ABSTRACT

Trajectory planning refers to the ability of a robot to plan its motions. The robotic arms on the International Space Station (ISS) and the Space Shuttle are currently hand operated by two humans [Rogers 1998]. It is the intention of NASA to develop software modules that automatically calculate a near optimal path trajectory with little human intervention. This project developed modules to perform such a task by use of genetic algorithms. The objective was to place the end effector as close as possible to a predetermined destination point in three-dimensional space. The algorithm developed decreased the distance from the destination as population size, number of generations, and number of processors were increased.
# TABLE OF CONTENTS

Abstract .............................................................................................................. ii

Table of Contents .............................................................................................. iii

List of Figures ...................................................................................................... iv

1. Introduction and Background ........................................................................ 1
   1.1 Genetic Algorithms ...................................................................................... 1
   1.2 Manipulator Trajectory Planning ................................................................. 2
   1.3 Descriptions ................................................................................................. 3
      1.3.1 Description of a Position ................................................................. 3
      1.3.2 Description of an Orientation ......................................................... 4
      1.3.3 Description of a Frame ................................................................. 6
   1.4 Previous Work ............................................................................................. 6
   1.5 Spherical Coordinates ............................................................................... 7

2. Robotic Manipulator Planning Using Genetic Algorithms ............................ 9
   2.1 Path Planning and Generation ................................................................. 9
   2.2 Applying Genetic Algorithms to Manipulators ....................................... 11
   2.3 The SRMS and the Methods ................................................................. 12

3. System Design .............................................................................................. 15
   3.1 Environment ............................................................................................. 15
   3.2 Path Production ......................................................................................... 15
      3.2.1 Calculating the Desired Straight Line Path .................................... 16
      3.2.2 Dividing the Straight Line ............................................................ 17
      3.2.3 Running the Genetic Algorithm .................................................. 18
3.3 File System ................................................................. 19
3.4 Script Files ............................................................. 20
4. Testing and Evaluation ................................................... 20
   4.1 Configuration Fitness Analysis ................................. 22
   4.2 Experimentation of Values ...................................... 23
   4.3 Evaluation of Results .............................................. 32
5. Future Work ............................................................. 31
6. Conclusion ............................................................... 32
Acknowledgement .......................................................... 33
Bibliography and References ............................................ 34
Appendix ................................................................. 35
LIST OF FIGURES

Figure 1.1 Vector relative to frame example ........................................... 4
Figure 1.2 Locating an object in position and orientation ............................. 5
Figure 1.3 Graphical representation of a spherical coordinate system .............. 8
Figure 2.1 Davidor Arm Configuration ..................................................... 10
Figure 2.2 Davidor's illustrative sequence of arm-configurations of an $n$-link structure .11
Figure 2.3 Mating of two paths after crossover ........................................... 12
Figure 2.4 SRMS Dimension and Joint Limits ............................................ 14
Figure 3.1 Directional displacement of a vector ........................................... 17
Figure 3.2 Dividing the Desired Straight Line ............................................ 18
Figure 4.1 Graph of GA for 10 generations .............................................. 24
Figure 4.2 Graph of GA for 50 generations .............................................. 25
Figure 4.3 Graph of GA for 100 generations ............................................ 26
Figure 4.4 Graph of GA for 500 generations ............................................ 27
Figure 4.5 Graph of GA for 1000 generations .......................................... 28
Figure 4.6 Graph of GA for 1500 generations .......................................... 29
Figure 4.7 Plot of x positions versus y positions ....................................... 30
Figure 4.8 Plot of x positions versus z positions ....................................... 31
Figure 4.9 3D Plot of SRMS trials ......................................................... 32
LIST OF TABLES

Table 2.1  SRMS Joint Limits ................................................................. 16

Table 4.1  Data for 10 generations ......................................................... 24

Table 4.1  Data for 50 generations ......................................................... 25

Table 4.1  Data for 100 generations ....................................................... 26

Table 4.1  Data for 500 generations ....................................................... 27

Table 4.1  Data for 1000 generations ..................................................... 28

Table 4.1  Data for 1500 generations ..................................................... 29
1. INTRODUCTION AND BACKGROUND

Robots have come a long way since their conception in tales of science fiction. Today, robots in many forms are used for a vast array of tasks. The types of tasks range anywhere from exploration of places unfit for humans to heavy industrial applications. The motion and control of these robots is handled manually or through programs. Robots can be programmed to follow a set of instructions repeatedly in an industrial setting. To program such robots, or manipulators, a robot programming language serves as an interface between the user and the industrial robot. Robots can sometimes adapt to changing settings with the use of sensors and communications to allow interface with humans or other equipment [Craig 1989]. Machine learning can be used to optimize the path of a manipulator. Genetic Algorithms (GAs) can facilitate learning by the control systems of these manipulators. GAs have been shown to produce near optimal results in many applications, but they do rely on humans to program how they learn [Goldberg 1989].

1.1 Genetic Algorithms

GAs are software procedures which use methods from biological genetics and evolution. Their main purpose is to efficiently search and obtain an optimal, or acceptably near optimal, solution to a particular problem. Like Darwinian evolution, the search proceeds in a survival-of-the-fittest fashion by a gradual manipulation of an initial population until the most desired solution to the problem is found [McCord 2003].

Creating and executing a simple genetic algorithm is fairly easy. A good simple algorithm is composed of three operators: 1) reproduction, 2) crossover, and 3) mutation.
Reproduction is the process by which individual strings are copied according to their function values; biologists refer to these as fitness values. A function value can be thought of as a measure of deservedness. This operator is an artificial version of Darwin’s concept of natural selection [Goldberg 1989].

After reproduction, crossover is performed. This is done by choosing mates at random or according to some heuristic. Each pair of selected mates undergoes crossover. Position(s) are selected along the length of the member where each mate exchanges information. For example, strings $A_1$ and $A_2$ are initially populated as follows:

$$
A_1 = 0 1 1 0 | 1 \\
A_2 = 1 1 0 0 | 0
$$

In this case, the random crossover point is denoted by the $|$ symbol. The resulting crossover yields two new strings that will be part of the new generation:

$$
A_1' = 0 1 1 0 | 0 \\
A_2' = 1 1 0 0 | 1
$$

After crossover, mutation is performed. Mutation makes small changes in a population. This is important because reproduction and crossover can cause valuable genetic material to be lost. The mutation operator protects against such lost by allowing the possible addition of new information. However, to prevent the evolution of the solution from being severely disrupted, mutation rates in genetic algorithms are usually very small, about one in one thousand [Goldberg 1989].
1.2 Manipulator Trajectory Planning

The first issue to consider in planning a path or trajectory of a manipulator is how to specify the particular trajectory through multidimensional space. “Trajectory” refers to the time history of position, velocity, and acceleration for each degree of freedom. Another issue is how trajectories are represented in the planning. The generation of the actual trajectory is crucial [Craig 1989].

The basic goal when working with a manipulator is to move from an initial position to a desired final position. This motion generally involves changing the manipulator’s position and orientation relative to its station. It is necessary to include a specific path description with a sequence of points and intermediate positions between the initial and final points [Craig 1989].

1.3 Description

The description of a manipulator is used to specify attributes of the objects inside a coordinate system. These objects are parts, tools, or even the manipulator itself. Descriptions exist to specify positions and orientations, or frames that contain the positions and orientations.

1.3.1 Description of a Position

After a coordinate system has been established, any point can be located with a 3 by 1 position vector. Vectors must be tagged with information identifying the coordinate system in which they are defined. Each element is usually a numeric distance along a specified axis and the result is the vector from the origin passing through the location.
An example is shown in Fig. 1.1 where the coordinate system \{A\} has labeled axes $X_A$, $Y_A$, $Z_A$, and the resulting vector $^A{\mathbf{P}}$ passing through point B.

![Figure 1.1 Vector relative to frame example.](image)

Individual elements of vector $^A{\mathbf{P}}$ are represented as follows:

\[
^A{\mathbf{P}} = \begin{bmatrix} X_A \\ Y_A \\ Z_A \end{bmatrix}.
\] (1.3)

Other representations can be created using, for example, spherical or cylindrical coordinates [Craig 1989].

1.3.2 Description of an Orientation

Often knowing where a point in space is located is not enough. Its orientation must also be specified. For instance, if the vector $^A{\mathbf{P}}$ in Fig. 1.1 locates a point in between the fingertips of a manipulator’s hand, the location of the hand cannot be completely specified until its orientation is specified. An orientation attaches a coordinate system to a body and gives a description of the coordinate system relative to the reference system [Craig 1989].
To describe this mathematically, unit vectors need to be denoted in the directions of the manipulator’s coordinate system $\mathcal{B}$ as $X_B$, $Y_B$, and $Z_B$. When written in terms of a principal coordinate system $\mathcal{A}$, these unit vectors are called $^A X_B$, $^A Y_B$, and $^A Z_B$ each being a 3 by 1 position matrix can be combined to make a 3 by 3 rotation matrix. Since this resulting matrix would be describing $\mathcal{B}$ relative to $\mathcal{A}$, we can conveniently use the notation $^A B R$:

The scalar elements $r_{ij}$ in Eq. (1.4) are components of a vector and are simply projections onto the reference frame. Hence, each component of $^A B R$ can be written as the dot product of a pair of unit vectors as shown in Eq. (1.5) [Craig 1989].

$$^A B R = \begin{bmatrix} ^A X_B & ^A Y_B & ^A Z_B \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}.$$

(1.4)
\[ a^b_R = \begin{bmatrix} ^aX_b & ^aY_b & ^aZ_b \end{bmatrix} = \begin{bmatrix} X_b \cdot X_A & Y_b \cdot X_A & Z_b \cdot X_A \\ X_b \cdot Y_A & Y_b \cdot Y_A & Z_b \cdot Y_A \\ X_b \cdot Z_A & Y_b \cdot Z_A & Z_b \cdot Z_A \end{bmatrix}. \] (1.5)

### 1.3.3 Description of a Frame

The information needed to fully describe the whereabouts of a manipulator hand is the position and the orientation. The necessity for the position-orientation pair occurs so often in robotics that an entity called a frame is used. The description of a frame is the equivalent of a position vector and a rotation matrix. For example, in Fig. 1.2, the vector AP locates a position and can be thought of as the position vector, and the other three vectors in \{B\} describes its orientation and can be thought of as the rotation matrix [Craig 1989].

### 1.4 Previous Work

This project was not the first to use genetic algorithms for trajectory planning of robot manipulators. Yuval Davidor developed an example of trajectory generation using genetic algorithms. In his research, Davidor used two different crossover methods (usually only one is employed) for plotting the path and the configurations of each link in a manipulator [Davidor 1991].

### 1.5 Spherical Coordinates

Weisstein defines spherical coordinates as a system of curvilinear coordinates that is natural for describing positions on a sphere. Defining \( \theta \) to be the azimuthal angle in the xy-plane from the x-axis with \( 0 \leq \theta \leq 2\pi \), \( \phi \) to be the polar angle from the z-axis with...
\(0 \leq \phi \leq 2\pi\), and \(r\) to be the distance (radius) from the origin. The values for \(\theta\), \(\phi\), and \(r\), can be related to Cartesian coordinates in the following manner:

\[
\begin{align*}
    r &= \sqrt{x^2 + y^2 + z^2} \\
    \theta &= \tan^{-1}\left(\frac{y}{x}\right) \\
    \phi &= \sin^{-1}\left(\frac{\sqrt{x^2 + y^2}}{r}\right) = \cos^{-1}\left(\frac{z}{r}\right)
\end{align*}
\]

where \(r \in [0, \infty)\), \(\theta \in [0, 2\pi]\), and \(\phi \in [0, \pi]\) in terms of Cartesian coordinates

\[
\begin{align*}
x &= r \cos \theta \sin \phi \\
y &= r \sin \theta \sin \phi \\
z &= r \cos \phi
\end{align*}
\]

the position vector is

\[
\begin{bmatrix}
r \cos \theta \sin \phi \\
r \sin \theta \sin \phi \\
r \cos \phi
\end{bmatrix}
\]

so the unit vectors are

\[
\begin{align*}
\hat{r} &= \frac{dr}{dr} = \begin{bmatrix} \cos \theta \sin \phi \\ \sin \theta \sin \phi \\ \cos \phi \end{bmatrix} \\
\hat{\theta} &= \frac{dr}{d\theta} = \begin{bmatrix} -\sin \phi \\ \cos \theta \\ 0 \end{bmatrix} \\
\hat{\phi} &= \frac{dr}{d\phi} = \begin{bmatrix} \cos \theta \cos \phi \\ \sin \theta \cos \phi \\ -\sin \phi \end{bmatrix}
\end{align*}
\]
A graphical representation of a spherical coordinate system is illustrated in Fig 1.3 [Weisstein 1999]:

Figure 1.3 Graphical representation of a spherical coordinate system
2 ROBOTIC MANIPULATOR PLANNING USING GENETIC ALGORITHMS

This project studied the possibility of applying genetic algorithms on a robotic manipulator in a software simulator for NASA. This simulator is a NASA proprietary program named MAGIK that is used to test the Shuttle Remote Manipulator System (SRMS) and the Space Station Remote Manipulator System (SSRMS). Software modules, employing genetic algorithms, were developed to find a near optimal path for the SRMS. Eventually, the results of this project may be applied to the SSRMS currently on the International Space Station (ISS).

2.1 Path Planning and Generation

This project approached path planning by means similar to those developed Yuval Davidor. His interest, as well as the interest of this project, was to capture the essence of trajectory generation so that an optimizing algorithm could be tested in a realistic and universal system [Davidor 1991].

Dvidor's experiment involved a 3-link 3-dof manipulator. His main purpose was to explore what he called the working envelope, which he defines as the geometric territory reachable by the end-effector. The arm configuration he constructs in Figure 2.1 has the structure of $A = (A_1, A_2, A_3)$ where each subscript denotes their respective links. It is important to note that the manipulator in this case is operating in a two-dimensional planar environment, each link being 1m in length [Dvidor 1991]. This approach was expanded to the SRMS, a 6-dof 3-link manipulator operating in a three-dimensional environment [Rogers 1998].
A displacement vector is a set of three angle changes corresponding to links 1, 2 and 3 respectively. The superscripts T and C denote the target and current positions respectively and Eq. 2.1 shows how to compute the displacement vector, which is scalar in this case [Davidor 1991].

\[
\Delta A = (\Delta A_1, \Delta A_2, \Delta A_3) = (\Delta A_1^T - \Delta A_1^C, \Delta A_2^T - \Delta A_2^C, \Delta A_3^T - \Delta A_3^C)
\]  

(2.1)

The inputs into Davidor’s simulation are composed of strings of link positions that define arm configurations. The displacement vectors are then computed according to Eq. 2.1 and the angle sequences are executed sequentially. The end-effector is assumed to have reached its target once it is within a 15-cm radius of the target. The angular difference of each is calculated between consecutive configurations [Davidor 1991].
Applying Genetic Algorithms to Manipulators

As mentioned earlier, the main purpose of a GA in this application is to search for a near optimal problem solution based on the least cumulative angular motion between all joints. In the particular environment of the SRMS, spherical coordinates are the best representation of the values. Only the configurations with a displacement location within the allowable tolerance range are considered as possible solutions. The actual tolerance ranges of the individual joints are shown in Table 2.1 [Rogers 1998].

The basic steps given in (1.1) are used to apply GA's to robotic manipulators. Initialization of the population consists of loading random individual link lengths and ranges of motion for each link. Selection then proceeds by using various selection methods to choose the parents to produce a new link configuration. Crossover is performed at a chosen link(s) for the two parents to exchange displacement information. Finally, mutation occurs to expand the mating pool. This is illustrated in Fig. 2.6 adapted from Davidor.

Figure 2.2 Davidor's illustrative sequence of arm-configurations of an \( n \)-link structure. Each segment of each configuration denotes the final angular position of the respective link.
2.3 The SRMS and the Methods

Application of the methods described above to the SRMS occurs in the following manner. Davidor's method deals with six-field string configurations executed sequentially. Each field represents an angle orientation of each joint on the SRMS. During the mating process, various joints are selected along each configuration to exchange information with another. These crossover and selection operations produce new possible solutions.

In calculating the available travel distance, joint limits and link lengths on the SRMS are supplied by NASA documentation as illustrated in Table 2.1 and Fig. 2.6. Table 2.1 shows the three different joint limits on the SRMS [Rogers 1998]. Only the
Software Stop limit is considered at this time. Figure 2.6 shows the SRMS's dimensions and joint limits.

Table 2.1 SRMS Joint Limits

<table>
<thead>
<tr>
<th>Joint</th>
<th>Reach Limit (deg)</th>
<th>Software Stop (deg)</th>
<th>Hardware Stop (deg)</th>
<th>Specification Travel (deg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shoulder Yaw (Joint 1)</td>
<td>+/- 175.4</td>
<td>+/- 177.4</td>
<td>+/- 180</td>
<td>+/- 180</td>
</tr>
<tr>
<td>Shoulder Pitch (Joint 2)</td>
<td>2.6</td>
<td>0.6</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td>Elbow Pitch (Joint 3)</td>
<td>-2.4</td>
<td>-0.4</td>
<td>2.4</td>
<td>2</td>
</tr>
<tr>
<td>Wrist Pitch (Joint 4)</td>
<td>+/- 114.4</td>
<td>+/- 116.4</td>
<td>+/- 121.4</td>
<td>+/- 120</td>
</tr>
<tr>
<td>Wrist Yaw (Joint 5)</td>
<td>+/- 114.6</td>
<td>+/- 116.6</td>
<td>+/- 121.3</td>
<td>+/- 120</td>
</tr>
<tr>
<td>Wrist Roll (Joint 6)</td>
<td>+/- 440.0</td>
<td>+/- 442</td>
<td>+/- 447</td>
<td>+/- 447</td>
</tr>
</tbody>
</table>
Figure 2.4 SRMS Dimension and Joint Limits
3. SYSTEM DESIGN

The system was designed in two parts. The first is the construction of the modules that generate the arm-configurations of a planned path for the SRMS and output them to script files in the format prescribed by the simulator. The second part runs the produced script files with the MAGIK robotic simulation program for visual verification. MAGIK is a robotic simulation program used to conduct interactive kinematic analysis of SRMS and SSRMS robotic operations [Rogers 1998].

3.1 Environment

The module and MAGIK script files were produced in the C programming language running on the Debian 3.01 Linux 2.4.21 platform. They were set up on the High Performance Computing Development Center (HPCDC) at Texas A&M University – Corpus Christi. The system is equipped with 12 Dell 2650 dual Xeon capable servers at 3.06 Ghz each with 12 GB total memory and approximately 1TB shared storage. Currently this system has a theoretical maximum of 36.72 Ghz processing bandwidth [Turner 2004]. The script files are opened by MAGIK and made to run each configuration sequentially. Microsoft Excel was employed to visual the produced data.

3.2 Path Production

The path planning of the SRMS for this project consisted of the following steps:

1) Calculate the most desired straight line path for the end effector to take as determined by known start and finish point coordinates
2) Dividing the line into twelve equal segments, one for each processor on the cluster, and calculating the corresponding (x,y,z) endpoints for those segments
3) Running the genetic algorithm to find the configuration of SRMS joints to position the end effector as close to a corresponding segment point as possible
4) Outputting each segment point configuration to a script file as prescribed by MAGIK
5) Running the produced script file in the simulator for visual verification of the path and destination

3.2.1 Calculating the Desired Straight Line Path

The first step in the path generation for the SRMS was to find the equation for the desired straight line. A perfect path is an absolute straight line, but the goal of the genetic algorithm was to find a most fit, or most nearly optimal, solution to the problem within a reasonable amount of time. In finding the line equation, an initial point and a finish point are known. Since the end-effector will be moving along this path, it can be treated as a displacement vector. With this in mind, The displacement vector \( \mathbf{v} \) with initial point \((x_1,y_1,z_1)\) and terminal point \((x_2,y_2,z_2)\) is:

\[
\mathbf{v} = (x_2-x_1,y_2-y_1,z_2-z_1) \tag{3.1}
\]

That is, if vector \( \mathbf{v} \) were positioned with its initial point at the origin, then its terminal point would be at \((x_2-x_1,y_2-y_1,z_2-z_1)\). Figure 3.1 shows \( \mathbf{v} \) from its initial point to its terminal point.
3.2.2 Dividing the Straight Line

The second step in this project was to divide the desired straight line into twelve equal segments. The division of a line from section 3.2.1 into that number was so that each processor would be executing the GA with a fitness value corresponding to how close a configuration landed the end-effector to each processor's segment point. Each segment point's coordinates were calculated by multiplication to a scale factor of \( n/p \), \( n \) being the processor number and \( p \) being the number of processors being used. This is illustrated in Figure 3.2. Each processor executes the algorithm using the coordinates \( n/p[(x,y,z)] \) as its goal to reach. After the specified number of generations has finished executing, the fittest configuration is the path point from that particular processor.
After the desired line equation is calculated and the line divided, each processor is sent the formula for determining its most fit point. This is the point the algorithm attempts to reach with the given number of generations. Testing was performed with 5000 generations as the maximum. It was after this many generations that overall fitness began to converge. The fitness heuristic used was nearness to the desired line segment point.

The algorithm starts off by producing an initial population of a size that is input by the user. This initial population is made up of random values for the five angles within their physical limits. Next, the end-effector \((x,y,z)\) coordinates are calculated, along with fitness. Once the initial population is created, it is rearranged in ascending order according to fitness. The part of the population with the more desired fitness values are then selected to move on to the next generation where they mated for reproduction. The new individuals replace those not chosen for mating, and the process repeats until the desired number of generations has passed. After the last generation, the processes send

---

**Figure 3.2 Dividing the desired straight line.** The line pictured has a distance of \(d\). The distance of a segment has a distance of \((n/p)d\). The variable \(n\) represents the processor number while \(p\) represents the number of processors on the cluster. The \((x,y,z)\) point of each segment is a scale factor of the final point.
their best configuration to process 0 where it arranges them into order. These are the points that the SRMS will follow. The last step is to write the joint angle values, (x,y,z) coordinates, and other necessary MAGIK commands to an external script file to be opened in the simulator. The algorithm is illustrated as follows:

NUMGENS = maximum number of generations
POPSIZE = population size
CONFIG = individual configuration
OPTCON = optimal configuration

Each processor generates initial POPSIZE number of angle CONFIGs randomly
Evaluate CONFIG fitness and (x,y,z) location of end-effector
Sort initial population in ascending order according to fitness
generation = 0;
While(generation < NUMGENS)
1. Select first half of population of mating
2. Produce offspring and replace back into population
3. Mutate small percent of generation
4. Re-evaluate (x,y,z) and fitness of end-effector
5. IF (Fitness(New) < OPTCON)
   THEN OPTCON = Fitness(New)
6. generation++;
IF(processor != 0)
   THEN Send local OPTCON to processor 0
ELSE
   Write OPTCON to external MAGIK script file

3.3 File System

In order to develop the script files needed, several files were created. Each file handles a specific set of operations that allow for the path generation. The files that were created and used are listed in Table 3.1 and accompanied with brief descriptions.

Table 3.1 Application Files
<table>
<thead>
<tr>
<th>Program</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constants.h</td>
<td>Generates random angle values for the configurations</td>
</tr>
<tr>
<td></td>
<td>Calculates end-effector (x,y,z) coordinates</td>
</tr>
<tr>
<td></td>
<td>Calculates point-to-point distance</td>
</tr>
<tr>
<td>Gaops.h</td>
<td>Performs genetic algorithm operations</td>
</tr>
<tr>
<td>Pathgen.c</td>
<td>Main program, calls other files</td>
</tr>
<tr>
<td></td>
<td>Assembles path and writes MAGIK script files</td>
</tr>
<tr>
<td>SetArrayElement.h</td>
<td>Performs array operations necessary for mating and regeneration</td>
</tr>
</tbody>
</table>

### 3.4 Script Files

The data output follows the format specified by MAGIK. In general, a MAGIK script statement is composed of four parts: an object type, an object name, a command keyword, and a command's parameters. An object name can be either MAGIK specific or can have another graphics node name. A command is simply a statement that affects the state or mode of the specified object. A sample command format that MAGIK will receive is as follows:

```
rms ["<name>"] position|rate <x><y><z><pitch><yaw><roll>
```

This command allows a user to specify the "name" of an object, in this case the SRMS, whereas the other variable are to specify the end-effector's (x,y,z) coordinates and the attitude configuration of the origin, this case the Space Shuttle [Rogers 1998]. The
values for x, y, and z are calculated, and the values for pitch, yaw, and roll can be coded using whatever origin orientation is needed.
4. EVALUATION AND RESULTS

The goal of this project was to seek a near optimal sequence of arm configurations for the SRMS. The most desired arm configuration(s) were those that came closest to producing the desired straight path for the end-effector to follow. The modules outputted the calculated information to script files in the format prescribed by the MAGIK Script Command Language. These script files were then accessed by the MAGIK simulator and run to observe whether or not the actual motions in MAGIK match those of the produced script files.

4.1 Configuration Fitness Analysis

During the experimental and evaluation phase of this project, tests were performed that evaluated how close to a destination point that the genetic algorithm could reach, given an initial population size and a number of generations. These tests showed that both population and generation size can influence the overall optimal configuration that was produced.

The tests were conducted within the physical constraints of the SRMS as prescribed by NASA. These constraints include the physical limits of all joint angles on the SRMS. Also, conditional checks were put into place so that the path generation program did not produce a configuration with a calculated reach beyond the physical length of the manipulator itself. These checks served a second purpose in that they ensured a configuration that would not collide with the Space Shuttle.

Values for the number of generation and population size were tested to visually graph how varying combinations of the two affect system performance. The number of
generations was set at 1500 as a maximum and the population size was set at 2000 as a maximum. Any values tested over the set maximums either produced convergence-like behavior or resulted in excessive memory and runtime.

4.2 Experimentation of Values

In the search for finding a near optimal configuration, values for generations and population size needed to be considered. For this project, the value set for population size is \{10, 50, 100, 500, 1000, 1500, 2000\}, and the set for generations is \{10, 50, 100, 500, 1000, 1500\}. These set were test in all possible combinations against the algorithms to not only track the general trend of the algorithm, but also to find a near optimal fitness configuration for the SRMS. The results of the tests are illustrated in the following tables and graphs. The tables differ according to the number of generations, and reflect the fitness values for each population size. The graphs are the visual representations of the tables.
Table 4.1 Data for 10 Generations

<table>
<thead>
<tr>
<th>Pop Size</th>
<th># gens</th>
<th>Distance from target (in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>21.46</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>26.803</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>22.429</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>5.034</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>5.512</td>
<td></td>
</tr>
<tr>
<td>1500</td>
<td>4.658</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>3.451</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.1 Graph of Algorithm for 10 Generations
Table 4.1  Data for 50 Generations

<table>
<thead>
<tr>
<th>Pop Size</th>
<th># gens</th>
<th>Distance from target (in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td>17.923</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td>17.252</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>5.454</td>
</tr>
<tr>
<td>500</td>
<td>50</td>
<td>0.362</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>5.373</td>
</tr>
<tr>
<td>1500</td>
<td></td>
<td>3.828</td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td>0.362</td>
</tr>
</tbody>
</table>

Figure 4.2  Graph of Algorithm for 50 Generations
Table 3.3 Data for 100 Generations

<table>
<thead>
<tr>
<th>Pop Size</th>
<th># gens</th>
<th>Distance from target (in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>12.327</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>26.803</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>11.575</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>0.362</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>0.362</td>
<td></td>
</tr>
<tr>
<td>1500</td>
<td>2.253</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>0.362</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.3 Graph of Algorithm for 100 generations
Table 3.4 Data for 500 Generations

<table>
<thead>
<tr>
<th>Pop Size</th>
<th># gens</th>
<th>Distance from target (in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td>9.692</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td>9.692</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>6.578</td>
</tr>
<tr>
<td>500</td>
<td>500</td>
<td>0.362</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>7.574</td>
</tr>
<tr>
<td>1500</td>
<td></td>
<td>1.889</td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td>0.362</td>
</tr>
</tbody>
</table>

Figure 4.4 Graph of Algorithm for 500 Generations
Table 4.5 Data for 1000 Generations

<table>
<thead>
<tr>
<th>Pop Size</th>
<th># gens</th>
<th>Distance from target (in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>9.594</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>9.431</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>2.073</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>0.362</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>2.77</td>
<td></td>
</tr>
<tr>
<td>1500</td>
<td>3.846</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>4.303</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.5 Graph of Algorithm for 1000 Generations

Fitness v. Popsize (#gens = 1000)

Figure 4.5 Graph of Algorithm for 1000 Generations
Table 4.6 Data for 1500 Generations

<table>
<thead>
<tr>
<th>Pop Size</th>
<th># gens</th>
<th>Distance from target (in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td>7.113</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td>1.889</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>1.889</td>
</tr>
<tr>
<td>500</td>
<td></td>
<td>0.362</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>2.77</td>
</tr>
<tr>
<td>1500</td>
<td></td>
<td>3.846</td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td>0.362</td>
</tr>
</tbody>
</table>

Figures 4.7 through 4.9 graphically show the end-effector positions when ran through MAGIK using varying values for the number of generations and population size. The values used for these trials are \{100, 500, 1000, 1500\} where generations and population were set at the same values. From the three graphs, it can be seen that as intended, the path of the SRMS became smoother as population and generations are increased. The
plots the value of 1500 show enough smoothness that it starts to show signs of convergence when compared to those of 1000. Figure 4.7 shows the plot of x positions versus y positions. Figure 4.8 shows the plot of x positions versus z positions. Figure 4.9 shows the 3D plot of the different trials.

Figure 4.7  Plot of x positions versus y positions for each series
<population size><number of generations>
Figure 4.8 Plot of x positions versus y positions for each series
<population size><number of generations>
4.3 Evaluation of Results

In general, all combinations of population sizes and generations produced results that approached zero distance as the population size increases. All the tests did experience short periods of local minimums, but the general trend was that of a closer approach towards a zero distance. The same trend was observed as the number of generations was increased in an attempt to find a configuration closer and closer to a zero distance from the destination.

During testing, the absolute closest distance to the destination was a value of 0.362 inches. This value occurred numerous times, but more often occurred when the population and generations reached their respective ceilings. This may indicate that the
algorithm would need some slight alterations. Further alteration of the algorithm was not
done because the value of 0.362 inches is an almost absolute optimal distance. The
improvement that was observed occurred when generation and population values were
increased.
5. FUTURE WORK

This project provided some groundwork for producing more elaborate algorithms to handle other applications as needed by NASA. The benefits from such path learning software will be evident in the improved efficiency and accuracy in the movement and operation of both the SRMS and the SSRMS.

Other areas for the expansion of this project include payload handling, collision detection and further expansion to the SSRMS. Future implementations could deal with how the SRMS would behave with a payload attached. Path planning can be integrated with collision detection, to make this a reliable means by which to control the manipulator as opposed to the current method.
6. CONCLUSION

The goal of this project was to produce software modules that would search and find a near optimal arm configuration sequence for the SRMS. The importance of this project is that NASA currently does not have software that will do such a task. It evaluated fitness on the basis of the distance that the end-effector lands at the end of its path away from a pre-determined destination.

Results of tests were recorded and analyzed graphically to observe the overall trend of the algorithm as the values of population and generations were increased. It was observed that the overall fitness improved as both were increased. The final result was a configuration that placed the end-effector only 0.362 inches away from the desired destination point.

This project demonstrated that genetic algorithms are more than reliable, and are definitely a viable alternative to human operation of the SRMS. This project may provide a way to reduce human error by allowing a program to plan the path for the SRMS in actual operation.
ACKNOWLEDGEMENTS

Luke Wilson, Jason Picarazzi, and Simon San Miguel for their invaluable help during the implementation process.

Barry Rogers at the Titan Corporation for his invaluable technical help on MAGIK.
BIBLIOGRAPHY AND REFERENCES


APPENDIX

The accompanying CD contains digital copies of this report, the project presentation, and source code for all the files listed in Table 3.1.