Drawing using the Scorbot-ER VII Manipulator Arm

Luke Cole
cole@lc.homedns.org

Adam Ferenc Nagy-Sochacki
u2546772@anu.edu.au

Jonathan Symonds
u3970199@anu.edu.au

October 29, 2007

Abstract

This report discusses how manipulator kinematics can be used to draw an illustration on a paper, using a pen mounted in the end-effector of an Scorbot-ER VII manipulator arm. The Scorbot has 5 degrees of freedom. The 5^{th} joint is for wrist rotation, so only 4 degrees of freedom were used to draw. The robots inverse kinematics were first developed along with a program to produce the trajectory for the joint space in order to draw the illustration. The trajectory for the joint space were then tested in a real-time simulation, which was highly successful. Finally the program was tested in the real world, using the real robot. As expected the results differed due to the performance of the robot itself, however after 8 attempts, most of the drawings produced by the robot arm were visually clear and as accurate as the robot could produce.

1 Manipulator Kinematics

The complete forward and inverse Kinematics for the Scorbot-ER VII [1] manipulator arm is some what complex, so this report will only discuss the required "Simplified" kinematics.

1.1 Simplified Manipulator Kinematics

The simplified manipulator kinematics represents the kinematics relation of the manipulator arm in a simplified drawing orientation. In this instance the pen is kept perpendicular to the drawing surface, this results in a simplified model of the manipulator with a straightforward solution.

1.1.1 Forward kinematics

Figure 1 shows the assigned reference frames. The Denavitt-Hartenberg parameters can easily be determined and are tabulated in Table 1. Given these parameters the appropriate transformation matrices can be determined upon application of Equation 1.

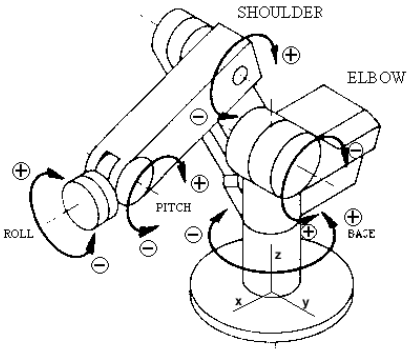

Figure 1: Manipulator Arm

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	d_1	θ_1
2	-90	a_1	d_2	θ_2
3	0	a_2	0	θ_3
4	0	a_3	0	θ_4
5	0	a_4	0	0

Table 1: DH Parameters for Simplified Kinematics

$$ {}^{i-1}_{i}T = \begin{pmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1} $$

Using Equation 1 we get the following transformations:

$$ {}^{0}_{1}T = \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2} $$

$$ {}^{1}_{2}T = \begin{pmatrix} c_2 & -s_2 & 0 & a_1 \\ 0 & 0 & 1 & d_2 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3} $$

$$ {}^{2}_{3}T = \begin{pmatrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4} $$

$$ {}^{3}_{4}T = \begin{pmatrix} 1 & 0 & 0 & a_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5} $$

Multiplying these matrices to form a transformation matrix from the base frame to the final tool is then completed as follows:

$$ {}^{2}_{4}T = {}^{2}_{3}T\,{}^{3}_{4}T \tag{6} $$

$$ = \begin{pmatrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & a_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7} $$

$$ = \begin{pmatrix} c_3 & -s_3 & 0 & a_3c_3 + a_2 \\ s_3 & c_3 & 0 & a_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{8} $$

$$ {}^{1}_{4}T = {}^{1}_{2}T\,{}^{2}_{4}T \tag{9} $$

$$ = \begin{pmatrix} c_2 & -s_2 & 0 & a_1 \\ 0 & 0 & 1 & d_2 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_3 & -s_3 & 0 & a_3c_3 + a_2 \\ s_3 & c_3 & 0 & a_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{10} $$

$$ = \begin{pmatrix} c_{23} & -s_{23} & 0 & a_1 + a_2c_2 + a_3c_{23} \\ 0 & 0 & 1 & d_2 \\ -s_{23} & -c_{23} & 0 & -a_2s_2 + a_3s_{23} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{11} $$

Finally,

$$ {}^{0}_{4}T = {}^{0}_{1}T\,{}^{1}_{4}T \tag{12} $$

$$ = \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_{23} & -s_{23} & 0 & a_1 + a_2c_2 + a_3c_{23} \\ 0 & 0 & 1 & d_2 \\ -s_{23} & -c_{23} & 0 & -a_2s_2 + a_3s_{23} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{13} $$

$$ = \begin{pmatrix} c_1c_{23} & -c_1s_{23} & -s_1 & c_1(a_1 + a_2c_2 + a_3c_{23}) - d_2s_1 \\ s_1c_{23} & -s_1s_{23} & c_1 & s_c(a_1 + a_2c_2 + a_3c_{23}) + d_2c_1 \\ -s_{23} & -c_{23} & 0 & -a_2s_2 + a_3s_{23} + d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{14} $$

1.1.2 Inverse kinematics

By placing the ${}^1_4T$ to the right hand side of the equation we can isolate θ_1 to simplify its calculation as follows:

$$\left[{}^0_1T\right]^{-1}[P] = \left[{}^1_4T\right] \tag{15}$$

$$[P] = \begin{pmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{16}$$

The inverse of a transformation matrix can be calculated using the following matrix

$$[T]^{-1} = \left(\begin{array}{c|c} R^T & -R^T P \\ \hline 0 & 1 \end{array}\right) \tag{17}$$

Applying this to ${}^0_1T$ yields the following result

$$\left[{}^0_1T\right]^{-1} = \begin{pmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & -d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{18}$$

Therefore,

$$\begin{pmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & -d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} c_{23} & -s_{23} & 0 & a_1 + a_2c_2 + a_3c_{23} \\ 0 & 0 & 1 & d_2 \\ -s_{23} & -c_{23} & 0 & -a_2s_2 + a_3s_{23} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{19}$$

In order to determine θ_1 element (2,4) in the matrix can be used.

$$d_2 = -\sin(\theta_1)\, p_x + \cos(\theta_1)\, p_y \tag{20}$$

Letting,

$$p_x = \rho\cos(\phi) \tag{21}$$

$$p_y = \rho\sin(\phi) \tag{22}$$

where,

$$\rho = \sqrt{p_x^2 + p_y^2} \tag{23}$$

$$\phi = Atan2(p_y, p_x) \tag{24}$$

Now,

$$\frac{d_2}{\rho} = -\sin(\theta_1)\cos(\phi) + \cos(\theta_1)\sin(\phi) \tag{25}$$

Therefore,

$$\sin(\phi - \theta_1) = \frac{d_2}{\rho} \tag{26}$$

$$\cos(\phi - \theta_1) = \pm\sqrt{1 - \frac{d_2^2}{\rho^2}} \tag{27}$$

$$\phi - \theta_1 = Atan2\left(\frac{d_2}{\rho}, \pm\sqrt{1 - \frac{d_2^2}{\rho^2}}\right) \tag{28}$$

This leaves the solution for θ_1 as shown in

$$\theta_1 = Atan2(p_y, p_x) - Atan2\left(d_2, \pm\sqrt{p_x^2 + p_y^2 - d_2^2}\right) \tag{29}$$

Now in order to determine joint angle three we must evaluate the elements from $(1,4)$ and $(3,4)$. First,

$$c_1 p_x + s_1 p_y = a_3 c_{23} + a_2 c_2 + a_1 \tag{30}$$

$$p_z - d_1 = -a_3 s_{23} - a_2 s_2 \tag{31}$$

Now by squaring equations 30 and 31 followed by addition θ_3 can be determined.

$$\begin{aligned}
(c_1 p_x + s_1 p_y - a_1)^2 + (d_1 - p_z)^2 &= (a_3 c_{23} + a_2 c_2)^2 + (a_3 s_{23} + a_2 s_2)^2 && (32)\\
&= a_2^2 + a_3^2 + 2a_2 a_3 (c_2 c_{23} + s_2 s_{23}) && (33)\\
&= a_2^2 + a_3^2 + 2a_2 a_3 c_3 && (34)
\end{aligned}$$

This leaves Equation 35 for θ_3

$$\theta_3 = \pm \arccos\left(\frac{(c_1 p_x + s_1 p_y - a_1)^2 + (d_1 - p_z)^2 - a_2^2 - a_3^2}{2a_2 a_3}\right) \tag{35}$$

Now by shifting the matrix ${}^1_2T$ onto the left side also, another equation can be determined that will allow us to determine θ_2.

$$\left[{}^1_2T\right]^{-1}\left[{}^0_1T\right]^{-1}[P] = \left[{}^2_4T\right] \tag{36}$$

$$\left[{}^1_2T\right]^{-1} = \begin{pmatrix} c_2 & 0 & -s_2 & -a_1 c_2 \\ -s_2 & 0 & -c_2 & a_1 s_2 \\ 0 & 1 & 0 & -d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{37}$$

$$\left[{}^{1}_{2}T\right]^{-1}\left[{}^{0}_{1}T\right]^{-1} = \begin{pmatrix} c_2 & 0 & -s_2 & -a_1c_2 \\ -s_2 & 0 & -c_2 & a_1s_2 \\ 0 & 1 & 0 & -d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & -d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{38}$$

$$\left[{}^{1}_{2}T\right]^{-1}\left[{}^{0}_{1}T\right]^{-1} = \begin{pmatrix} c_1c_2 & s_1c_2 & -s_2 & s_2d_1 - a_1c_2 \\ -c_1s_2 & -s_1s_2 & -c_2 & c_2d_1 + a_1s_2 \\ -s_1 & c_1 & 0 & -d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{39}$$

$$\begin{pmatrix} c_1c_2 & s_1c_2 & -s_2 & s_2d_1 - a_1c_2 \\ -c_1s_2 & -s_1s_2 & -c_2 & c_2d_1 + a_1s_2 \\ -s_1 & c_1 & 0 & -d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} c_3 & -s_3 & 0 & a_3c_3 + a_2 \\ s_3 & c_3 & 0 & a_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{40}$$

Now by evaluating elements $(1, 4)$ of Equation 40 we are left with the following formula.

$$c_1c_2p_x + s_1c_2p_y - s_2p_z + d_1s_2 - c_2a_1 = a_3c_3 + a_2 \tag{41}$$

$$c_1\left(c_1p_x + s_1p_y - a_1\right) + s_2\left(d_1 - p_z\right) = a_3c_3 + a_2 \tag{42}$$

Substituting $A = (c_1p_x + s_1p_y - a_1)$, $B = (d_1 - p_z)$ and $C = a_3c_3 + a_2$ it is easy to solve for θ_2 via reduction to a polynomial.

$$c_2A + s_2B = C \tag{43}$$

where,

$$c_2 = \frac{1 - U^2}{1 + U^2} \tag{44}$$

$$s_2 = \frac{2U}{1 + U^2} \tag{45}$$

Substituting Equations 44 and 45 into Equation 43 and rearranging Equation 46 is obtained.

$$\left(C + A\right)U^2 - 2UB + \left(C - A\right) \tag{46}$$

This can then be solved via the quadratic formula rendering

$$U = \frac{B \pm \sqrt{B^2 + A^2 - C^2}}{A + C} \tag{47}$$

where,

$$\theta_2 = 2\arctan\left(U\right) \tag{48}$$

Now in order place the end of the pen at the desired position we must offset the point P these equations would obtain in the z direction by a_4. The angle of 4^{th} joint, θ_4, can also be simply determined based on θ_2 and θ_3.

$$p_z = p'_z - a_4 \tag{49}$$

$$\theta_4 = 90 - \theta_2 - \theta_3 \tag{50}$$

Substituting the z offset in Equation 49 we obtain the following set of formulas for the inverse kinematics of the Scorbot.

$$\theta_1 = Atan2\left(p_y, p_x\right) - Atan2\left(d_2, \pm\sqrt{p_x^2 + p_y^2 - d_2^2}\right) \tag{51}$$

$$\theta_3 = \pm\arccos\left(\frac{\left(c_1 p_x + s_1 p_y - a_1\right)^2 + \left(d_1 - p'_z + a_4\right)^2 - a_2^2 - a_3^2}{2a_2 a_3}\right) \tag{52}$$

$$\theta_2 = 2\arctan\left(\frac{\left(d_1 - p'_z + a_4\right) \pm \sqrt{\left(d_1 - p'_z + a_4\right)^2 + \left(c_1 p_x + s_1 p_y - a_1\right)^2 - \left(a_3 c_3 + a_2\right)^2}}{\left(c_1 p_x + s_1 p_y - a_1\right) + \left(a_3 c_3 + a_2\right)}\right) \tag{53}$$

$$\theta_4 = 90 - \theta_2 - \theta_3 \tag{54}$$

Given the results of the equations above. The appropriate number of encoder clicks can be determined with equations (for robot A):

$$\psi_1 = 23040\left(\frac{2\theta_1}{\pi}\right) \tag{55}$$

$$\psi_2 = 23040\left(\frac{2\theta_2}{\pi}\right) + 11130 \tag{56}$$

$$\psi_3 = 23040\left(\frac{2\theta_3}{\pi}\right) + 8380 \tag{57}$$

$$\psi_4 = 28800\left(\frac{2\theta_4}{\pi}\right) + 682 \tag{58}$$

$$\psi_5 = 9600\left(\frac{2\theta_5}{\pi}\right) + 7658 \tag{59}$$

Although in this case θ_5 was never changed.

2 Trajectory Planning

Trajectory planning involves many considerations such as operational space, joint space and actuator space, let alone working in a dynamic environment. In an attempt to remove these considerations, a drawing program was developed, which allows the robot to draw the required illustration using the same trajectory as the human used to create the illustration. The program is called `drawer`, which is discussed in the Section 3.

In order to maximise the amount of work when drawing a picture on paper, it was decided that it would be most beneficial to break down each move the Scorbot undertook into various processes. This approach allows us to run different drawing, plunging and retracting speeds. Initially this was implemented via breaking down each vector path and determining the appropriate time step to take between position. When implemented on the robot it was found that a significant delay existed between each MOVE command, prohibiting the use of a series of MOVE commands to draw a smooth line. Along with this shortfall the Scorbot will only move between points on a fixed trajectory path, either parabolic or trapezoidal. This greatly restricts the user control over end effector speed when planning a trajectory for the manipulator arm. In order to circumvent these shortfalls while still maximising the available draw time, the vector path of each tool process was run with a single MOVE command and the total process time.

In an attempt to smooth the trajectory path, interpolation was performed. The interpolation procedure populated the distance between points at a specified distance.

3 Software

Figure 2 shows the algorithm used. Here `drawer` is the primary program for the manipulator to know how to draw the desired pattern. The output from `drawer` is then given to `generate_vector_path`, which interpolates the desired drawing and adds timing for each desired position based on the required process. From this data, either `generate_program` can be used to generate Advanced Control Language (ACL) [1] program files for the Scorbot robot or `generate_simulation_data` can be used to generate data for the simulator to interpret. See attached for the source code.

3.1 Drawer

As discussed above and in section 2, `drawer` tracks how a human user draws. The program simply loads up any image you desire to trace, then like any graphics program the user does a series of mouse clicks to trace out the illustration. This allows each vector path (connected object) starting position, trajectory path and ending position to be logged. Currently `drawer` simply outputs a file where each line has the format:

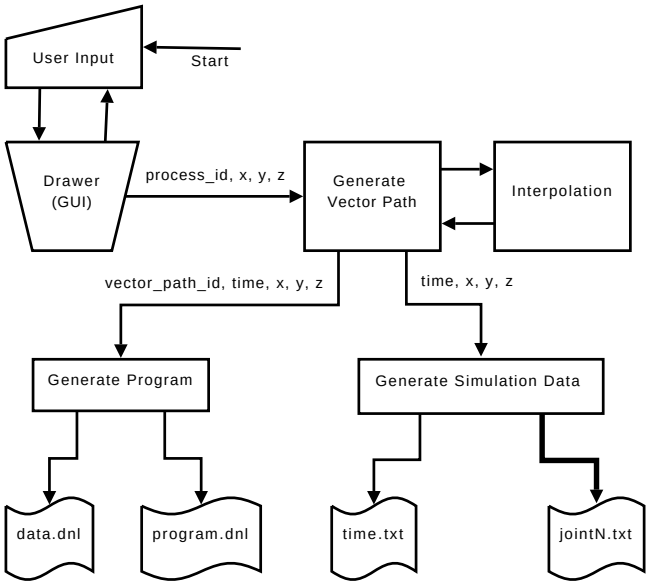

Figure 2: Algorithm

```
PROCESS_ID   X   Y   Z
```

where, `X`, `Y` and `Z` are x_i, y_i and z_i respectively for the i^{th} point drawn by the user. Assuming we are drawing on a uniform flat surface, we can set $z_i = 0$ for $i = 0, 1, ..., N$. Before the drawing process begins, the robot will first move to the desired position, then put pen to the paper (plunging). These process are defined by `PROCESS_ID`s (see Table 2). When moving between connected objects and plunging z_i will clearly be non-zero.

Process Option	Process ID
PROCESS_MOVE	1
PROCESS_PLUNGE	2
PROCESS_DRAW	3

Table 2: Process ID's

3.2 Generate Vector Path

The output from `drawer` is given to `generate_vector_path`, which groups the points into vector paths for each given process. Each vector path is then interpolated. The newly interpolated vector paths are then timestamped, in order to move through the vector path at the desired process speed. Currently

`generate_vector_path` simply outputs a file with that data, where each line has the format:

```
VECTOR_PATH_ID   TIME_STAMP   X   Y   Z
```

where, `X`, `Y` and `Z` are x_i, y_i and z_i respectively for the i^{th} point the arm will move to. `TIME_STAMP` is the time at which the arm must be at this position. Finally, `VECTOR_PATH_ID` is the ID of the group of points that define a vector path.

A routine was implemented to perform interpolation over a vector path. Given a vector path and desired density of points, the routine determines the same vector path with uniformly distributioned points through the path of the vectors.

3.3 Generate Program

With knownledge of the manipulator arm kinematics, and a list of cartesion points to move through and the desired times at which the end effector must move to these points. We have enough information to control the robots joints to perform the given task, in this case drawing an illustration on paper. The routine `generate_program` performs this operation using the data from `generate_vector_path` to simply output two files in the format of the Advanced Control Language (ACL). `data.dnl` defines all the joint angles the arm will move to, and `program.dnl` defines how to move through the joint space.

In detail, `data.dnl` is simply a ACL array defining the joint space for drawing the illustration. `program.dnl` simply uses the ACL `MOVES` command to move from one joint vector to another joint vector in the defined joint space (`data.dnl`). Moving between joint vectors is done using the ACL `TRAPEZE` speed profile, to ensure almost constant speed for each process. `program.dnl` also defines the travel time between joint vectors (last joint vector timestamp minus first joint vector timestamp).

4 Results

Figure 3 shows the image used by `drawer` to trace out the desired outline patten of the well known Transformers [4] autobot symbol, along with a string in the font used by the Transformers movie [2]. Once the input traced out the desired patten the algorithm could then produce the ACL program files which could be simply downloaded to the manipulator arm's computer or given to a simulator.

4.1 Simulation

Before testing the outputted joint space and timing on the two real robots available in a real world, we tested it in a simulation. Since we have the timing for the joint vectors, our simulation will be close as possible to real time. The simulation was developed in Solidworks COSMOSMotion [3] and built upon the

Figure 3: Desired Traced Drawing

simple description from the DH parameters. The simulation takes in a set of 4 files each one describing the joint angle at a point in time and varies the joints accordingly. As seen in Figure 4, a trajectory path is represented and allows quick visual verification of the accuracy of the forwards and inverse kinematics.

From the simulation information the predicted pen tip velocity and acceleration can be obtained. Figure 5 shows the velocity and acceleration of the (pen) tip of the end-effector. The primary known deviation between these simulated results and the actual robot will occur here. Where the velocity profile between each consecutive move will be of a parabolic or trapezoidal shape. The simulation however assumes a linear response. Thus in the real world, the joints will obtain a faster velocity as the total path time is still identical however the acceleration will be much more controlled and progressive. The real robot will eliminate the spikes apparent on these simulation graphs, and allow for a more fluent motion.

4.2 Real Robot, Real Time, Real World

Once the joint space and timing output was found to be correct in the simulation, testing on a real robot in a real world could begin. Figure 4 shows the first set of results using the robot known as *Robot B*. Figure 5 shows the second set of results using the robot known as *Robot A*. Table 3 shows the different environment settings.

Since we were confident the manipulator would perform like the simulation, the only concerns were the performance of the robots (A and B) themselfs, since they are over ten years old and have been abused over the years by students. After seeing the result of the first attempt, it was noted the robot performs better in different drawing locations (drawing area) on the table. So to improve the drawing the arm was re-programmed to draw in areas that appears to reduces the occuring manipulator arm oscillations. The first two drawing attempts also show, that even without interpolation the arm failed to move in straight lines.

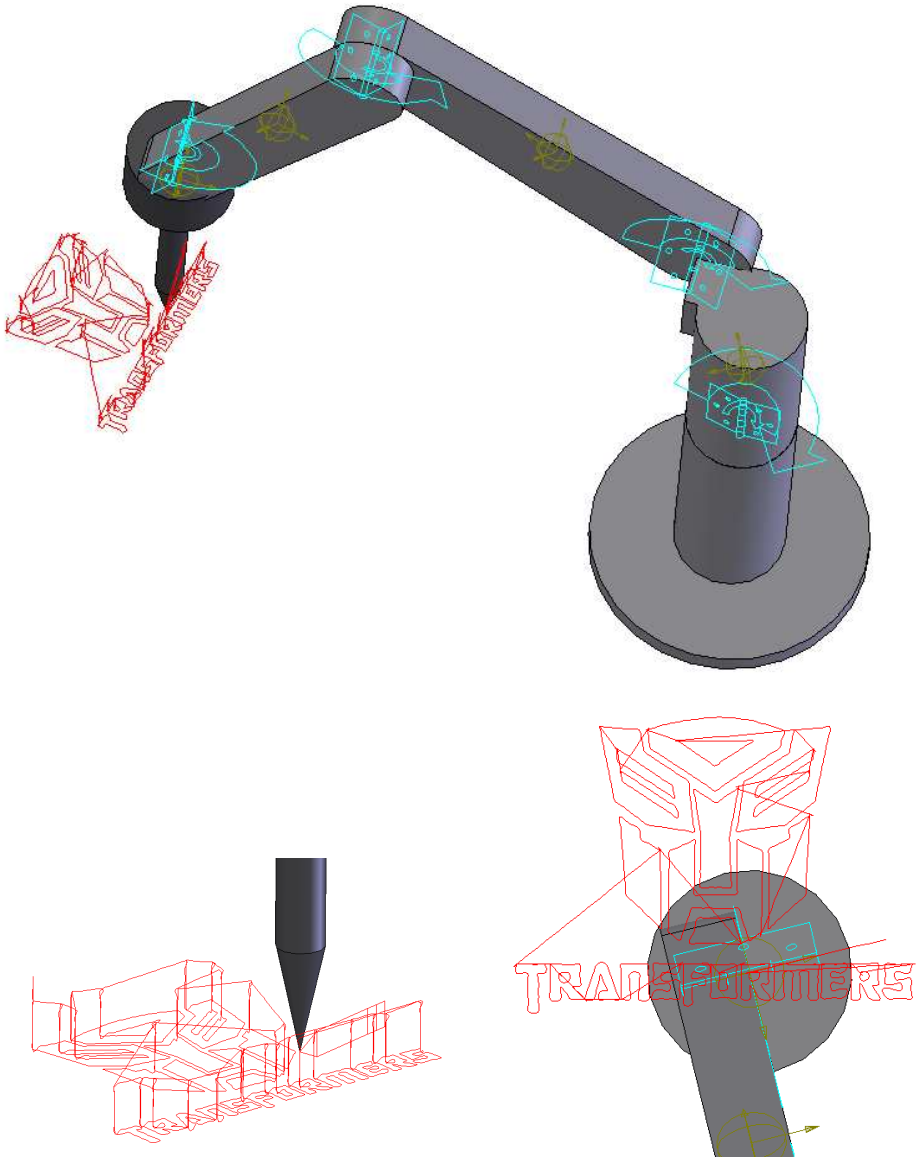

Figure 4: Simulation Screen Shots

Finally in attempt 4, a witnessed and signed attempt was produced which clearly shows a much better illustration of the autobot. Unfortunately it appears, the poor performance of the robot made the text unreadable. The second set of experiments where performed on *Robot A*, since the first three test where done

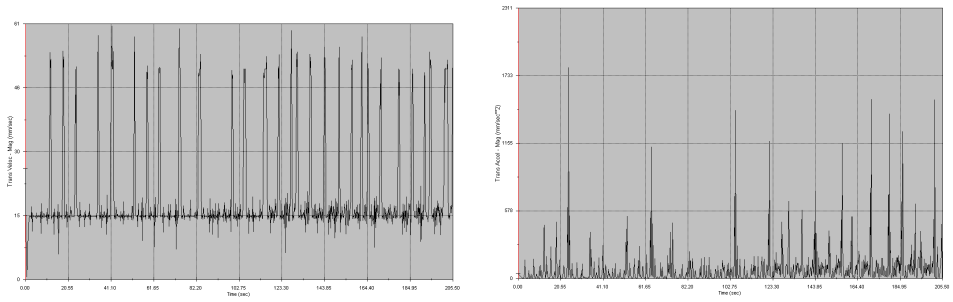

Figure 5: Velocity and Acceleration of End-effector in Simulation

Attempt	Robot	Drawing Area	Interpolation	Text Width	Date/Time
1st	B	$[4000, 100, 0]$	None	None	24/10/07 4:30pm
2nd	B	$[4000, -200, -10]$	None	None	24/10/07 4:40pm
3rd	B	$[4000, -1500, -10]$	Yes	85mm	24/10/07 4:50pm
4th	B	$[4000, -2500, -10]$	3mm	85mm	24/10/07 5pm
5th	A	$[4000, 0, -10]$	Yes	85mm	24/10/07 5:30pm
6th	A	$[3000, 0, 0]$	Yes	85mm	24/10/07 5:40pm
7th	A	$[3000, 0, 0]$	Yes	120mm	24/10/07 5:50pm
8th	A	$[3000, 1000, -10]$	3mm	150mm	26/10/07 2pm

Table 3: Experiment Environment Settings

very quickly, so we forgot to adapt *Robot A*'s end-effector link length to the program, causing the drawing pressure to effect the drawing, let alone the pen life time. For final attempt, a simple drawing test was used to find the best drawing area and then, a witnessed and signed attempt which clearly shows the pen has the correct pressure and produced readable transformer text. Unfortunately by this time, *Robot A* had a uncontrollable twitch, causing (artistic) oscillation.

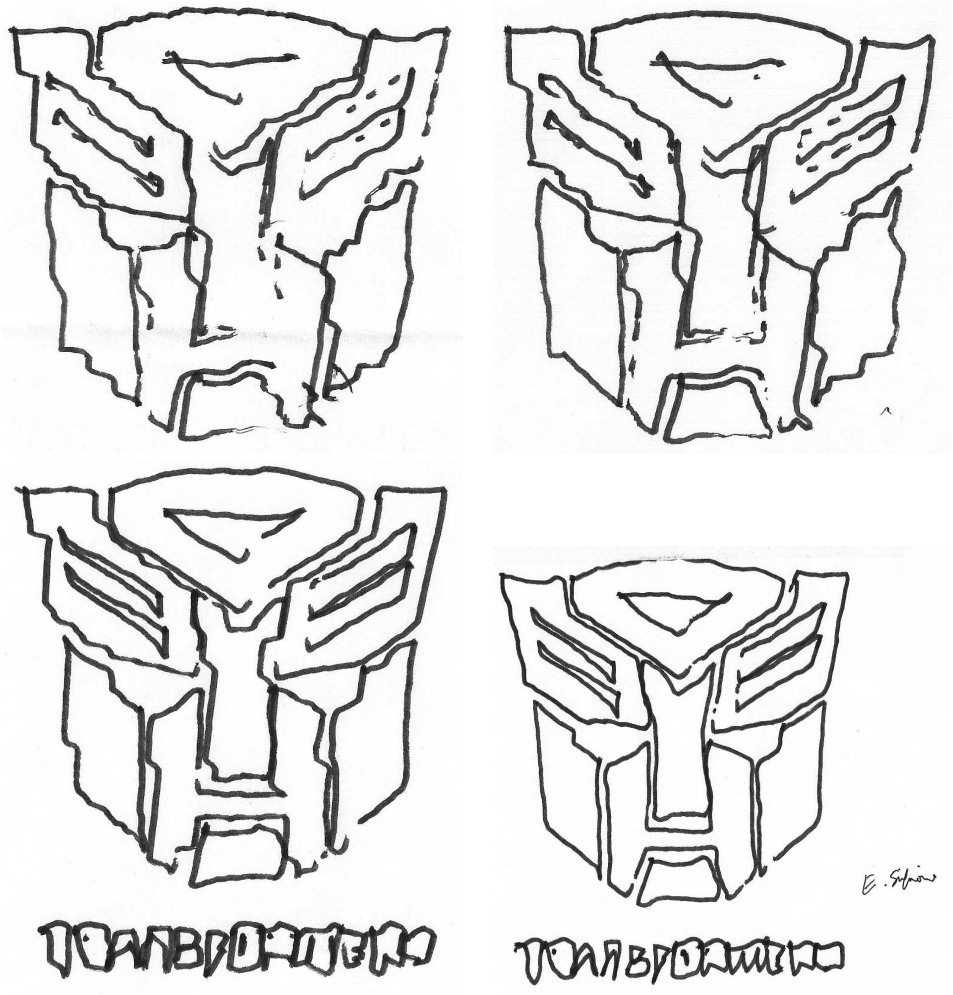

Table 4: First set of Drawing Attempts on *Robot B*, from 1^{st} Attempt (top left) to 4^{th} Attempt (bottom right)

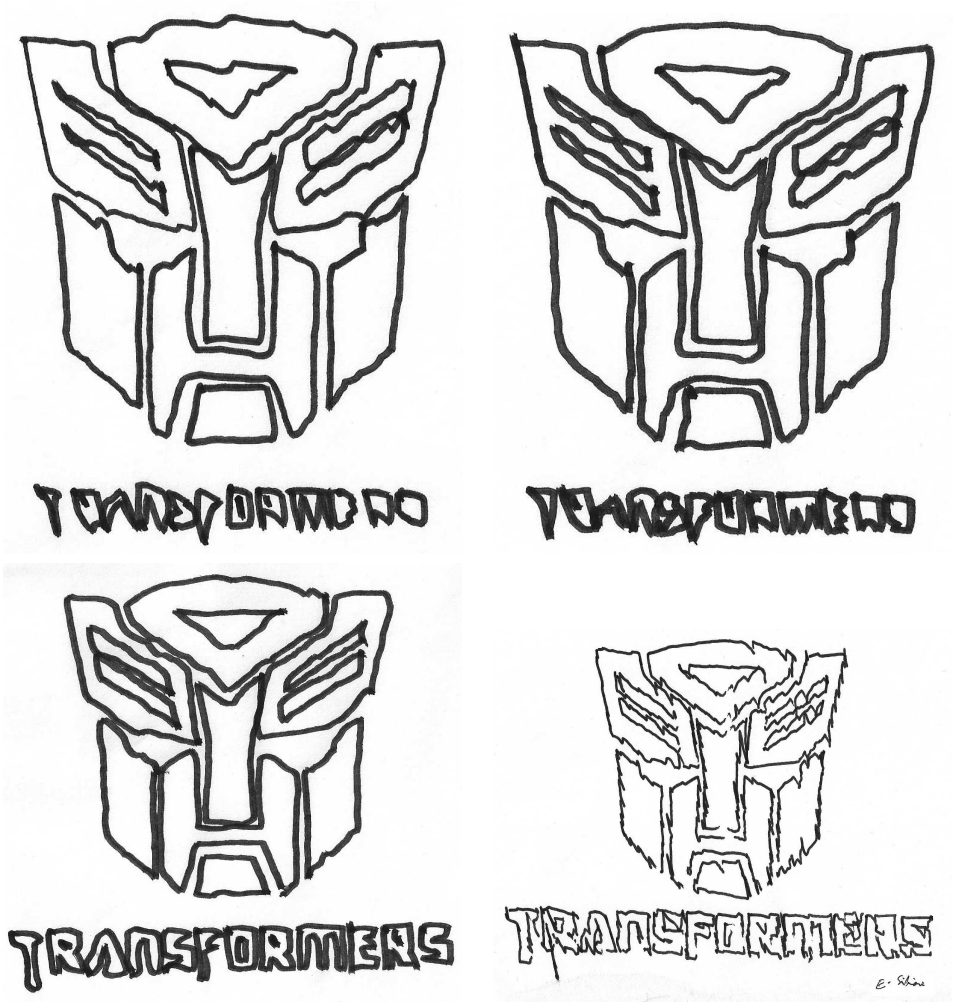

Table 5: Second set of Drawing Attempts on *Robot A* from the 5^{th} Attempt (top left) to 8^{th} Attempt (bottom right)

4.3 Discussion

Drawing on a piece of paper with any basic manipulator arm is not an easy task. Humans have many sensors, such as vision and touch that allow a human to draw with ease, even as a young child. However an manipulator arm, such as the one discussed in this report only have motors to move move the links, and encoders to read how much the each link has truly moved. The resolution of the encoders clearly effect the resolution for the movement of the end effector, in this case a pen drawing on a piece of paper. Even with high resolution encoders, a manipulator arm with gears and/or belts such as the Scorbot experience backlash and wear, which require maintenance for optimal performance. Furthermore, humans generally draw/write via supporting the weight of their hand on the drawing surface, which clearly has its advantages. Even though Scorbot is basically "disabled" in terms of a human drawing ability, we can still expect reasonable accurate illustrations and readable text.

From examining the drawing attempts it can be seen that attempt 4 through to attempt 7 produced reasonable illustrations of the autobot logo's and attempts 6 and 7 produced readable block text (through increased text size). Clearly attempts 4 and 8 are also applying the correct pen pressure, which was found to be 10mm. While attempt 4 appears to be the most accuate, credit must be given to attempt 8, which clearly domonstrates the robots artistic[1] skills.

While the Scorbot has issues itself, causing it to draw with some oscillations, let alone reducing the life time of the pen. The robot was mounted to a very poor table, which would be incapable of providing a very uniform flat surface, which was assumed in the `drawer` settings, discussed in Section 3. Consequently changing the pen pressure somewhat, further reducing the life time of the pen.

5 Conclusion

Clearly the experiments were a complete success. While the drawings are not perfect, they are as good as it gets with such a old, used and abused manipulator robotic arm. The simulation, as excepted was very useful to validate generated trajectories, and further shows the algorithm developed performs as required.

6 Future Work

- Fixed angled pen. So the pen writes at a fixed angle with respect to the table platform.
- Fixed angled pen in the direction of the trajectory, if possible given the fixed 6^{th} joint.

[1] The reality of a real robot in a real world, or the beauty of nature/robots?

- Numerical implementation of the inverse kinematics, rather then a derived approach. eg, Newtons method, non linear programming. For interest rather then drawing a better picture
- Mount different colour pens to the end-effector offset to the centre of wrist rotation. Then angle end-effector somewhat and rotate the wrist to change pen colours.
- Use a camera to provide the feedback to the computer and have a real-time computer vision algorithm determine the trajectory path with (or instead) of Inverse Kinematics.

References

[1] Intelitek (formally Eshed Robotec), 2007. http://www.intelitek.com.

[2] Hasbro. Transformers movie, 2007. http://www.transformersmovie.com.

[3] SolidWorks, 2007. http://www.solidworks.com.

[4] Wikipedia. Transformers series, 2007. http://en.wikipedia.org/wiki/Transformers_series.