

**Simulation of the Inverse Kinematics of a Protein Backbone in Robotics Application**

by

Nabila Lau

Dissertation submitted in partial fulfillment of  
the requirements for the  
Bachelor of Technology (Hons)  
(Information and Communication Technology)

MAY 2011

*Universiti Teknologi PETRONAS  
Bandar Seri Iskandar  
31750 Tronoh  
Perak Darul Ridzuan*

**CERTIFICATION OF APPROVAL**

**Simulation of the Inverse Kinematics of a Protein Backbone in Robotics Application**

by

Nabila Lau

A project dissertation submitted to the  
Computer and Information Sciences Department  
Universiti Teknologi PETRONAS  
in partial fulfillment of the requirement for the  
BACHELOR OF TECHNOLOGY (Hons)  
(Information and Communication Technology)

Approved by,



**Dr Alan Oxley**  
Professor  
Computer & Information Sciences Department  
Universiti Teknologi PETRONAS

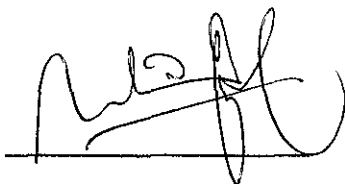
*(PROF. ALAN OXLEY)*

MAIN SUPERVISOR

UNIVERSITI TEKNOLOGI PETRONAS  
TRONOH, PERAK  
MAY 2011

## **CERTIFICATION OF ORIGINALITY**

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

A handwritten signature in black ink, appearing to read 'Nabila Lau', written over a horizontal line.

**NABILA LAU**

## **ACKNOWLEDGMENT**

I would like to thank my supervisors, Prof. Dr. Alan Oxley and Dr. Alan Giffin Downe, for their support in the research such as the valuable time spent for discussions. Aside from that, I would like to acknowledge the strong support I have from the university regarding my final year project. Also, thank you to my family for the motivation I received throughout the developmental phase of the project.

## **Simulation of the Inverse Kinematics of a Protein Backbone in Robotics Application**

Nabila Lau

Faculty of Computer and Information Sciences, Universiti Teknologi PETRONAS

### **Abstract**

A lot of research had been conducted regarding the structures of protein, one of the fundamental macromolecules in the human body system. In order to understand how diverse biological functions work, it is crucial to researchers to study protein structures and their dynamic behavior. This leads to the several successes in the field of computational structural biology –through combining both technology and biology fields together. Moreover, recent studies promote the research on protein backbone and its possible flexible movements for the purpose of exploring the possibilities of protein macromolecule have for advancement of the scientific field.

Hence with the inspiration from those studies and through the incorporation of Inverse Kinematics (IK) algorithm, this work will focus on how protein backbone of **minimal 3 joints movements** will be modeled in the form of robotic appendages with the respective dihedral angles similar to that of an ideal protein's using one existing algorithm. Theoretical movements of the robotic appendages will be simulated as well.

These robotic appendages aim to prove their usefulness in various applications such as in medical, manufacturing and other related robotic domains. Application of protein backbone inspired robotic appendages is further accentuated in the form of 3D simulation in this project.

*Keywords: Inverse Kinematics, Protein backbone, Biologically-inspired robots*

## TABLE OF CONTENTS

<b>List of Figures .....</b>	<b>4</b>
<b>List of Tables.....</b>	<b>5</b>
<b>List of Appendices .....</b>	<b>6</b>
<b>Nomenclature.....</b>	<b>7</b>
<b>Chapter 1: Introduction.....</b>	<b>8</b>
1.1. Project Background .....	8
1.2. Problem Statement.....	10
1.3. Objective and Scope of Study .....	10
<b>Chapter 2: Literature Review .....</b>	<b>13</b>
2.1. Protein flexibility in backbone design .....	13
2.2. Solutions for IK applicable for protein backbone .....	14
2.3. Protein Backbone as Robotic Appendage .....	18
2.4. Computer Simulation.....	20
<b>Chapter 3: Methodology .....</b>	<b>22</b>
3.1. Research Design .....	22
3.1.1. Algorithms Proposed .....	24
3.1.1.1. Triangulation .....	24
3.1.1.2. Faster Gradient Following Algorithm .....	26
3.1.1.3. Inverse Jacobian .....	28
3.2. Past Results for research.....	29
3.3. Instrumentation.....	30
3.4. Procedure.....	30

3.5. Efficiency Analysis .....	33
3.6. Modeling.....	33
3.7. Software Testing.....	40
<b>Chapter 4: Results and Discussion.....</b>	<b>42</b>
4.1. Prototype.....	42
4.2. Data Gathering and Analysis Method .....	46
4.3. Results and Analysis.....	46
4.3.1. Correctness of the Appendage Movement .....	46
4.3.2. Efficiency of the Robotic Appendage .....	47
4.3.3. Usability Testing: System Usability Scale .....	48
4.4. Discussion.....	50
4.5. Potential Applications in Robotics .....	52
<b>Chapter 5: Delimitations, Recommendations and Conclusion .....</b>	<b>53</b>
5.1. Delimitations .....	53
5.2. Recommendations .....	53
5.3. Conclusion.....	54
<b>References .....</b>	<b>55</b>
<b>Appendices .....</b>	<b>59</b>

## LIST OF FIGURES

<i>Figure 1: Simple illustration of a robotic appendage modeled similarly to a protein fragment.</i>	11
<i>Figure 2: CCD solution method for protein loop closure problem (reprinted with permission from Canutescu &amp; Dunbrack, 2003)</i>	17
<i>Figure 3: Ideal values of protein backbone angles (reprinted from Berkholz et al, 2000 with permission from Elsevier)</i>	22
<i>Figure 4: Law of Cosines</i>	25
<i>Figure 5: Triangulation Method</i>	26
<i>Figure 6: Simple two-jointed appendage with target</i>	27
<i>Figure 7: Gantt chart for FYP1 and FYP2</i>	31
<i>Figure 8: Spiral Model</i>	32
<i>Figure 9: Simplified Petri Net model for one algorithm</i>	34
<i>Figure 10: Petri Net Model for two algorithms</i>	36
<i>Figure 11: State Reachability Graph</i>	38
<i>Figure 12: Rough GUI design</i>	42
<i>Figure 13: GUI using OpenGL</i>	43
<i>Figure 14: Welcome Screen</i>	43
<i>Figure 15: Window for the Biomimetic Robotic Appendage</i>	44
<i>Figure 16: Window for the moving appendage and random target</i>	44
<i>Figure 17: Functions in both windows</i>	45
<i>Figure 18: Rotated views in both windows</i>	45
<i>Figure 19: Correctness of the Appendage</i>	47



## LIST OF TABLES

<i>Table 1: Key Milestones for FYP2</i>	<i>31</i>
<i>Table 2: SUS result</i>	<i>49</i>
<i>Table 3: SUS result in graphical form</i>	<i>49</i>
<i>Table 4: Range of robotic precision</i>	<i>51</i>

## LIST OF APPENDICES

<i>Appendix 1 (Activity Diagram)</i> .....	59
<i>Appendix 2 (Sequence Diagram)</i> .....	60
<i>Appendix 3 (Class Diagram)</i> .....	61
<i>Appendix 4 (Data Flow Diagram)</i> .....	62
<i>Appendix 5 (Context Diagram)</i> .....	63
<i>Appendix 6 (SUS Questionnaire)</i> .....	64
<i>Appendix 7 (Coding Snippets)</i> .....	65
<i>Appendix 8 (Prototype)</i> .....	74

## NOMENCLATURE

### 1. Biology Field

Term	Definition
Amino-acids	A molecule consisting of the basic amino group (NH <sub>2</sub> ), the acidic carboxylic group (COOH), a hydrogen atom (-H), and an organic side group (R) attached to the carbon atom
Antibody	Protein used as immune defense against foreign agents (antigens)
Atom	The fundamental building block of a chemical element.
Backbone	The main structural feature of a polymer (chain-like) molecule from which many side chains branch off
Bond-length	Distance between the nuclei of two atoms which have formed bonds with each other
Conformation	The three-dimensional arrangement of side groups on a molecule which can freely rotate into different positions without breaking any bonds.
Enzyme	A catalyst or a chemical produced by cells to speed up specific chemical reaction
Sequence library	Database that contains the order in which subunits appear in a chain, such as amino acids in a polypeptide or nucleotide bases in a DNA or RNA molecule
Side-chain	Atoms of an alpha-amino acid other than the alpha-carboxyl group, the alpha-amino group, the alpha-carbon, and the hydrogen attached to the alpha-carbon

Note: All definitions of terms are taken from <http://www.biology-online.org/dictionary/>

### 2. Kinematics Field

Dihedral angle	Figure formed by two intersecting planes
End effector	Device at the end of a robotic appendage, designed to interact with the environment

Note: All definitions of terms are taken from <http://www.websters-online-dictionary.org/definitions/>

## CHAPTER 1

### INTRODUCTION

#### 1.1. PROJECT BACKGROUND

A protein (Creighton, 1993) is portrayed as sequence of amino-acids, linked by bonds of peptides. The *backbone* is modeled as a serial linkage with protruding side-chains. Three significant atoms are N, C<sub>α</sub>, and C; contributed by the amino acids with two dihedral degrees of freedom (DOF) to the backbone. A rising interest in biology is the study of IK movements with regards to a protein backbone.

Kinematics is a branch of mechanics that portrays motion in abstraction without any reference to mass and force. The most commonly known solutions for the kinematics of manipulators will be forward kinematics and IK. Forward kinematics is to find the position of the end-effector given the links lengths and joint angles. IK, on the contrary, seeks for the values of joint angles given the position and orientation of the end-effector.

As always, IK is considered a much difficult problem to solve as compared to Forward Kinematics. This is because in contrast with IK, the angles in Forward Kinematics are known in advance prior to solving the position of the end effector. Another difficulty with IK is that there could be several possible solutions (even to an infinite level) and there lies a possibility where no solutions could be found as well (Kavraki, 2007). Hence, it is important to choose the right computational algorithm for IK solutions in order to produce the sought-after results in applications.

In animated and robotic motions, IK has been known and used widely. In comparison to human complex actions, all of us perform IK everyday by reaching out for books on the table, without even realizing the angles and the position of our limbs with respect to the simple and impulsive movement. In IK situations, the position of end effector is already pre-determined, and algorithms via mathematical solutions can be developed to calculate the arrangements of joints with the possible angles depending on the number of DOFs.

Meanwhile, the term 'biologically-inspired' (also known as *biomimetic*, *biomimicry*, and *bionics*) is the emulation of nature designs into solving human's everyday problems. In the recent years, biologically-inspired robots are getting popular in research studies and industry applications.

The emergence of these robots has already created a spectacular dimension of design stage involving various mimics of biological nature. The importance of nature mimicry in robotic designs is described aptly in this note: "Fundamental understanding of the morphology and functionality of soft structures in nature, however, increases insight and can lead to new design concepts in soft robotics. The natural world demonstrates the potential capabilities of soft robots" (Trivedi, Rahn, Kier, & Walker, 2008).

Hence, the concept of biomimicry can be applied to the movements of the protein backbone structure in order to show how they (backbone structural conformations) can be handy in robotic applications.

## **1.2. PROBLEM STATEMENT**

However, simply by modeling and assembling the biomimetic robotic applications based on protein backbone structures from scratch are costly, time-consuming and difficult. It would be best to simulate the physical properties of the robot properly using a visualization program to gain richer insight of how far the research could go and predict the usefulness of the biomimetic protein-based robotic application.

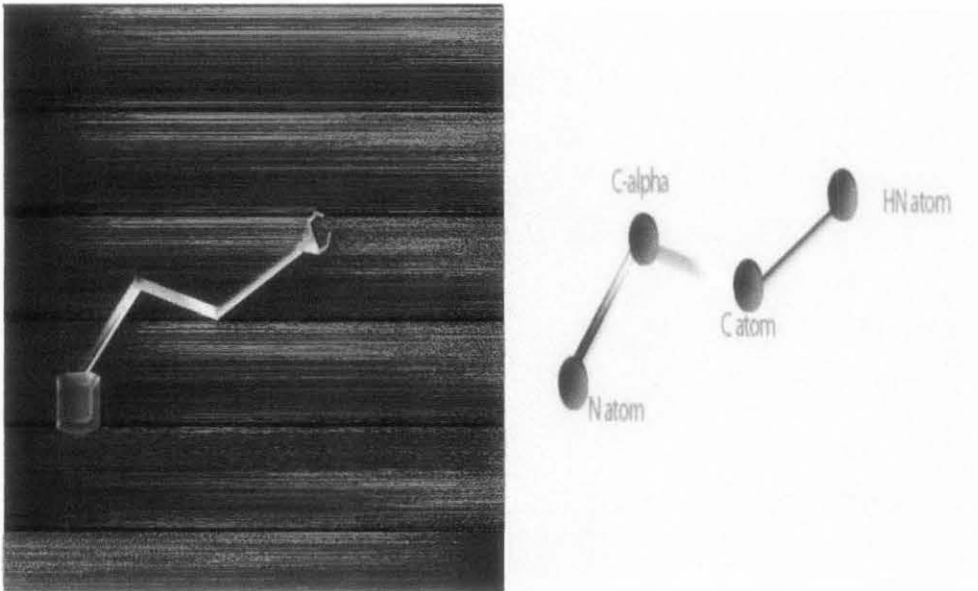
Aside from that, as much as protein backbone has been compared to a robotic application, biomimetic simulation of the possible movements of the robotic appendages via IK algorithm is something new and yet to be explored further. Most IK algorithms only portray a solution at a time and do not show intermediate positions of the robotic appendage.

Thus, a 3D simulation program will be the best approach for this research. The visualization program itself will ease the understanding of IK concept. Animated and graphical representations of the appendages are provided as well by the system.

## **1.3. OBJECTIVE AND SCOPE OF STUDY**

The main idea of this project involves integration of the domains from both robotics (IK concept) and biology (protein backbone flexibility) fields for the 3D simulation program. The concept of biologically-inspired robots is therefore applied in this project. With regards to the significance of biologically-inspired robots in this project, the protein backbone flexibility is duly captured for the purpose.

Hence, one could observe how the protein backbone itself can be modeled into robotic appendages with similar joints and number and value of DOFs by concentrating a **minimal of 3 joints movements** (Figure 1).



**Figure 1: Simple 2D illustration of a robotic appendage modeled similarly to a protein fragment.**

Thus, a system for the visualization of protein backbone structures in robotic appendages via IK algorithm is envisioned for this project. On top of that, this project aims to explain and predict the performance of robotic applications via the visualization system. The system is could be used to improve the design of protein-based robotic solutions in the near future. A simulation result will be carried out using OpenGL libraries and Microsoft Visual Studio 2010 using primarily the C++ language. The project will be completed, as expected by the university, by the end of the second semester of the Final Year.

The 3D simulation program will be also useful to those who would like to know more about IK concepts and its applications in biomimetic robotics. Hopefully, the research completed by the author would be beneficial as a predictive tool for domestic and non-domestic applications involving the robotic appendages modeled to ideal protein backbone structure.

To summarize, the main objectives of this research will be as follows:

- To create a visualization system for the biomimetic robotic appendages
- To explain and predict the performance of robotic application via the system



## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1. PROTEIN FLEXIBILITY IN BACKBONE DESIGN**

Most early design methods of the protein were through fixed atomic positions on a backbone template (Mandell, 2009). Simulation from fixed to flexible backbone structures involved complex calculations and large computational resources which were limited back then. The ultimate goal for the design of protein was to engage researchers in searching for sequences of amino acids that will form stable protein structures. Due to the concepts formulated over the years, the knowledge of protein structures has been implemented such as in enzyme designs, sequence libraries, and others.

However, protein backbone has to be flexible in nature in order to accommodate the crucial biological processes, such possible conformational changes to the backbone during processes in the enzymes and antibodies. Therefore, backbone flexibility has to be modeled accordingly to understand the protein sequence, functions, and structures as the sequences of protein will determine the unique 3D structures, which in turn determine the functions of the protein itself (Wu, 2008). As a result, predictive methods have been formulated via computational modeling in order to design and construct new probable functions of protein (Mandell & Kortemme, 2009).

In addition, it is vital to note that the various shapes of a protein structure are due to torsion angles in the backbone, not really to the bond-length of the protein (Coutsias et al, 2005). Hence, DOFs of the robotic appendages are important to be modeled accurately in this project as per abstraction of the shape of idealized protein structure.

Representations of protein motion had been accurate when using Monte Carlo and molecular dynamics simulation techniques (Adcock & McCammon, 2008). Monte Carlo involves the usage of random steps in generating a series of conformations, which molecular dynamics involves Newton equations to compute particle motions. Both of these simulation techniques are computationally intensive, especially for particles with many DOFs.

This brings into the picture of how protein motions are important to the protein loop closure problem. Loop closure problem is the search of the suitable segments in a molecule of chain that is “geometrically consistent with preceding and following parts of the chain whose structures are given” (Coutsias, Seok, Jacobson, & Dill, 2003). Among some of the reasons protein loop closure being actively studied by biologists are that loops of protein play vital roles with respect to protein binding as well as its active activity as enzymes.

Recently, the IK concepts, which have been used widely in robotic field, are applied to protein backbone’s flexibility modeling (Canutescu & Dunbrack, 2003). The similarity between the two fields lies in the notion that both of them needed IK to solve the loop closure problems in protein, and closure problems in robotic kinematics. DOFs can be represented as main adjustable parameters in the form of dihedral angles in both of the problems (Liu, 2006).

## **2.2. SOLUTIONS FOR IK APPLICABLE FOR PROTEIN BACKBONE**

IK itself has been used widely in computer animated and robotic-based motions. One of the IK-based graphical animation simulation works is Style-Based Inverse Kinematics

(Grochow, Martin, Hertzmann, & Popovic, 2004) which uses IK to produce the most likely human poses while satisfying constraints. The Style-Based IK provides accurate image outputs of human poses and this program can be used for generation of poses used in computer animation and vision.

However, one major drawback of the Style-Based IK program will be the inability to model dynamics and the need for speed, especially for real-time synthesis for optimization. The paper also lacks the lead to future work involving robotic mechanisms while capturing IK in real-life applications rather than in just animation and computer graphics.

Looking into the motivation of such graphical simulation program, on the other hand, this brings the concern of the existing probable IK algorithms to be used to solve and generate movements of robots towards target.

As mentioned earlier, IK is used to solve loop closure problem. Several solutions have been formulated to overcome the loop closure problem in protein structures. Examples of the IK solvers are the numerical and analytical solvers (Ho, Komura, & Lau, 2005). Numerical solvers seek to get new 3D coordinates of end effectors close to current orientation through iteration and approximation whilst the analytical solvers use calculations by inversion of forward kinematics equations.

When we compare between the two, analytical solvers can provide solutions very quickly but only applicable for simple structure. An instance of an analytical solver is the Law of Cosines. Meanwhile, an example of the numerical solver will be the Jacobian matrices.

In 1999, Wedemeyer and Scheraga had solved the problem via polynomial equations for tripeptides with 6 DOFs. Jacobian matrix of the first derivatives of the distances of atoms of the loop, DOFs taken into calculation, has been used as an algorithm in solving loop closure problem. The algorithm itself, referred as 'random tweak', uses the Lagrange multipliers by minimizing changes in dihedral angles while at same time satisfying constraints on the between atom to atom distances (Shenkin et al., 1987).

As of today, IK solution method based on Lagrange multipliers is known as the best solver (Ho, Komura, & Lau, 2005). However, this method could not handle inequality constraints and its computational time grows cube proportional to the number of constraints. The end result will be poorer performance and higher costs when there are more constraints.

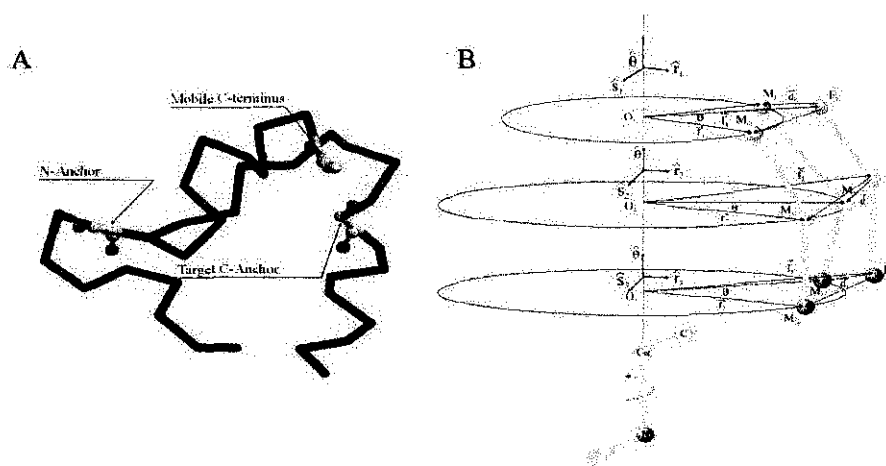
As a result, linear programming has been introduced by Ho et al (2005) via the Linear Programming based IK solver (LPIK) as its performance is almost at par with the Lagrange multipliers method and is more suitable for real-time applications, such as 3-D games and virtual environment, with large DOFs and constraints. Ho et al portrayed the performance of LPIK is still stable and efficient even with multiple animation objects simulated (also mean more constraints). However, for the subject of protein backbone in this project, such solution is not needed as DOFs and constraints are not large.

Another popular approach of solving IK which is also a practice in the protein backbone is through the Cyclic Coordinate Descent (CCD). CCD was originally developed to solve IK in robotics (Wang & Chen, 1991).

In CCD algorithm, the end effector of the robotic appendages will move towards the target object by adjusting one degree of freedom at a time (Canutescu & Dunbrack, 2003). This is an iterative algorithm whereby the joints of the appendages are modified along the DOFs in order to bring the end effector closer to the target object.

There are some reasons as to the usage of CCD to compute protein backbone movements and loop closure IK-based solutions (Canutescu & Dunbrack, 2003). Advantages of CCD are inclusive of that CCD is easy to implement and its algorithm can be computed quickly as compared to Jacobian and Lagrange. Aside from that, CCD is also numerically stable.

Nonetheless, the con with this solution method is there will be no guarantee for the return of all solutions, and even if there is a solution, chances are CCD may miss it. For the subject of this project, such disadvantage is not considered of much weight as this project does not focus on providing all solutions. The concern is more on the ability of the robotic appendages to move in order to achieve the target in a sensitive manner.



**Figure 2: CCD solution method for protein loop closure problem (reprinted with permission from Canutescu & Dunbrack, 2003)**

In the paper written by Endou et al (2006), IK had been applied in modeling protein complex movements in order to study the conformation changes in protein backbone, where the changes are notable. The research applied motion planning which is widely used in robot motion and probabilistic road mapping. The study of IK application in the paper is not comprehensive as the Endou et al emphasized more on the physiology of the six protein backbone structures and their possible movements used.

Endou et al shows the relationship of protein backbone flexibility modeling and robotic abstraction although they did not elaborate on the connection of how IK solving method for protein backbone flexibility can be applied on various robotics applications as well. This is the gap where the author seeks to identify in close relation to this project by using two different IK solver algorithms.

### **2.3. PROTEIN BACKBONE AS ROBOTIC APPENDAGE**

The inspiration for protein backbone to be modeled as robotic appendage could be as simple as how the similarity between them is very much apparent, both in design and solution methods used. Biologically-inspired robotics and applications had been introduced much earlier among humankind throughout the evolution of life. Human wants and desire always cause human beings to refer to nature for solutions and inspirations (Cohen & Breazeal, 2003).

Biomimicry, a term which originated from Greek words, is used for this biological-inspired approach as a design principle. The term is straightforward; it simply means imitation of the natural biological designs in various domains of application.

As we can see in the paper produced by Czyzewski and Barron (2007), protein and peptide structures had been studied for biomimicry purposes. The mimicry was aimed after scrutinizing the structural designs of protein in order to assist prediction of function of protein. However, the methodology of mimicry performed was not elaborated much in the paper even though the authors recognized that this is a new field, only recently explored into after foreseeing the prospective possibilities of venturing deeper into the studies of protein structure and its flexibility.

Meanwhile, in the article written by Fay and Snoeyink (2003), techniques from robotic had been applied to miniscule fragment of protein backbone, which refer to the IK concepts. However, their concepts were more to reverse engineering; from the formulation of a robotic appendage's (which has 6 dihedral DOFs) IK solutions and then, apply those robotic concepts on fragments of protein backbone. The protein backbone is also designed with 6 dihedral DOFs. In return, the IK solver will return all real-valued solutions with unchanged chain endpoint and orientations. The more the degrees of freedom are involved, the more solutions could be found. Nonetheless, the article also did not elaborate more beyond the conceptual of robotic physics on the protein backbone structure. Hence, the notion of implementing and realizing robot mechanism from fragments of protein backbone is understudied in the paper and also other papers to date (Singh, Latombe, & Brutlag, 1999).

Another inspiration for the biological touch in this project could come from the several robotic applications these days. One of them is the biomimetic approach to IK of a robot appendage, which is modeled based on a human arm (Artemiadis, Katsiaris, & Kyriakopoulos, 2009). The benefit of this paper is that the application of IK is very apparent with robot appendage's 5 DOFs movements. With observation, the IK approach in solving closed-loop was used in the robot joint trajectories and new

anthropomorphic motions of the robot moving in the complex 3D environment. On a definite note, the paper itself serves to illustrate on how the research could inspire the modeling of this project's robotic appendages to a protein backbone structure and mimic the possible movements at the DOFs.

#### **2.4. COMPUTER SIMULATION**

Last in this section, the usage of computer simulation is highlighted appropriately for the benefit of the project. Simulation in the form of 3D has been used for various researches and other fields like tourism, gaming visuals, mechanistic view, proposed experimental environments, and others. Interactive simulations are said to be able to stimulate recalls from learner, involving preceding knowledge and its application (Bill, 2003).

Among the ways to support cognitive knowledge is through a computer generated situation through tools like games, role playing and graphical simulation alone.

For the purpose of better understanding experimental theories or proposition, simulation program itself is indeed useful. Some of such simulation programs can be seen through the experiments conducted on possible mechanism of surgical robotics. Such as, in the journal written by Hayashibe et al on robotic surgery simulation, the motion of a surgical robot is simulated in advance to be used for the pre-operative planning and training procedures. The modeling of ZEUS, a robotic appendage used for surgical operation, is integrated with IK and correspondent DICOM images for the simulation program. Experienced surgeons who used the system are happy with the realism of the robot motions and deem the system is useful in pre-surgical settings (Hayashibe et al, 2006).



To complete this section, the three main areas of the research, namely the protein backbone, inverse kinematics, biologically-inspired robots, have been addressed accordingly to illustrate the relationship of the three with respect to the project. In this paper, biomimicry using the protein backbone for the visualization of robotic appendages in a simulation program is hereby proposed.

Unlike previous works which focus more on applying the abstraction of robotic theories on the fragment of protein structure, this work intends to prove the reverse possible on realizing robotic applications based on the modeling protein backbone's dihedral angles and DOFs. 3D simulation program will prove the possibility of mimicking the structure of protein backbone on robotic appendages which are useful for both domestic and non-domestic applications.

In addition, the IK algorithms are applied to the movement of the robotic appendages in achieving target.

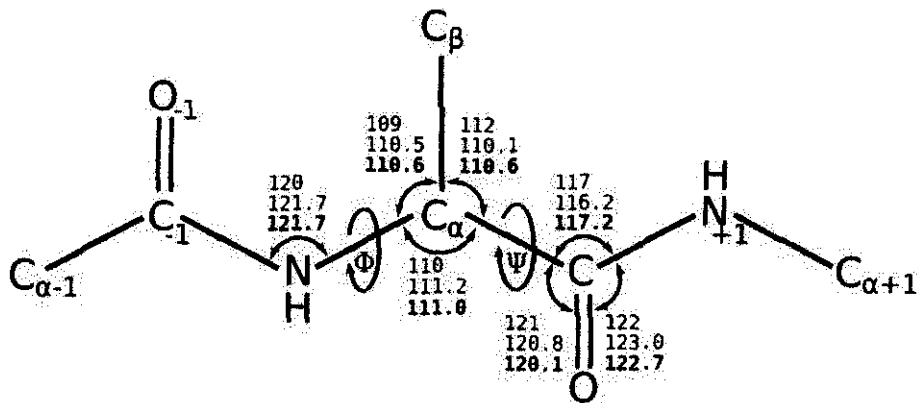
The rest of the paper continues with the methodology, results and discussion. The paper then concludes with delimitations, recommendations for future studies and conclusion.

## CHAPTER 3

### METHODOLOGY

#### 3.1. RESEARCH DESIGN

The main output of this project is a stand-alone application of simulation-based modeling and movements of the robotic appendages modeled similarly to a protein backbone with ideal angles (Figure 3).



**Figure 3: Ideal values of protein backbone angles (reprinted from Berkholz et al, 2000 with permission from Elsevier)**

IK movements were simulated evidently through the robotic appendage as that is purely envisaged for the program – to show the audience how IK could be used for robotic appendages applications.

It was important to observe and analyze the protein backbone movements before designing the robot based on the backbone. The number of DOFs were considered and fixed since the more DOFs, the more solutions (which are infinite) produced. 6 DOFs should be perfect for this project. The aim of the robotic appendages was to be able to move towards a randomly moving target in the simulation later. Analysis was completed regarding the efficiency and precision of the movements.

Given the desired position, the robotic appendage must be able to reach the position while arrangements of links to the movement and angles of DOFs (with constraints from protein backbone model of course) programmed. In the beginning, CCD algorithm was first thought of use for the research purpose but the algorithm was too complex to be implemented in a short duration of developmental phase. Hence, unlike most IK based robotic simulations, CCD is not used in this project.

During the developmental phase of the research, two new algorithms were chosen for the system, namely the triangulation and faster gradient following algorithms. The system was first envisaged to portray robotic appendage movements using these two algorithms. However, due to time constraint, two algorithm implementations were not feasible for this project. Hence, in this case, only one algorithm will be tested for the program, chosen from either triangulation, faster gradient following or using numerical method which guarantees reaching target precisely: inverse Jacobian and Euler mathematics.

### 3.1.1. ALGORITHMS PROPOSED

#### 3.1.1.1. TRIANGULATION

This algorithm was developed by Mukundan and Muller-Cajar and was tested against the popular CCD algorithm for its efficiency. The algorithm uses the law of cosines to determine joint angles quickly when given a target. The similarity of this algorithm with CCD is that it also iterates through every joints to the end-effector.

---

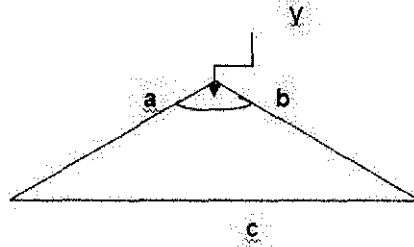
```
foreach Joint  $i$  do
  Calculate  $\vec{a}_i, c_i$ ;
  if  $c_i \geq a + b$  then
     $\vec{a}_i = \vec{c}_i$ ;
  end
  else if  $c_i < |a - b|$  then
     $\vec{a}_i = -\vec{c}_i$ ;
  end
  else
     $\theta = \cos^{-1}(\vec{a} \bullet \vec{c}) - \cos^{-1}(\frac{-(b^2 - a^2 - c^2)}{2ac})$ ;
    if  $\vec{a}_i = -\vec{c}_i$  or  $\vec{a}_i = \vec{c}_i$  then
       $\vec{r} = (0, 1, 0)$ ;
    end
    else
       $\vec{r} = (\vec{a} \times \vec{c})$ ;
    end
    rotate( $\vec{a}_i$  by  $\theta$  about  $\vec{r}$ );
  end
end
```

**Algorithm 1:** The triangulation algorithm for a kinematic chain with no constraints

Law of Cosines is used in calculating general triangles:

$$c^2 = a^2 + b^2 - 2ab \cos \gamma, \quad [1]$$

$c$  is the triangle side opposite of angle  $\gamma$  as shown in the figure below.



**Figure 4: Law of Cosines**

Formula [1] is used to complete a triangle in the three-dimensional space. In this algorithm, properties of triangles are used to move the robotic appendages towards the target. We require one iteration only to reach the target, first moving the joint which is furthest away from the end-effector.

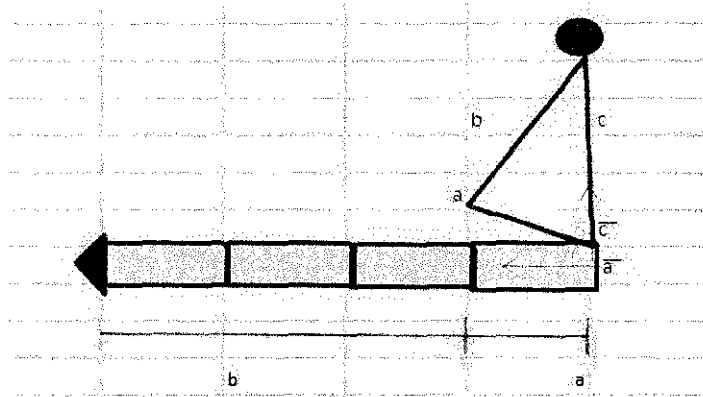
In the ideal case, the angle of  $\theta$  can be calculated through the formula as per below:

$$\theta = \cos^{-1}(\vec{a} \cdot \vec{c}) - \delta_b \quad [2]$$

Each of the robotic appendages will be rotated by  $\theta$  calculated in [2] respective to the axis of rotation vector  $r$ :

$$\vec{r} = (\vec{a} \times \vec{c}) \quad [3]$$

Also, the first two appendages will be rotated, leaving the remaining ones straight. However, it would also consider possibilities of non-ideal cases such as when the target is too far or too near from the current appendage.



**Figure 5: Triangulation Method**

When the target is too far from appendage,  $c > a+b$  whereby the solution to this is by rotating the appendage until the vector  $a$  is equal to target vector  $c$ .

$$\vec{a} = \vec{c} \quad [4]$$

On the other hand, when target is too close,  $c < |a-b|$  hence rotate the current appendage so that vector  $a$  is equal to negative of vector  $c$ .

$$\vec{a} = -\vec{c} \quad [5]$$

The algorithm was not implemented in any graphics application yet. Probably it was still under experimentation that it was not efficient enough to be used in graphics involving 3D animation.

### 3.1.1.2. FASTER GRADIENT FOLLOWING ALGORITHM

This algorithm is a variant from the Simple Gradient Following to solve the problem of taking a long time for the tip of the joints to reach the target. With this algorithm, the tip or end-effector could move quickly to the target as calculation reaches nearer to a minimum value. Below is the algorithm:

```

dist = Calc_Distance(a, b)

old_gradient_a = 0
old_gradient_b = 0

while (dist > 0.1)
{
    gradient_a = Calc_Distance(a+1, b) - Calc_Distance(a-1, b)
    gradient_b = Calc_Distance(a, b+1) - Calc_Distance(a, b-1)

    have we gone past it?

    if sign(old_gradient_a) != sign(gradient_a) then
        a -= speeda * old_gradient_a / (gradient_a - old_gradient_a)
        speeda = 0
    else
        speeda += ga

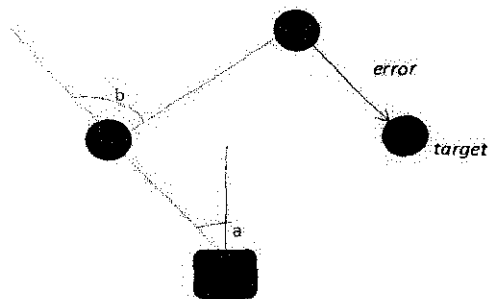
    if sign(old_gradient_b) != sign(gradient_b) then
        b -= speedb * old_gradient_b / (gradient_b - old_gradient_b)
        speedb = 0
    else
        speedb += gb

    move
    a -= speed_a
    b -= speed_b

    dist = Calc_Distance(a, b)
}

```

The algorithm will always calculate the gradient and rotating the appendages until the tip gets close enough to the target. However, the tip would not reach the target precisely and would always take a longer time to calculate the positions due to its iterative behavior.



**Figure 6: Simple two-jointed appendage with target**

### 3.1.1.3. INVERSE JACOBIAN

This approach leans heavily with the concept of vector calculus. The Jacobian matrix is the matrix of first-order partial derivatives with reference to other vectors. Formula 6 shows the Jacobian matrix of partial derivatives to relate a change in any component of  $\mathbf{x}$  to a change in any component of  $\mathbf{f}$ . Jacobian matrix  $J(\mathbf{f}, \mathbf{x})$  shows how each component of  $\mathbf{f}$  varies with respect to each joint angle  $\mathbf{x}$ .

$$J(\mathbf{f}, \mathbf{x}) = \frac{d\mathbf{f}}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_N} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \frac{\partial f_M}{\partial x_1} & \dots & \dots & \frac{\partial f_M}{\partial x_N} \end{bmatrix} [6]$$

The amount of incremental change on each iteration is defined by the relationship between the partial derivatives of the joint angles,  $\theta$ , (represented with  $\mathbf{x}$ ) and the difference between the current location of the end effector,  $\mathbf{Y}$ , and the desired position to reach target,  $\mathbf{Y}_d$ .

Below is the sample coding used for the system in order to inverse the Jacobian matrix:

```
MatrixXf LinkedStructure::pseudoInverse()
{
    //Computing pseudoinverse
    MatrixXf j = jacobian();
    MatrixXf jjtInv = (j * j.transpose());
    jjtInv = jjtInv.inverse();

    return (j.transpose() * jjtInv);
}
```



The algorithm of inverse Jacobian:

- Compute the instantaneous effect of each joint
- Uses linear approximation for the motion
- Finds and compute the linear combination to bring end effector to goal position
- Once a step is taken, solution needs to be recomputed

The effect of using this algorithm is that the movement will be in curved lines/paths, not straight. This avoids jaggy form of movement and it looks smooth on the screen.

When using Inverse Jacobian, the end effector would be really sensitive to the target position and hence allowed the appendage to move accordingly to reach target. The algorithm was chosen and IK was programmed into the appendage appendage based on the tutorial by Alexandros Dermenakis in his website (<http://alexandrosdermenakis.com/tutorials>).

### **3.2. PAST RESULTS FOR RESEARCH**

Even though most of the animation involving 3D was developed using CCD, it was not considered for this research. The reason was its complexity as an IK solver implementation. Hence, it would not be feasible to incorporate the whole algorithm in such short developmental time given by the university.

The triangulation algorithm has been applied a simple OpenGL application in C++ to gauge its performance against CCD algorithm. Mukundan and Muller-Cajar (2007) found that by moving target to 1000 new positions, CCD reached the targets by 92.97% with a mean of 8.727 iterations. On the other hand, triangulation reached each target on the first iteration as hypothesized.

On the other hand, the faster gradient following algorithm has not used in research since it was developed by Elias (2004) to introduce novices to IK animation algorithm. This research aims to portray the reliability of this algorithm in terms of judging how close (calculated in units) the end effector could move to the target.

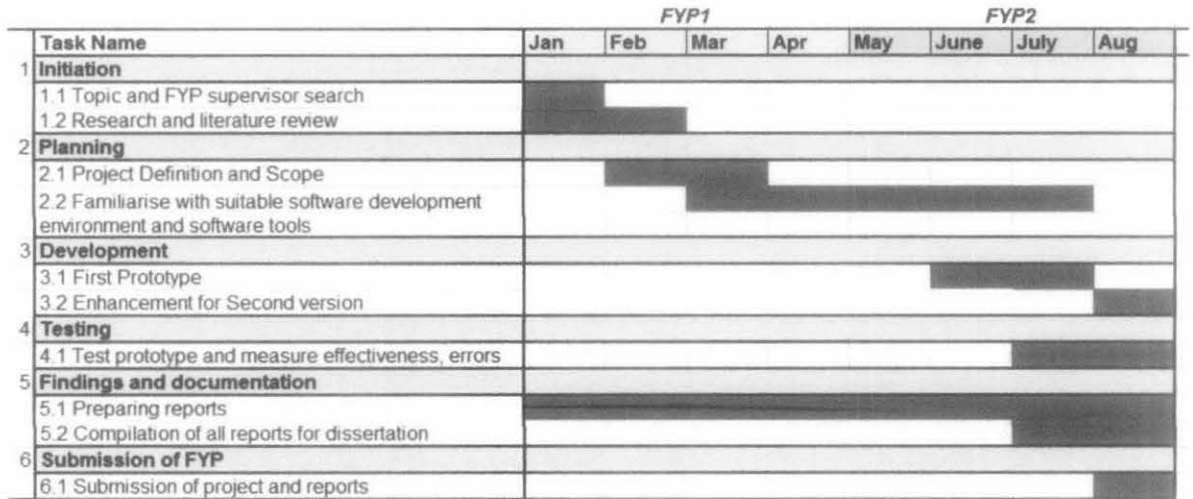
Both of the algorithms were not efficient enough to show the movement of random target being reached by the end effector of the robotic appendage. Hence, the inverse Jacobian was used to develop this program since it was more efficient – and it is able to reach target at almost all of the time.

### **3.3. INSTRUMENTATION**

For the purpose of this research, OpenGL libraries were used for the graphical and GUI development. For GUI, GLUT library was fundamental in creating the functions such as buttons, spinner buttons, and rotating ball for viewing purposes. The output would be a Win32 console program which was written with C++ language. C++ was chosen as it is a language that is suitable for system programming.

### **3.4. PROCEDURE**

To ensure timeliness in completing the project, Gantt chart was used to track the progress of the research study and implementation.



**Figure 7: Gantt chart for FYP1 and FYP2**

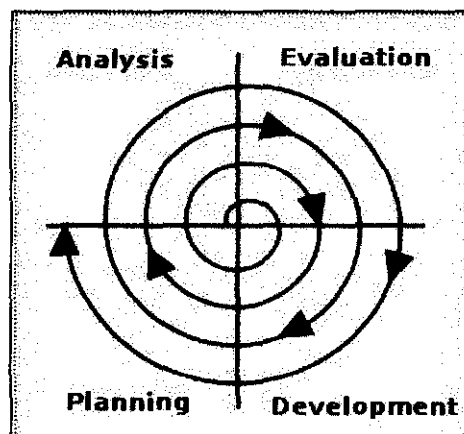
The key milestones that the author needs to achieve this semester will be as per below, the ones highlighted in yellow were already met:

Week 7 (July, 2011)	Submission of Progress Report to supervisor
Week 11 (August, 2011)	Pre-SEDEX
Week 12 (August, 2011)	Submission of Dissertation for external examiner
Week 13 (August, 2011)	Dissertation Submission and Viva presentation
Week 14 (August, 2011)	Technical Report Submission

**Table 1: Key Milestones for FYP2**

In the initial stage of the project development, the prototype for the simulation program was communicated to supervising lecturers for further review. As such, communication strategy was practiced. Crucial steps were first to outline the objectives for the communication strategy (Argenti, 2008). The objectives must portray the vision of this project clearly. Clear objectives would ensure the scope of the project would not go off the track. As the research progressed, it became important to communicate this project accordingly to the lecturers through the project presentation, interim, dissertation and technical reports. Feedback received throughout the development of this project was accounted for as serious contemplation for future improvements. Among the feedback received were regarding the functions available through the project itself as well as the research methodology.

The research was conducted using the spiral development whereby the prototype was developed in an iterative manner with incremental changes around each spiral (Planning, Development, Analysis and Evaluation). System was developed gradually, while going through all phases, until it reached the requirements. In this research, it has 3 spirals of developmental phases to reach the first working prototype.



**Figure 8: Spiral Model**

Prior to developing the project, the Activity, Sequence, Class, Data Flow and Context diagrams were constructed for graphical representation of the system [Appendix 1-5]

### **3.5. EFFICIENCY ANALYSIS**

The 3D program application was designed to measure the ability of the robotic appendages to reach certain target on the simulation environment while user manipulated the angles, for purpose of experimental results- similar to that of a protein backbone's ideal values of dihedral angles. The performance of the system would be measured based on the successful attempts by the robotic appendages to reach and grab each of the targets shown. The higher the ratio of success, the system would be deemed as efficient enough.

$$\text{Assume a, the ratio} = \frac{\text{Number of times robot reach target}}{\text{Number of times target is shown on screen per period of time, t}}$$

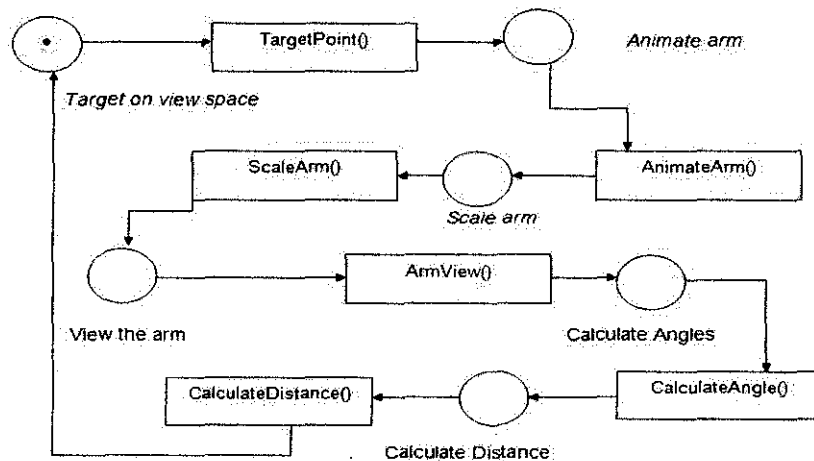
### **3.6. MODELING**

For this work, the system was modeled with boundaries in the experimental frame. In the initial plan of the research, two algorithms would be implemented by the system whereby user is able to select either algorithm to simulate the robotic appendage. Modeling was first designed with Petri Net models involving the two algorithms, namely Triangulation and Faster Gradient Following as this was in the initial plan of

developmental phase. Later on, only one algorithm (inverse Jacobian) was chosen for the implementation due to time constraint.

Below are the inputs, outputs and constraints involved in the Petri Net models:

- *Inputs: Target position, type of algorithm (either Triangulation or Faster Gradient Following)*
- *Outputs: Movement of the robotic appendages, iteration value, distance target from end effector, joint angles*
- *Constraints: Rotation respective to joint angles, appendage lengths, fixed robotic base, space, and scale*



**Figure 9: Simplified Petri Net model for one algorithm**

Since this project involves a visualization system to portray how the robotic appendage moves according to the Inverse Kinematics Algorithm, hence it will require the interaction from user.

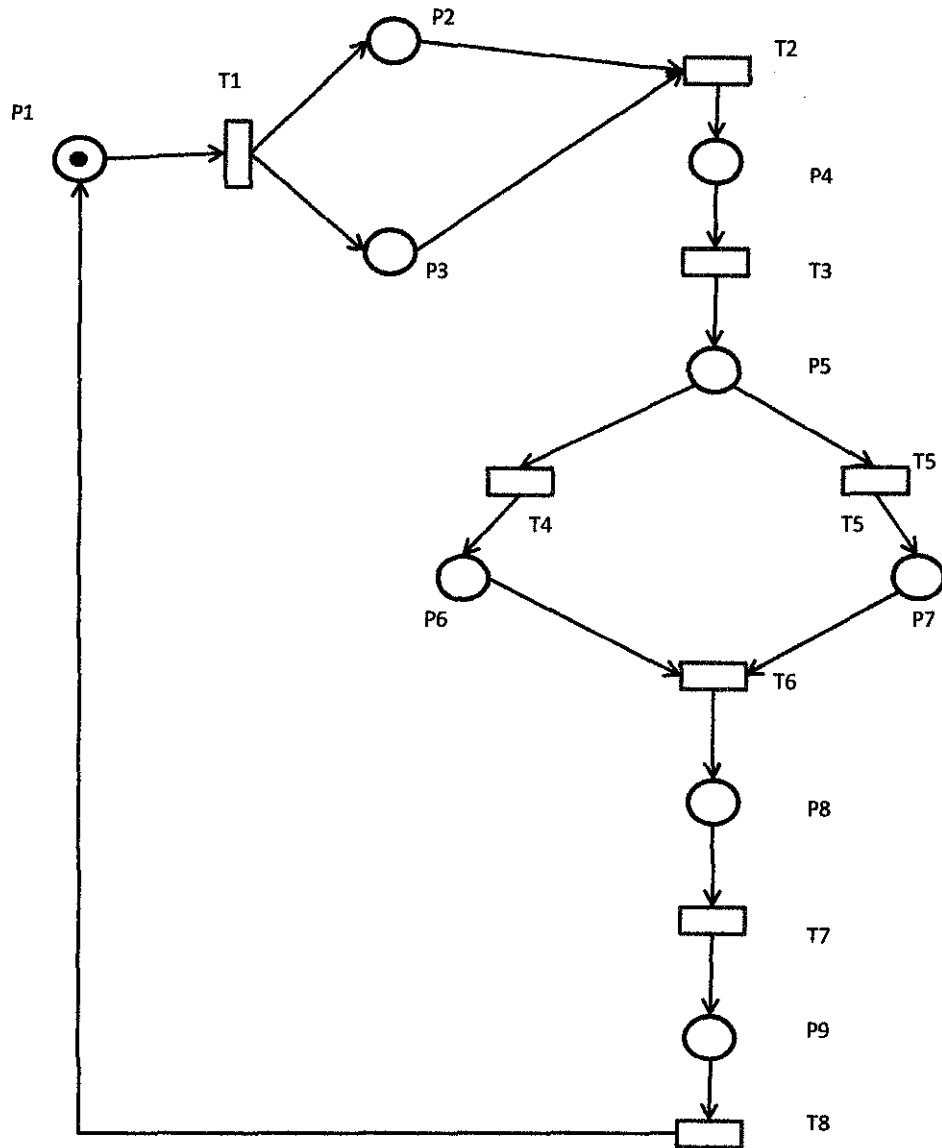
User will first click on type of algorithm (either Triangulation or Faster Gradient Following) before proceeding to determine where the target should be on the workspace/viewport. The constraint is only one algorithm to be chosen at one time via the checkbox button. Once target spot is clicked by the user, user needs to click on the Animate button. The robotic appendage will move accordingly via the chosen Inverse Kinematics algorithm towards the specified target on the viewport.

At this point, the user can actually scale the appendage through the spinner button on the graphical user interface.

User can also rotate the appendage in the X, Y and Z perspective. This can be done by drag and rotate button on the graphical user interface as well.

In the system itself, the distance of the tip of the appendage to the target will be calculated prior to reaching the target. Aside from that, angles of the three limbs would also be taken into calculation in order to show Inverse Kinematics movement of the robotic appendage on the screen.

After this process, the loop can be repeated for user to choose a new target point on the viewport once more with or without changing the type of algorithm.



**Figure 10: Petri Net model for two algorithms**

Places:

P1: Program is ready, robotic appendage in the display mode on the viewport



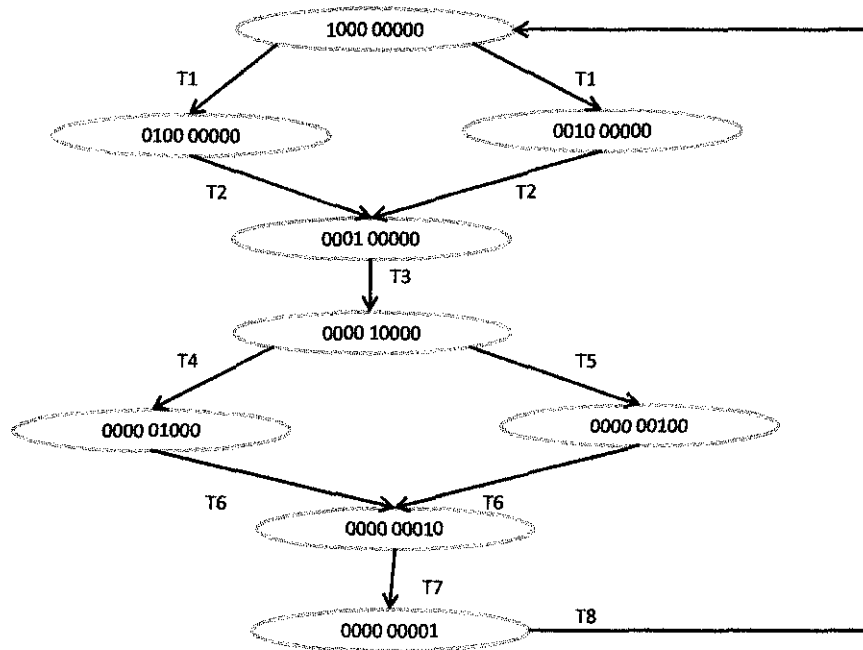
- P2: Triangulation algorithm is executed
- P3: Faster gradient following algorithm is executed
- P4: Target spot is shown the viewport
- P5: Robotic appendage moves according to the chosen algorithm
- P6: Robotic appendage grows or shrinks according to selected scale by user
- P7: Robotic appendage is rotated in X, Y or Z perspective
- P8: Distance from tip of appendage to the target is calculated
- P9: Angles of the three limbs are calculated via the algorithm in order to reach target

Transitions:

- T1: Select algorithm
- T2: Select target spot on the viewport
- T3: Animate button
- T4: Scale appendage
- T5: Rotate appendage via View button
- T6: Robotic appendage on the viewport, operation modules on standby
- T7: Distance from tip to target is used for the calculation in the algorithm
- T8: Angles are calculated in order to rotate appendage towards the target. Robot appendage moves to target

The initial Petri Net model for two algorithms was then analyzed with state reachability and its boundedness, safeness, and liveness

- State Reachability (with Reachability tree)



**Figure 11: State Reachability Graph**

The reachability graph shows that the Petri Net is reachable from its initial marking point, P1 to the last transition in the process, calculated angles shown on the graphical user interface.

There is a sequence of firings from P1 to T7, whereby the process can be repeated again through T8 to the initial marking, P1.

- Boundedness, Safeness, and Liveness

The Petri Net is bounded since the number of tokens of each place does not exceed a finite number, i.e. only one token per each place at a time.

The Petri Net is also safe as it is 1-bounded where every marking has only one token at a time. When the net itself is bounded, it is guaranteed that there will be no overflows in the buffers, no matter how the firing sequence may be.

The Petri Net is also live in the sense it has the complete absence of deadlock situations. In this Petri-Net each transitions:

*T1: L-1 live, fire at least once in the firing sequence*

*T2: L-1 live, fire at least once in the firing sequence*

*T3: L-1 live, fire at least once in the firing sequence*

*T4: L-1 live, fire at least once in the firing sequence*

*T5: L-1 live, fire at least once in the firing sequence*

*T6: L-1 live, fire at least once in the firing sequence*

*T7: L-1 live, fire at least once in the firing sequence*

*T8: L-1 live, fire at least once in the firing sequence*

Hence, in overall for the Petri-Net, it is considered to be live or L-4 live since it is L1-live for every marking in the transitions from initial marking. A live Petri-Net ascertains that there will be no deadlocks regardless of the firing sequence.

The revised Petri-Net was measured also with the inverse Jacobian algorithm (the only algorithm used for the system development) and the result was similar: it is bounded, safe, and live or L-4 live.

### **3.7. SOFTWARE TESTING**

Apart from efficiency analysis, the application program would be tested under controlled conditions to ensure it performs as expected. The testing would also help to validate the correctness of the program and specify errors if they exist (such as software bugs).

The aim of software testing was not only to look for the errors but also to gauge the potential or hidden problems in order to minimize the risk of system failure.

The reliability of the software would be tested based on how many times the system crashed or froze while the user was using the system for a controlled period of time.

Aside from that, it would be useful to test the errors of the system in which the robotic appendages did not reach the target (the tip was not at all near to the target point). This was because that even sometimes the algorithm implementation did not cause the appendage to reach the target point. The system's implementation of the algorithm could be flawed as well, leading to erroneous results. From the samples tested over a period of time, the accuracy of the system can be determined as well.

Next, the usability testing test would be performed on the system as well. The following questions were first drafted for the group of testers:

- 1) How much time taken by the tester to understand and use the system?
- 2) How many mistakes tester make while using the system? Wrong buttons clicked and ex cetera?
- 3) How quick the tester becomes familiar with the system?

4) How fast user recalls system's functions and buttons the next time the tester uses the system?

5) How does the tester feel when he/she is using the system?

In order to generate precise data sampling, a System Usability Scale (SUS) was used to identify the usability performance of the system.

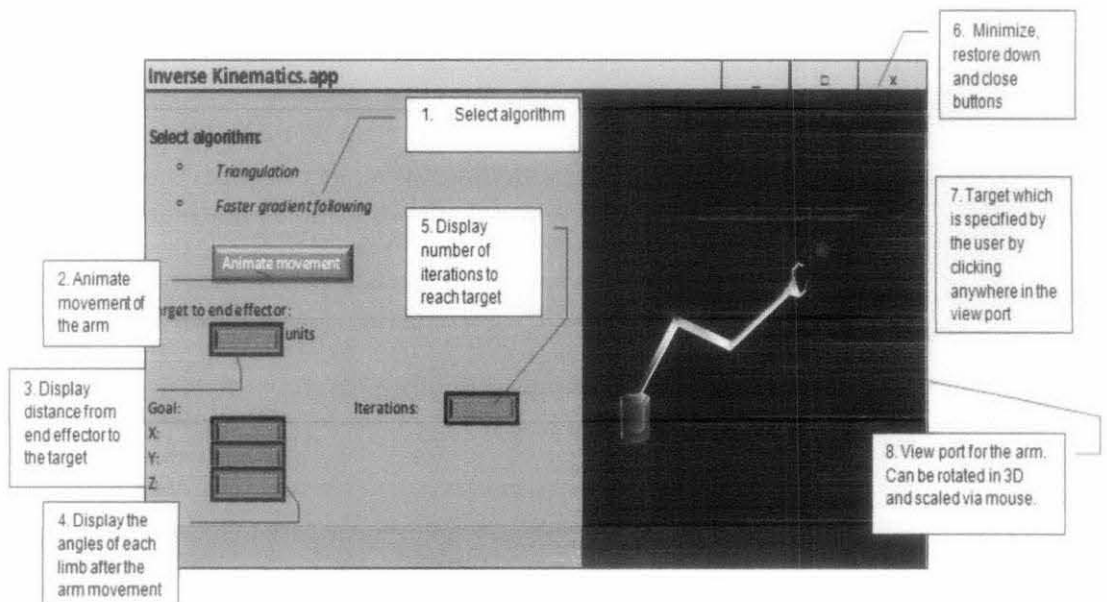
## CHAPTER 4

### RESULTS AND DISCUSSION

The visualization system implemented one existing algorithm, using the inverse Jacobian and Euler mathematics. Usage of the algorithm will enable user to visualize the movement of the appendage with respect to the target itself.

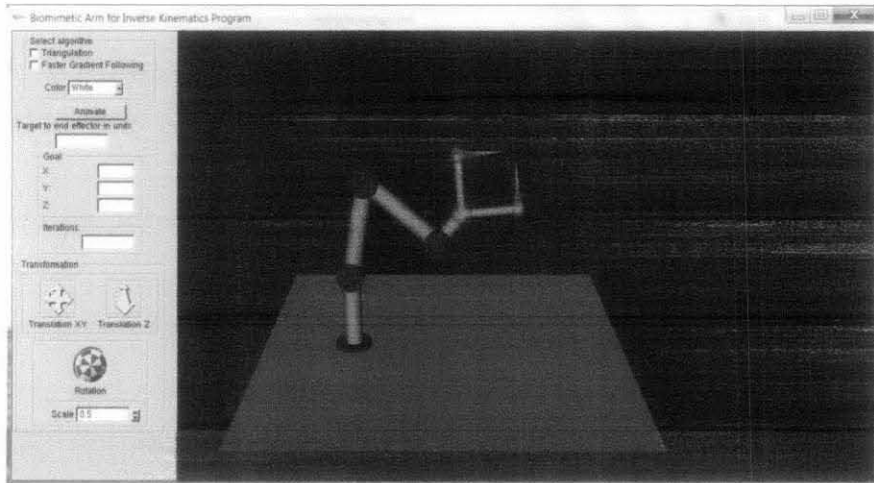
#### 4.1. PROTOTYPE

Below is the first suggested graphical user interface prototype design for the visualization system.



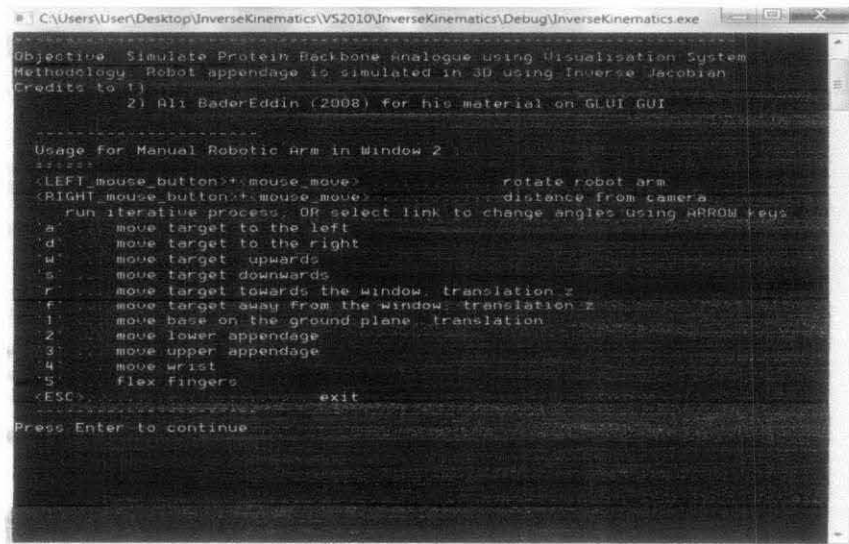
**Figure 12: Rough GUI design**

As per initial plan, below image is the GUI for the system with two algorithms implemented.



**Figure 13: GUI using OpenGL**

The finalized GUI and system of the biomimetic inverse kinematics robotic appendage are as shown below (Welcome Screen with instructions).



**Figure 14: Welcome Screen**

When the user presses the Enter keyboard button, two windows will appear on the screen [see Appendix 8 for more examples].

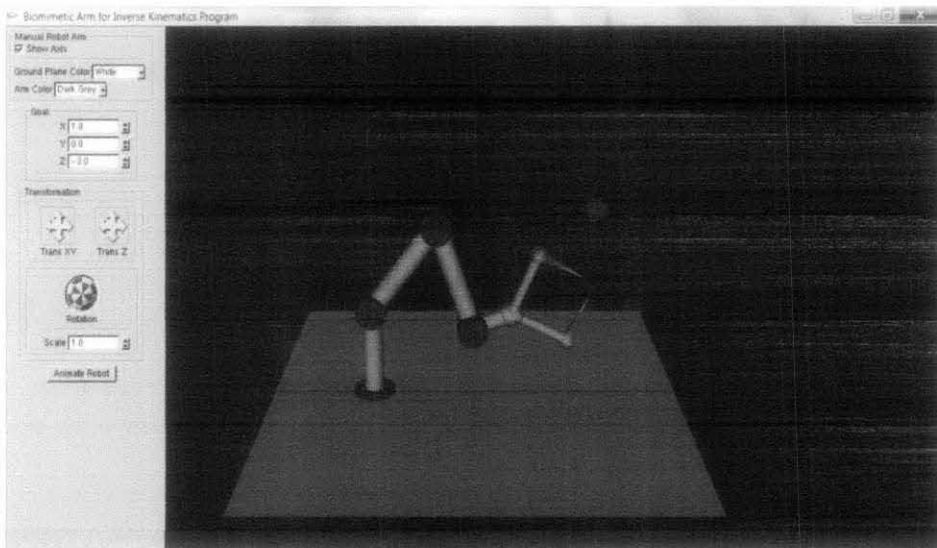


Figure 15: *Window for the Biomimetic Robotic Appendage*

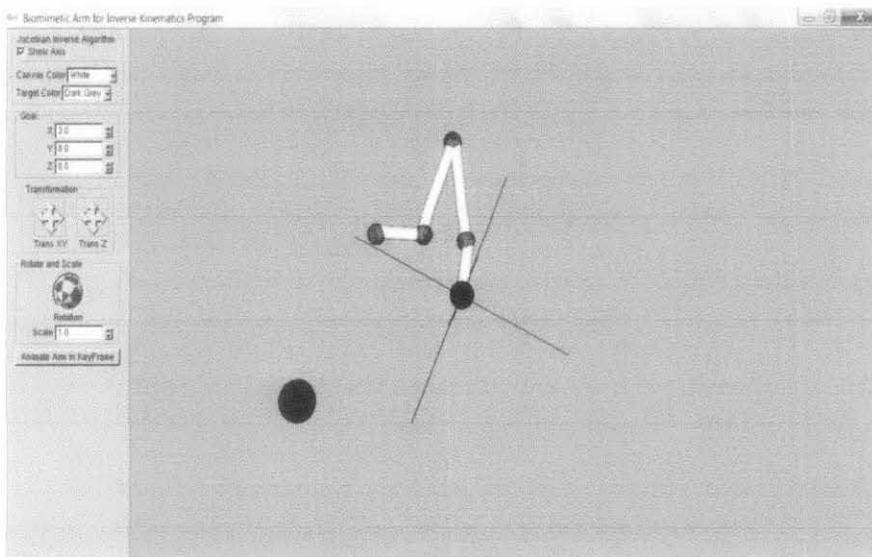


Figure 16: *Window for moving appendage and random target*



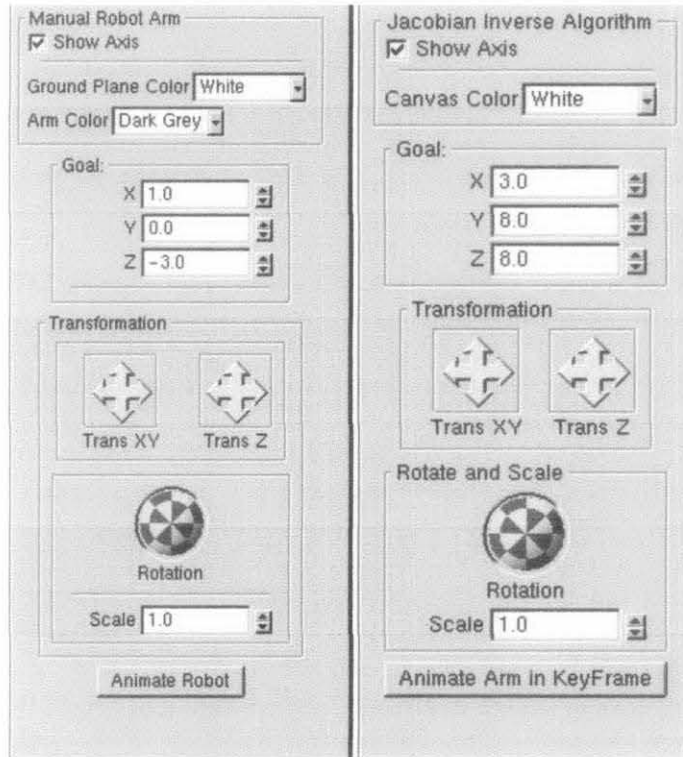


Figure 17: Functions in both windows

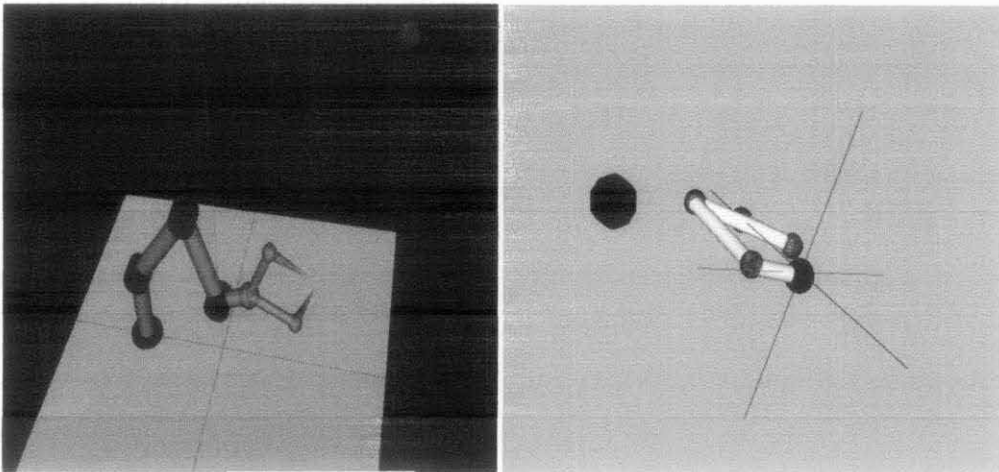


Figure 18: Rotated views in both windows

## **4.2. DATA GATHERING AND ANALYSIS METHOD**

Data gathering was mainly conducted via observation of the robotic appendages performance via the visualization system. Results were then recorded accordingly prior to interpreting the results and drawing appropriate conclusions.

Analysis was based on the following attributes:

1. Correctness of the appendage: Identify the ideal path of the appendage movement and measure the deviations, if any. Did it move to the target?
2. Do user usability testing based on the graphical user interface as well as the functions on the screen. Are the user satisfied with the interface and the system in overall from human computer interaction perspective?

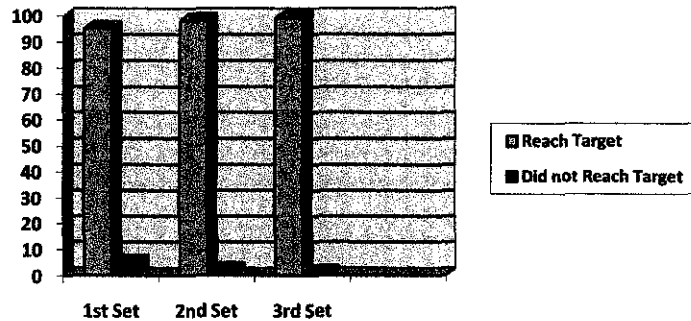
## **4.3. RESULTS AND ANALYSIS**

### **4.3.1. CORRECTNESS OF THE APPENDAGE MOVEMENT**

The appendage was simulated for 100 times where position of the random target will be different for each time. The end effector was observed with respect to the random target positioning. The result was the end effector was very sensitive to the random target movements- it will always follow the target's movements.

For only 5 times whereby the robotic appendage stopped moving (due to system's bugs) in the period of testing the end effector did not reach the random target. Due to this the system is debugged and the tests were conducted for 2 more sets of 100 times.

Results were with 98 times and with 99 times of reaching target.



**Figure 19: Correctness of the Appendage**

#### 4.3.2. EFFICIENCY OF THE ROBOTIC APPENDAGE

Assume  $a$ , the ratio =  $\frac{\text{Number of times robot reach target}}{\text{Number of times target is shown on screen per period of time, } t}$

The biomimetic robotic appendage was simulated for 100 times in which the target was moving randomly for 100 times as well. Period of time was 5 minutes (5x 60= 300 seconds).

Based on the simulation tests, the robotic appendage was able to reach 80 times out of 100 times (the remaining 20 times was not counted as the robotic limbs were colliding with each other even though the end effector touched the target. Robotic appendage was not moving normally).

Hence the ratio,  $a$ :  $80 / (100) = 0.80$ , which indicates the robotic appendage can be considered as near to highly efficient ( $a=1$ ).

### **4.3.3. USABILITY TESTING: SYSTEM USABILITY SCALE**

System Usability Scale (SUS) is a simple ten question scale system used to portray global view of the usability of a particular system. The feature of this scale system is almost similar to a Likert scale where respondent will indicate degree of agreement or disagreement. SUS was used as part of the usability testing program at Digital Equipment Co. Ltd., United Kingdom (Brooke, 1996).

10 respondents were chosen randomly to use the system prototype without much briefing or discussion taken place. After using the system for 5-10 minutes, respondents were asked to record their immediate response to each question on the SUS form [Appendix 6]. Respondents were asked to mark each question and if they could not answer any of the questions, they were informed to mark the centre point of the scale.

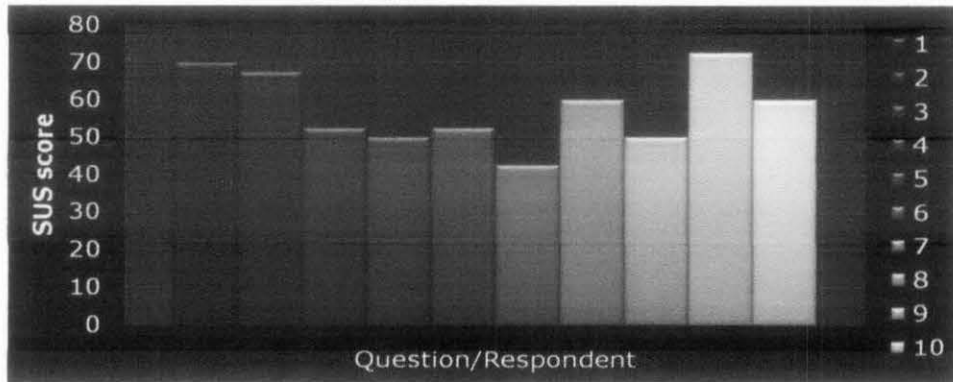
The SUS results will yield the overall usability of the biomimetic system. All questions are crucial to determine the usability rating of the system; none can be meaningful on its own. SUS score system is calculated by summing the score contributions of all questions:

0. Score contribution ranges from 0 to 4.
1. Questions 1, 3, 5, 7, and 9: Score contributions are calculated by the scale position (from Strongly Disagree, 1 to Strongly Agree 5) minus 1.
2. Questions 2, 4, 6, 8, and 10: Score contributions are calculated by 5 minus the scale position.

The sum is then multiplied with 2.5 for the overall usability value.

System Usability Scale											
Question/Respondent	1	2	3	4	5	6	7	8	9	10	SUS Score
1	5	1	5	5	3	3	3	1	3	1	70
2	3	2	5	4	5	2	4	2	4	4	67.5
3	3	3	3	4	3	3	4	3	4	3	52.5
4	5	3	3	5	5	5	5	5	3	3	50
5	4	3	4	4	3	3	2	2	3	3	52.5
6	5	5	4	5	5	5	4	5	4	5	42.5
7	3	4	3	2	3	1	3	1	4	4	60
8	3	3	3	3	3	3	3	3	3	3	50
9	4	2	4	4	4	2	4	1	5	3	72.5
10	4	2	2	3	5	1	3	1	4	2	60
<b>Average</b>											<b>57.75</b>

**Table 2: SUS result**



**Table 3: SUS result in graphical form**

Questions 1, 3, 5, 7, and 9 indicate whether the system is positively usable from the respondent's perspective. Questions 2, 4, 6 and 8 indicate the negative feeling of the respondent after using the system.

Result of the SUS rating is 57.75/100.00. Most of the users found the system is usable in general but they would like the graphical user interface to be improvised.

#### **4.4. DISCUSSION**

The objectives of this research are to:

- To create a visualization system for the biomimetic robotic appendages
- To explain and predict the performance of robotic application via the system

Based on the results received from the testing conducted on the system, both of the objectives are achieved successfully. A visualization system comprising of two windows has been designed and implemented with the inverse Jacobian algorithm. The system is able to visualize and simulate the movement of a robotic appendage rotating on a spherical joint towards random target on the screen.

Aside from that, the performance of the system is considered good in terms of able to perform the following functions:

- Rotating both of the appendages in both windows
- Spinner buttons to move the target in X, Y, Z directions
- Moving the appendage towards the target in inverse kinematics
- Appendage is modeled according to a protein backbone
- User can change the colors of the biomimetic robotic appendage, ground floor and background

Only the precision of the robotic appendage has not been tested in this system.

- How close the end effector to the target? Assume 0.5 units as the maximum radius value to the target. Once end effector reaches  $< 0.5$  units from target, it is considered target is successfully reached.

Involve test tries with 100 different positions of the target, and measure in statistical sampling mod, mean and median values. Develop range for precision:

0.09 – 0.5 units	Quite precise
2.01- 0.08 units	Very precise

**Table 4: *Range of robotic precision***

Precision of the appendage can be tested in the future studies of the project.

#### **4.5. POTENTIAL APPLICATIONS IN ROBOTICS**

- A. Medical field especially in surgical operating room – used alongside Da Vinci and ZEUS robotic appendages. It is especially appropriate for miniscule incision such as keyhole surgery. Also this application can be used in nanorobotics involving surgery.
- B. Manufacturing to substitute human capabilities in repetitive motions such as welding, pick and place products on conveyor belt, and ex cetera.
- C. Space Research to inspect hard-to-reach areas in the outer space equipped with camera and for the safety of the astronauts. Current appendage used is Canadarm.
- D. Provides dexterity in movements in harsh environments such as repairing drilling pipes underground the sea, cleaning hazardous and radioactive wastes and ex cetera.
- E. Education to better understand the protein loop closure problem solved with IK.
- F. Implantation of robotic appendage for limbless or crippled persons.
- G. Circuital and Electronics where circuits can be connected without human intervention via the biomimetic protein nanorobotics.
- H. Biotechnology research and implementation whereby nanorobotics used for monitoring and detection of miniscule results involving living microorganisms or as sensors
- I. Mathematical model or computational model inspired by the protein movement in network system (mobile technology, internet, and ex cetera) to process information.



## **CHAPTER 5**

### **DELIMITATIONS, RECOMMENDATIONS AND CONCLUSION**

#### **5.1. DELIMITATIONS**

There are several delimitations worth noting about this research. The research will not focus on improvising any specific applications of the robotic appendage, either in the non-domestic or domestic environments. However, the research does envisage the usage of the robotic appendages for current robotic applications existing in the industry as long as the physical structure and its movements are suitable for the relative environment. The next delimitation is the scope of the research does not include any specific protein dihedral angles used as the manipulating variables for the robotic appendages modeling.

#### **5.2. RECOMMENDATIONS**

With this, the research could be further enhanced in the future to include it in specific environment and measure its effectiveness (movements and dynamism) as compared to existing robotics design. Aside from that, the system can be implemented as a comparison of two algorithms to evaluate the efficiency and effectiveness of both in simulating the appendage.

Additionally, it would be an improvement to this research if better IK algorithm could be used to employ optimization of performance with regards of the robotic appendages movements and specific flexibility based on different protein backbone structure modeling

## REFERENCES

- Adcock, S.A. & McCammon, J.A. (2008). Molecular dynamics: Survey of methods for simulating the activity of proteins. *Howard Hughes Medical Institute. Chem Rev.* 106 (5), 1589-1615.
- Argenti, P.A. (2008). Corporate Communication. Fifth edition. Boston: McGraw-Hill.
- Artemiadis, P.K., Katsiaris, P.T., Kyriakopoulos, K.J. (2010). A biomimetic approach to inverse kinematics for a redundant robot appendage. Springer Science+ Business Media, LLC.
- Berkholz, D.S., Shapovalov, M.V., Dunbrack, R.L., & Karplus, P.A. (2000). Conformation dependence of backbone geometry in proteins. *Cell Press :Structure* 17, 1316–1325.
- Bill, D.T. (2003). Simulation for experiential learning. Retrieved from [www.liquid.group.com](http://www.liquid.group.com) at 2011, February 10.
- Brooke, J. (1996). "SUS: a "quick and dirty" usability scale".
- Canutescu, A.A., & Dunbrack, R.L. (2003). Cyclic coordinate descent: a robotics algorithm for protein loop closure. *Protein Science, Vol. 12*, 963–972.
- Cohen, Y.B., & Breazeal, C. (2003, March). Biologically inspired intelligent robotics. *Proceedings of the SPIE Smart Structures Conference San Diego, CA*, 5051-02.

- Coutsias, E.A., Seok, C., Jacobson, M.P., & Dill, K.A. (2004) . A Kinematic View of Loop Closure. *Wiley Periodicals J Comput Chem: Vol. 25*, 510-528.
- Coutsias, E.A., Seok, C., Wester, M.J., & Dill, K.A. (2006). Resultants and Loop Closure. *International Journal of Quantum Chemistry, Vol. 106*, 176-189.
- Creighton, T.E. (1993). *Proteins: Structures and molecular properties*. W. H. Freeman and Company, New York, 2nd edition.
- Czyzewski, A.M., & Barron, A.E. (2007, November). Protein and peptide biomimicry: Gold-mining inspiration from nature's ingenuity. *AIChE Journal 2008, Vol. 54*, 1-7.
- Endou, A., Carpio, C.A.D., Qiang, P., Ichiishi, E., Tsuboi, H., Koyama, M., Hatakeyama, N., Takaba, H., Kubo, M. (2006). Robotic path planning and protein complex modeling considering low frequency intra-molecular loop and domain motions. *Genome Informatics. Vol. 17, No.2* , 270-278.
- Elias, H. (2004). *Inverse Kinematics - Improved Methods*. Retrieved from [http://freespace.virgin.net/hugo.elias/models/m\\_ik2.htm](http://freespace.virgin.net/hugo.elias/models/m_ik2.htm)
- Fay, A.L. & Snoeyink, J. (2003). Backbone motion by inverse kinematics for protein design. *Biogeometry News*. Retrieved from <http://biogeometry.duke.edu/newsletter/issues/> at 2011, February 10.
- Grochow, K., Martin, S.L., Hertzmann, A., Popovic, Z.(2004). Style-based inverse kinematics. *ACM Trans. Graph., Vol. 23, No. 3*, 522-531.

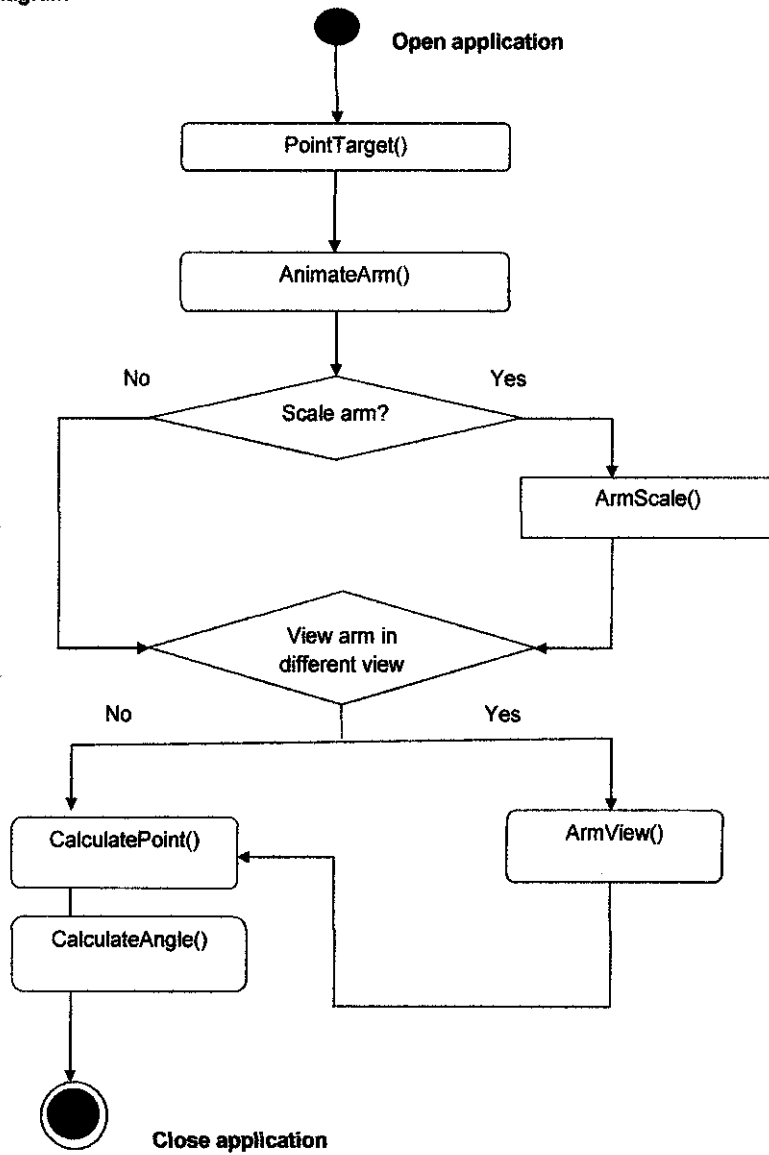
- Guanfeng, L., Milgram, R.J., Dhanik, A., & Latombe, J.C. (2006). On the inverse kinematics of a fragment of protein backbone. *10th Symposium on advances in robot kinematics*, 201-208.
- Hayashibe, M., Suzuki, N., Hashizume, M., Konishi, K., Hattori, A. (2006, July). Robotic surgery setup simulation with the integration of inverse-kinematics computation and medical imaging. *Computer Methods and Programs in Biomedicine Vol. 83*, No. 1, 63-72.
- Ho, E.S.L., Komura, T., & Lau, R.W.H. (2005). Computing inverse kinematics with linear programming. *VRST '05: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, Monterey, USA*, 163–166.
- Kavraki, L. (2007). Protein inverse kinematics and loop closure problem. *The Connexions Project, module m11613*.
- Lapresté, J.T., Jurie, F., Dhome, M., Chaumette, F. (2004). An Efficient Method to Compute the Inverse Jacobian Matrix in Visual Servoing.
- Mandell, D.J. (2009, August). Backbone flexibility in computational protein design. *Current Opinion in Biotechnology Vol. 5*, No. 4, 420-428.
- Mandell, D.J. & Kortemme, T. (2009, November). Computer-aided design of functional protein interactions. *Nature Chemical Biology. Vol. 5*, No. 11, 797-807.

- Mukundan, R. & Muller-Cajar, R. (2007, December). Triangulation: a new algorithm for inverse kinematics. *Proceedings of Image and Vision Computing New Zealand*, 181-186.
- Shenkin, P.S., Yappendageush, D.L., Fine, R.M., Wang, H.J., & Levinthal, C. (1987). Predicting antibody hypervariable loop conformation. I. Ensembles of random conformations for ringlike structures. *Biopolymers Vol. 26*, 2053-2085.
- Singh, A.P., Latombe, J.C., & Brutlag, D.L. (1999). A Motion Planning Approach to Flexible Ligand Binding. *Proc. 7th ISMB*, 252-261.
- Trivedi, D., Rahn, C.D., Kier, W.M., & Walker, I.D. (2008, September). Soft robotics: biological inspiration, state of the art, and future research. *Applied Bionics and Biomechanics Vol. 5*, No. 3, 99-117.
- Wang, L.T. and Chen, C.C. (1991). A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Trans. Robotics Automation Vol 7*, 489-499.
- Wu, Y.C., Shehu, A., & Kavraki, L.E. (2005). Modeling Protein Flexibility With Spatial And Energetic Constraints. Rice University, Houston, TX.
- Wu, Z. (2008). Lecture Notes On Computational Structural Biology. World Scientific Publishing Co. Pte. Ltd., Singapore.

# APPENDICES

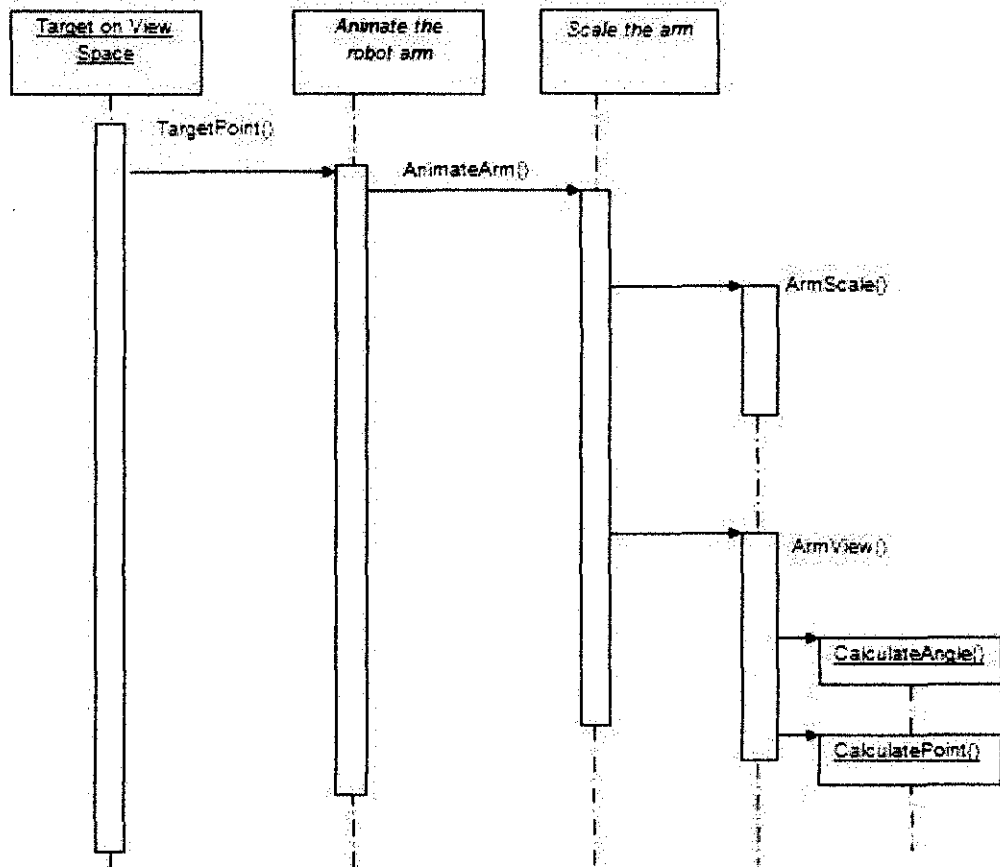
## APPENDIX 1

Activity Diagram



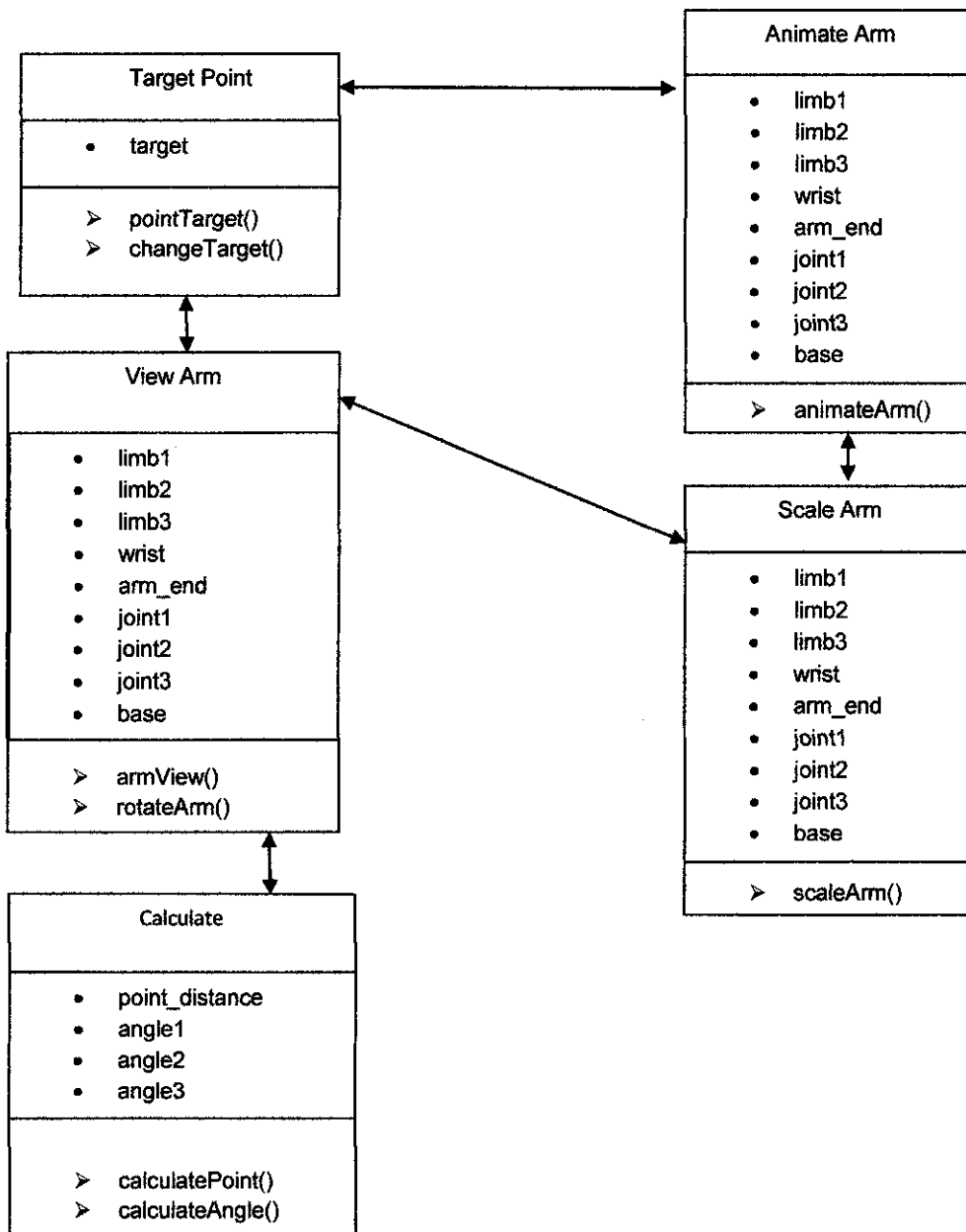
## APPENDIX 2

Sequence Diagram



### APPENDIX 3

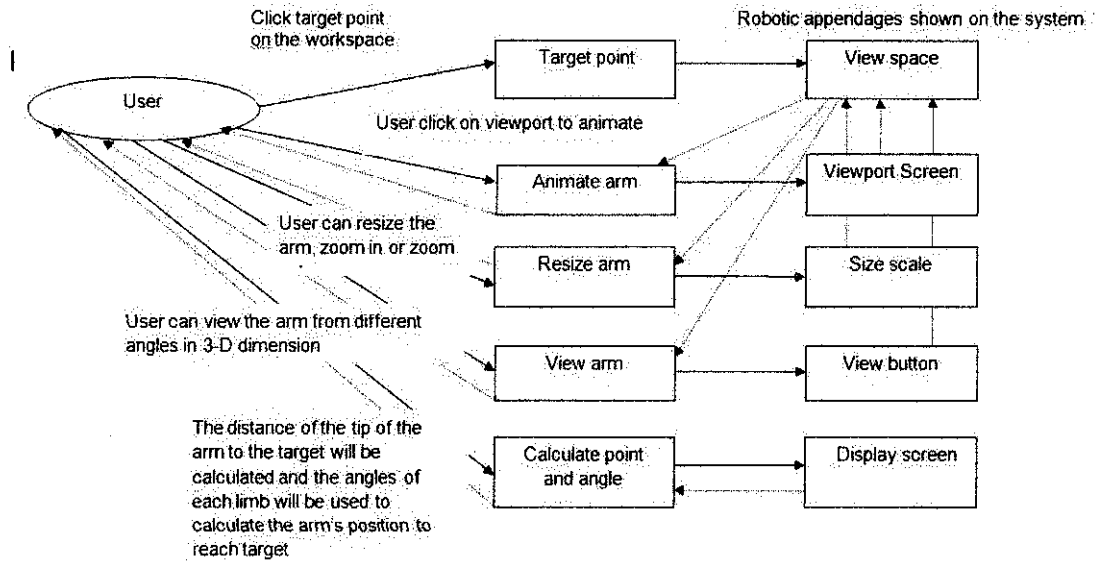
#### Class Diagram





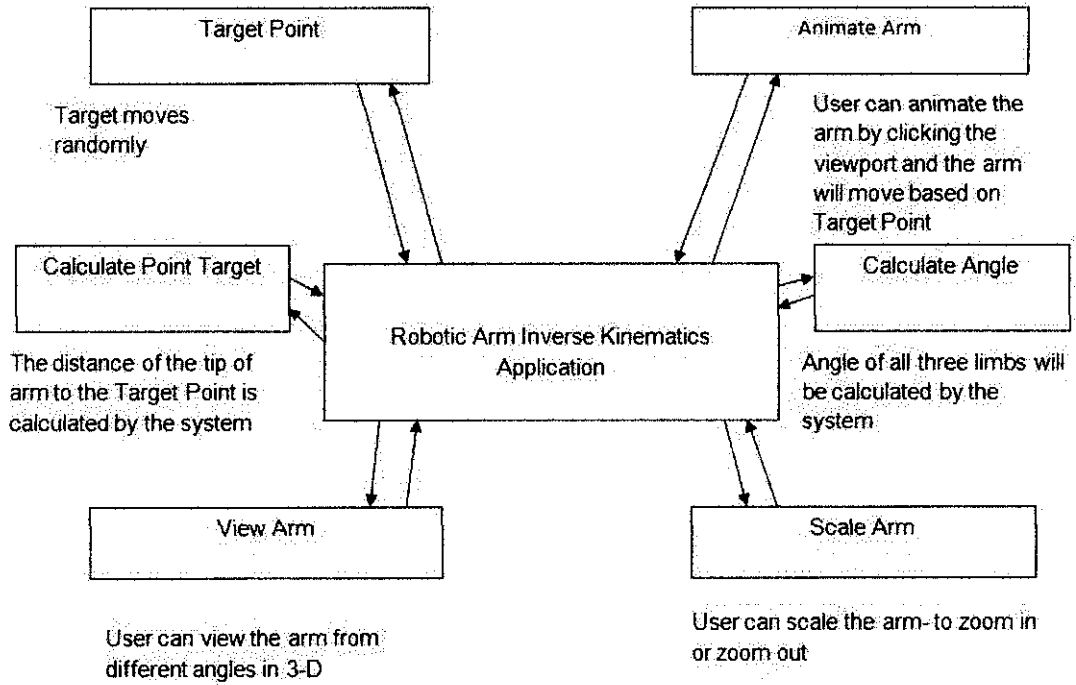
## APPENDIX 4

### Data Flow Diagram



## APPENDIX 5

### Context Diagram



## APPENDIX 6

### SUS Questionnaire

	Strongly disagree						Strongly agree
1. I think that I would like to use this system frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2. I found the system unnecessarily complex	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3. I thought the system was easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4. I think that I would need the support of a technical person to be able to use this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5. I found the various functions in this system were well integrated	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6. I thought there was too much inconsistency in this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7. I would imagine that most people would learn to use this system very quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8. I found the system very cumbersome to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
9. I felt very confident using the system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
10. I needed to learn a lot of things before I could get going with this system	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

## APPENDIX 7

### Coding Snippets

#### A. Main function:

```
int main (int argc, char **argv)
{

// GLUT initialization.
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
glutInitWindowSize(width, height);
glutInitWindowPosition (window_x, window_y);
//create first window
win1 = glutCreateWindow (window_title);
inverse();
// Register call backs.
glutDisplayFunc(display_win1);
if (full_screen)
    glutFullScreen ();
init();
setupGLUI ();
glutReshapeFunc(reshapeMain Window);
glutKeyboardFunc(graphicKeys);
glutMotionFunc(mouseMovement);
glutIdleFunc(idle);
glutTimerFunc(60, timer, 0);

glEnable(GL_COLOR_MATERIAL);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_DEPTH_TEST);
glEnable(GL_NORMALIZE);
glEnable(GL_CULL_FACE);
GLfloat global_ambient[] = { 1.0f, 1.0f, 1.0f, 1.0f };
glLightModelfv(GL_AMBIENT_AND_DIFFUSE, global_ambient);

// Enter GLUT loop.
infoMessage();
```

```

HelpMessage();
fprintf(stderr, "Press Enter to continue...");
fgetc(stdin);
win2= glutCreateWindow(window_title);
glEnable(GL_COLOR_MATERIAL);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_DEPTH_TEST);
glEnable(GL_NORMALIZE);
glEnable(GL_CULL_FACE);

// View in full screen if the full_screen flag is on
if (full_screen)
    glutFullScreen ();
init();
setupGLUI2 ();
glutDisplayFunc (display);
glutReshapeFunc(reshape);
glutKeyboardFunc(keyboard);
glutMouseFunc(MouseFunc);
glutMotionFunc(MotionFunc);

glutMainLoop();

return 0;
}

```

## **B. Function calling the link structure of the moving appendage**

```

void inverse (void){

srand ( time(NULL) );

// Constructing the linked structure by
//adding links
//for (int i = 1; i <= 4; i++)
//{
Color c = {1.0f, 1.0f, 1.0f, 1.0f};
Link *l = new Link(c);

```

```

l->mAngle = 0;
l->mLength = 8;
::l.addLink(l);
    //}
        Link *s = new Link(c);
s->mAngle = -50;
s->mLength = 15;
::l.addLink(s);
        Link *t = new Link(c);
t->mAngle = -60;
t->mLength = 15;
::l.addLink(t);
        Link *v = new Link(c);
v->mAngle = 45;
v->mLength = 8;
::l.addLink(v);
    }

```

### C. Function for GLUI

```

void setupGLUI2(){

    GLUI_Translation *trans;

    // Create GLUI horizontal subwindow (placed on bottom)
    //glui_h_subwindow = GLUI_Master.create_glui_subwindow (main_window,
    GLUI_SUBWINDOW_BOTTOM);

    // Create GLUI vertical subwindow (placed on left)
    glui_v_subwindow = GLUI_Master.create_glui_subwindow (win2,
    GLUI_SUBWINDOW_LEFT);

    //-----
    // 'Object Properties' Panel
    //-----

    // Add the 'Object Properties' Panel to the GLUI vertical subwindow
    GLUI_Panel *op_panel = glui_v_subwindow->add_panel ("Manual Robot
    Appendage");

    // Add the Faster Gradient Following Check box to the 'Object Properties' Panel

```

```

glui_v_subwindow->add_checkbox_to_panel (op_panel, "Show Axis", &axis);
// Add a separator
glui_v_subwindow->add_separator_to_panel (op_panel);

// Add the Color listbox to the 'Object Properties' Panel
GLUI_Listbox *plane_listbox = glui_v_subwindow->add_listbox_to_panel
(op_panel,
                                     "Ground Plane
Color",      &listbox_item_id, PLANE_COLOR_LISTBOX, glui_callback2);

// Add the items to the listbox
plane_listbox->add_item (1, "Black");
plane_listbox->add_item (2, "Blue");
plane_listbox->add_item (3, "Cyan");
plane_listbox->add_item (4, "Dark Grey");
plane_listbox->add_item (5, "Grey");
plane_listbox->add_item (6, "Green");
plane_listbox->add_item (7, "Light Grey");
plane_listbox->add_item (8, "Magenta");
plane_listbox->add_item (9, "Orange");
plane_listbox->add_item (10, "Pink");
plane_listbox->add_item (11, "Red");
plane_listbox->add_item (12, "White");
plane_listbox->add_item (13, "Yellow");

// Select the White Color by default
plane_listbox->set_int_val (12);

// Add the Color listbox to the 'Object Properties' Panel
GLUI_Listbox *appendage_color_listbox = glui_v_subwindow-
>add_listbox_to_panel      (op_panel, "Appendage Color",
&listbox_item_id,      APPENDAGE_COLOR_LISTBOX,
glui_callback2);

// Add the items to the listbox
appendage_color_listbox->add_item (1, "Blue");
appendage_color_listbox->add_item (2, "Dark Grey");
appendage_color_listbox->add_item (3, "Green");
appendage_color_listbox->add_item (4, "Magenta");

```

```

appendage_color_listbox->add_item (5, "Orange");
appendage_color_listbox->add_item (6, "Pink");
appendage_color_listbox->add_item (7, "Red");
appendage_color_listbox->add_item (8, "Yellow");
// Select the White Color by default
appendage_color_listbox->set_int_val (2);

//-----
// 'Object Type' Panel
//-----
GLUI_Panel *panel = glui_v_subwindow->add_panel("Goal:");
// Add the scale spinner
GLUI_Spinner *goalx = glui_v_subwindow->add_spinner_to_panel (panel,
"X", GLUI_SPINNER_FLOAT, &spherePos[0], GOALX_SPINNER,
glui_callback2);
GLUI_Spinner *goaly = glui_v_subwindow->add_spinner_to_panel (panel,
"Y", GLUI_SPINNER_FLOAT, &spherePos[1], GOALY_SPINNER,
glui_callback2);
GLUI_Spinner *goalz = glui_v_subwindow->add_spinner_to_panel (panel, "Z",
GLUI_SPINNER_FLOAT, &spherePos[2], GOALZ_SPINNER,
glui_callback2);

// Add separator
glui_v_subwindow->add_separator_to_panel (panel);

//-----
// 'Transformation' Panel
//-----

// Create transformation panel 1 that will contain the Translation controls
GLUI_Panel *transformation_panel = glui_v_subwindow->add_panel
("Transformation");

// Create transformation panel 1 that will contain the Translation controls
GLUI_Panel *transformation_panel1 = glui_v_subwindow-
>add_panel_to_panel (transformation_panel, "");

```



```

// Add the Animate Button
glui_v_subwindow->add_button ("Animate Robot", ANIMATE_BUTTON,
glui_callback2);
// Let the GLUT vertical subwindow know where its main graphics window is

// Add the xy translation control
glui_v_subwindow->add_column_to_panel (transformation_panel1, FALSE);
trans=glui_v_subwindow-> add_translation_to_panel( transformation_panel1,
"Trans XY", GLUT_TRANSLATION_XY, &TransXYZ[0] );
// Set the translation speed
trans->set_speed( 0.05f);
// Add column, but don't draw it
glui_v_subwindow->add_column_to_panel (transformation_panel1, false);

// Add the z translation control
glui_v_subwindow->add_column_to_panel (transformation_panel1, FALSE);
trans=glui_v_subwindow-> add_translation_to_panel( transformation_panel1,
"Trans Z", GLUT_TRANSLATION_XY, &TransXYZ[2] );

// Set the translation speed
trans->set_speed( 0.05f);
// Add column, but don't draw it
glui_v_subwindow->add_column (false);

// Create transformation panel 2 that will contain the rotation and spinner
controls
GLUI_Panel *transformation_panel2 = glui_v_subwindow-
>add_panel_to_panel (transformation_panel, "");

// Add the rotation control
glui_v_subwindow->add_rotation_to_panel (transformation_panel2, "Rotation",
rotation_matrix, ROTATION, glui_callback2);

// Add separator
glui_v_subwindow->add_separator_to_panel (transformation_panel2);

// Add the scale spinner
GLUI_Spinner *spinner = glui_v_subwindow->add_spinner_to_panel
(transformation_panel2, "Scale", GLUT_SPINNER_FLOAT, &scale,
SCALE_SPINNER, glui_callback2);

```

```

// Set the limits for the spinner
spinner->set_float_limits ( -4.0, 4.0 );

glui_v_subwindow->set_main_gfx_window( win2 );
}

```

#### D. Function for Display

```

// This function is called to display the scene.
void display ()
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

if (glutGetWindow() == win1) {
glTranslatef(3.0f, 0.0f, -140.0f);

glTranslatef( (GLfloat)TransXYZ[0], (GLfloat)TransXYZ[1], -
(GLfloat)TransXYZ[2] );
// Rotation using X mouse.

beta = 180.0 * xMouse;
glRotatef(beta, 0, 1, 0);
alpha = 180.0 * yMouse;
glRotatef(beta, 1, 0, 0);
// Apply the scaling
glScalef (scale, scale, scale);

// Apply the rotation matrix
glMultMatrixf (rotation_matrix);
l.draw();

glPushMatrix();

if (axis1)
// Draw a red x-axis, a green y-axis, and a blue z-axis. Each of the

```

```

// axes are ten units long.
glBegin(GL_LINES);
    glColor3f(1, 0, 0); glVertex3f(-20, 0, 0); glVertex3f(20, 0, 0);
    glColor3f(0, 1, 0); glVertex3f(0, -20, 0); glVertex3f(0, 20, 0);
    glColor3f(0, 0, 1); glVertex3f(0, 0, -20); glVertex3f(0, 0, 20);
glEnd();

glFlush();
glColor3f(0.0f, 0.0f, 1.0f);
glTranslatef(0, -targetPoint(1), targetPoint(0));
glutSolidSphere(Target[0], Target[1], Target[2]);
glPopMatrix();
glFlush();}

else
{ // Displaying Window 2

    gluLookAt(0,2,4, 0,0.5f,0, 0,1,0);
    // Apply the rotation matrix
    glmMultMatrixf(rotation_matrix);
    // Apply the scaling
    glScalef(scale, scale, scale);
    glTranslatef( (GLfloat)TransXYZ[0], (GLfloat)TransXYZ[1], -
(GLfloat)TransXYZ[2] );

    DrawGroundPlane(16);
    DrawRobotAppendage(16);

    DrawTarget();
    Sleep(5);

}
glutSwapBuffers();
}

```

#### E. Moving the target using keyboard

```

void keyboard (unsigned char Key, int x, int y)
{

```

```
if (Key>='1' && Key<='5') RobotControl=Key-'1';
if (Key==' ') change();
if (Key==27) exit(0);    // ESC

switch(Key){

case 'a' : spherePos[0] -= 0.05f;
break;

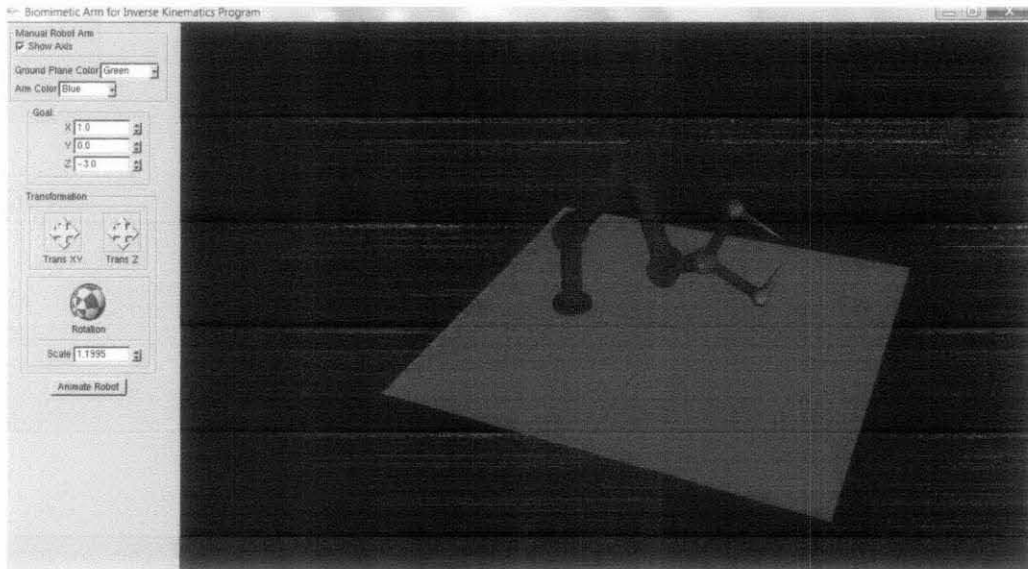
case 'd' : spherePos[0] += 0.05f;
break;
case 'w' : spherePos[1] += 0.05f;
break;
case 's' : spherePos[1] -= 0.05f;
break;

case 'r' : spherePos[2] += 0.05f;
break;

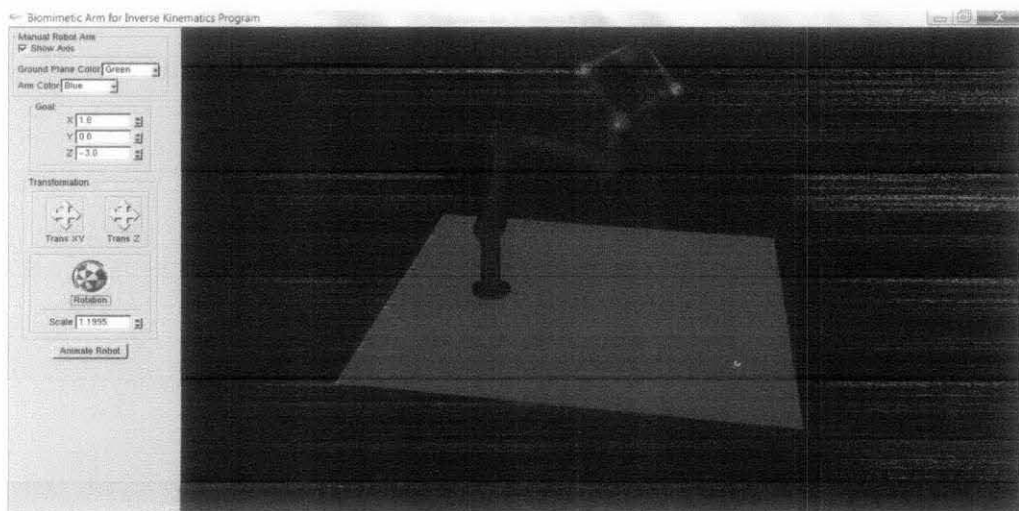
case 'f' : spherePos[2] -= 0.05f;
        break;}
}
```

## APPENDIX 8

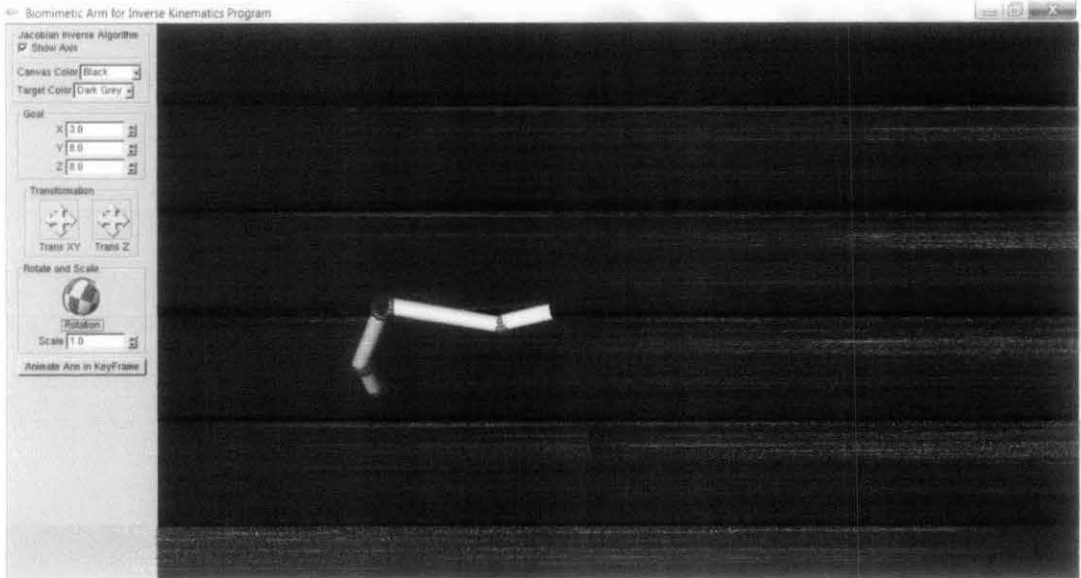
### Prototype



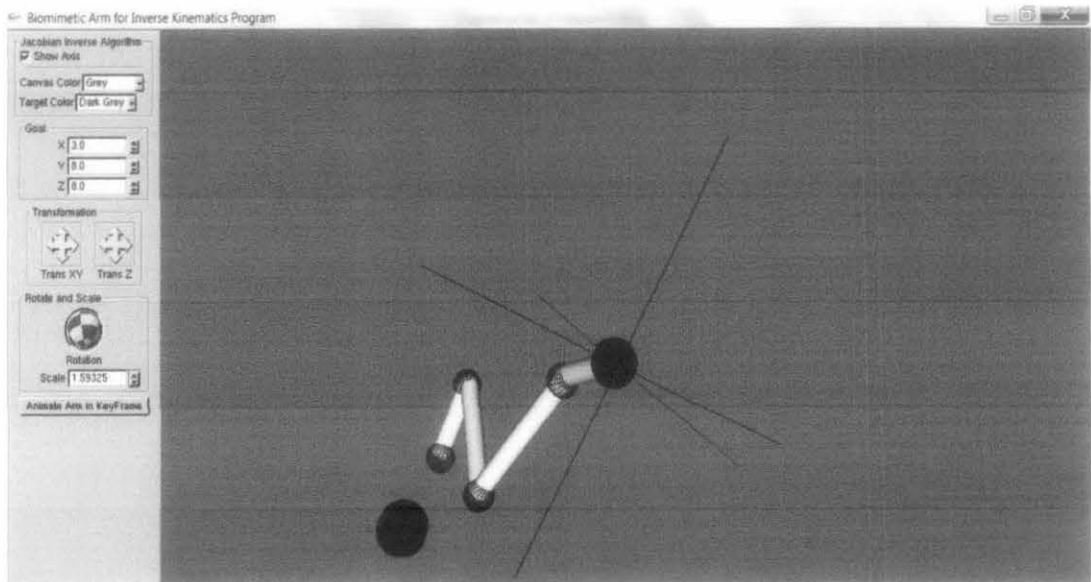
**Scaled to 1.1995 of its original size and ground plane color to green**



**Moving target with keyboard to the appendage**



**Background color changed to black, appendage moving towards target**



**Appendage scaled to 1.59, rotated with mouse**