CM-2, CM-2+ and CM-5 controller exclusively. The motors used for the experiments are shown in Figure 5 Detailed information about the hardware can be found at [14].

The computer captures the image to be drawn, processes it, generates the trajectory, maps the trajectory to the workspace and generates the control signals. These control signals are transferred independently to the motors using the serial to USB converter. The motors get the signals and perform the actuation, which makes the manipulator trace the trajectory. The manipulator is fitted with a pen as the end-effector. The motion of the manipulator draws the desired shape on the canvas.

V. Experimental Results

For our all experiments we use servo moter Dynamixel and USB2Dynamixel. We connect USB2Dynamixel to a Laptop with configuration (Intel core i3 processor, 6 GB RAM, USB Port 3.0). Required power supply is given to all the motors and USB2Dynamixel (9 Volt approx). The physical hardware setup is shown in Fig. 7. The figure shows a 3-link manipulator mounted at the corner of the drawing canvas. A red coloured pen is used as the end-effector.

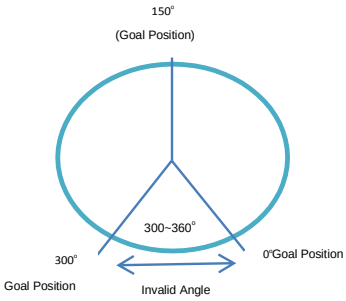

Fig. 6. Internal structure of Dynamixel 12+

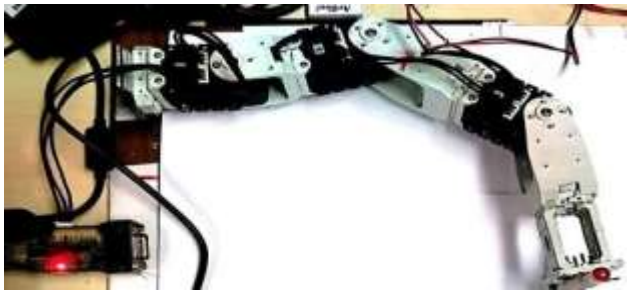

Fig. 7. Hardware configuration setup

Fig. 8. input circle image at Top and resulted image Draw by manipulator at run time at bottom.

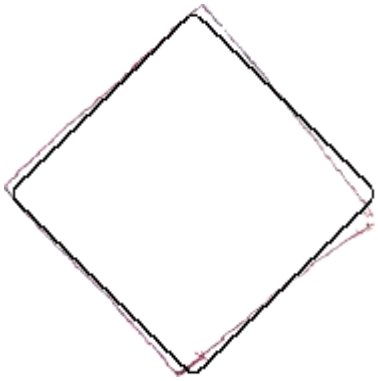

Fig. 9. input rectangle image at Top and resulted image Draw by manipulator at run time at bottom.

Results of the real time experiments of our input shape trajectory drawing manipulator are shown in Figures 8, 9, 10, 11. The original trajectory is shown in black ink while the trajectory drawn by the manipulator is shown in red ink. The two drawings are re-scaled and shown over each other so as to indicate the error in drawing the object.

In the first test scenario of our manipulator, we gave the image of a circle as trajectory to the 3 link manipulator and the result image is shown in Fig. 8. A circle forms a simple shape with a constant curvature and requires the manipulator to show small changes in joint positions. The experiment is very good in judging continuous tracking ability of the manipulator. It can be seen that the circle ends and starts at the same position, which means nearly zero error accumulated in the drawing process. We can see that the actual image and the drawn image both have very small dissimilarities.

The second test scenario of our input trajectory is a rectangular image, which has some sharp corner point turn for the manipulator. The results are shown in Fig 9. In this test case the manipulator followed the straight line trajectory more accurately whereas the corner area has some errors. It is easiest for the manipulator to take smooth trajectories which translates to smooth velocity changes in the joint space. A sharp change ideally cannot be traced unless the manipulator very briefly stops at the corner. Small errors at the corners are hence expected.

In the third test scenario of our manipulator, the input trajectory is a pentagon which has more sharp turns and straight lines within the trajectory, and its real time the result is shown in Fig 10. Again we can visualize that both shapes have some dissimilarities which is largely caused by the corners. As the number of sides of the polygon are increased, the polygon tends towards a circle, and smoother are the corner transitions making it easier for the manipulator. However more corners can imply a higher errors due to transitions at each corner.

In the forth test scenario for our manipulator, the input trajectory is a random geometric shape, for which the result is shown in Fig 11. This image is introduced to make the corners very sharp and not easily smoothable, and hence the expectation is a large error. The manipulator faces problem of making sudden transitions at the corners. Since the corners are very steep, the error gets accumulated by the time the motors are able to make the transition. Thus the error is very large, which can be seen from the results.

VI. Conclusion

Using robots for creating artistic images is a very exciting problem which calls for challenges in image processing, trajectory generation and control. In this paper we showed a mechanism to draw simple shapes on a drawing canvas. The shapes were supplied as images to the algorithm, which were

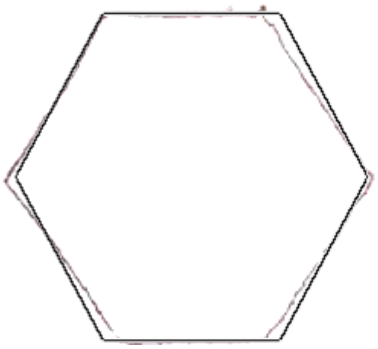

Fig. 10. input pentagon image at Top and resulted image Draw by manipulator at run time at bottom.

Fig. 11. input random image at Top and resulted image Draw by manipulator at run time at bottom.

processed by resolution reduction, edge detection, trajectory generation, solving inverse kinematics and finally controlling the 3-links of the manipulator. Results showed that the manipulator could draw all the regular shapes with a great degree of precision and accuracy. The drawn shapes nearly represented the original shapes given as an image input.

We need to maintain the closeness of the input trajectory and the trajectory tracked by the manipulator's end effector. There are a number of factors which can produce errors in tracking. A major source of error is the hardware itself. Every motor has a resolution of operation and the error can only be ascertained within that resolution. Dynamixel Ax 12+ servo motors used for experiments has a resolution of 0.35 degrees[10]. Errors smaller than this cannot be eliminated. Higher configuration servo motors may have more resolution to resolve this error. Further, our manipulator has lesser number of degrees of freedom and a lesser total outreach which limits the complexity of shapes that can be drawn. We have not currently accounted for feedback control which can significantly reduce the errors. Trajectory speed assignments and optimization is another important task. All this shall be looked into in the future.

The work is a very positive step towards using robots for sketching complicated drawings. The ability to sketch simple shapes gives the most important ability to the robot in pursuit of drawing complex images, which was demonstrated in the work.

ACKNOWLEDGMENT

The authors would like to thank Ajay Kumar and Rajeev Gupta from IIIT Allahabad, India for their invaluable help during the preparation of this manuscript.

REFERENCES

[1] Craig, John J. Introduction to robotics: mechanics and control. Vol. 3. Upper Saddle River: Pearson Prentice Hall, 2005.
[2] Fu, King Sun, Ralph Gonzalez, and CS George Lee. Robotics: Control, Sensing, Visison and Intelligence, Tata McGraw-Hill Education, 1987.
[3] Calinon, S.; Epiney, J.; Billard, A., "A humanoid robot drawing human portraits," Humanoid Robots, 2005 5th IEEE-RAS International Conference on , vol., no., pp.161,166, 5-5 Dec. 2005 doi: 10.1109/ICHR.2005.1573562
[4] Anupam Shukla, Ritu Tiwari, Rahul Kala. "Real life applications of soft computing". CRC Press(2010):465-485.
[5] Berger, Itamar, et al. "Style and abstraction in portrait sketching." ACM Transactions on Graphics (TOG) 32.4 (2013): 55.
[6] Ata, Atef A., and Thi Rein Myo. "Optimal point-to-point trajectory tracking of redundant manipulators using generalized pattern search." arXiv preprint cs/0601063 (2006).
[7] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pt. I, pp. 379–423, 1948;pt. II, pp. 623–656, 1948.
[8] Jaafar, J.; Shauri, R.L.A., "Three-fingered robot hand for assembly works," System Engineering and Technology (ICSET), 2013 IEEE 3rd International Conference on , vol., no., pp.237,241, 19-20 Aug. 2013 doi: 10.1109/ICSEngT.2013.6650177
[9] Osaki, A.; Kaneko, T.; Miwa, Y., "Embodied navigation for mobile robot by using direct 3D drawing in the air," Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on , vol., no., pp.671,676, 1-3 Aug. 2008 doi: 10.1109/ROMAN.2008.4600744
[10] Dynamixel 12+ online `http://support.robotis.com/en/product/dynamixel/dxl_ax_main.htm`
[11] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.
[12] Soille, P., Morphological Image Analysis: Principles and Applications, Springer-Verlag, 1999, pp. 173-174.
[13] Shukla, Anupam, Ritu Tiwari, and Rahul Kala. "Towards hybrid and adaptive computing." SCI 307 (2010): 187-207.
[14] USB2Dynamixel online `http://support.robotis.com/en/product/auxdevice/interface_main.htm`
[15] Miyashita, Naoko, Masashi Kishikawa, and Masaki Yamakita. "3D motion control of 2 links (5 DOF) underactuated manipulator named AcroBOX." American Control Conference, 2006. IEEE, 2006.