UNIVERSITY OF KWA-ZULU NATAL

PARALLEL ROBOT DESIGN INCORPORATING A DIRECT END EFFECTOR SENSING SYSTEM

STUDENT:
Ahmed Asif Shaik – 201293525
BScEng
School of Mechanical Engineering
Durban
ashaik@csir.co.za

SUPERVISOR:
Prof. Glen Bright
Brightg@ukzn.ac.za

November 2007

Submitted in fulfilment of the academic requirements for the degree of Master of Science in Engineering at the School of Mechanical Engineering, University of KwaZulu Natal.

Preface

The author hereby states that this entire dissertation, unless specifically stated otherwise, is his own work, and has not been submitted in part or whole to any other university. This dissertation records the work completed by the author at the School of Mechanical Engineering, University of KwaZulu Natal from January 2005 to November 2007.

A. A. Shaik

Abstract

This dissertation details the development of a parallel robot with an integrated direct end effector sensing system, from concept to prototype model and includes details of research, design, simulation, construction, assembly and testing.

Current research in parallel robots is insufficient as compared to serial type machines, even though their existence has been known for some time. The reasons are the difficulty in conceptualising unique parallel mechanisms, achieving machines that are capable of high accuracy, solving their complex kinematics, dynamics and control problems. There are many advantages of parallel machines that rival the serial type, and these warrant further studies.

The second aspect of this project was the design of a direct end effector sensor system. Many existing automated multi-axis machines operate under overall 'open loop' control. The exact position in space of the end effector or tool head, for those machines, is not sensed directly but is calculated by software monitoring sensors on actuator axes.

This sensor system and robot structure was designed specifically for use in the agricultural and general food processing/packaging industries. The accuracy and repeatability of such a machine and its sensor system are in the millimetre range.

Acknowledgements

The work presented in this dissertation was carried out under the supervision of Prof. Glen Bright of the School of Mechanical Engineering, University of KwaZulu Natal. I wish to thank Prof. Bright for his unwavering support and dedicated supervision.

Additionally I owe a great deal of gratitude to the following people and organisations:

- My parents for their support and encouragement throughout my academic career. To my brother Dr. M. Z. Shaik for all the financial support and tireless motivation given without question during the toughest and most challenging year of my life.

- Mr. Mike Smith and the rest of the staff of the Mechanical Engineering Workshop for their expert skill and assistance in the construction of the mechanical apparatus.

- The National Research Foundation (NRF) and the University of KwaZulu Natal for all financial support received.

List of Acronyms

ADC	Analogue to Digital Converter
AGPS	Assisted Global Positioning System
ALU	Arithmetic and Logic Unit
API	Application Programming Interface
ATX	Advanced Technology Extended
BE	Bull's Eye
CAD	Computer Aided Design
CD	Carrier Detect
CdS	Cadmium Sulphide
CMOS	Complementary Metal Oxide Semiconductor
COO	Cell of Origin
CPU	Central Processing Unit
CTS	Clear to Send
D	Dimensional
DGPS	Differential Global Positioning System
DMA	Decoupled Motion Axes
DOF	Degree of Freedom
DSR	Data Set Ready
DTR	Data Terminal Ready
FK	Forward Kinematics
FKP	Forward Kinematics Problem
GPS	Global Positioning System
GUI	Graphical User Interface
IC	Integrated Circuit
IK	Inverse Kinematics
IKP	Inverse Kinematics Problem
LCD	Liquid Crystal Display
LTI	Linear Time Invariant
MAST	Multi Axis Simulation (Shake) Table
P	Prismatic
PC	Personal Computer
PCB	Printed Circuit Board
PKM	Parallel Kinematics Machine
PWM	Pulse Width Modulation
PS	Power Supply
R	Rotational
RTS	Request to Send
S	Spherical
SE	Solid Edge
SLAM	Simultaneous Location and Mapping
TOA	Time of Arrival
TOF	Time of Flight
TTL	Transistor, Transistor Logic
U	Universal
USART	Universal Synchronous Asynchronous Receiver Transmitter
VB	Visual Basic
VOR	VHF Omni-directional Ranging
VSLAM	Visual Simultaneous Localization and Mapping

Table of Contents

Table of Figures

1 Introduction

1.1 Mechatronics

This was a mechatronics project as it consists of parts from 4 engineering disciplines. This term was first mentioned in 1969 by a senior engineer, Mr. Tetsuro Mori, of the Japanese company Yaskawa. Mechatronics is the synergistic combination of several engineering disciplines, consisting of technologies from mechanical, electronic, control, and software engineering as illustrated in Figure 1. [1]

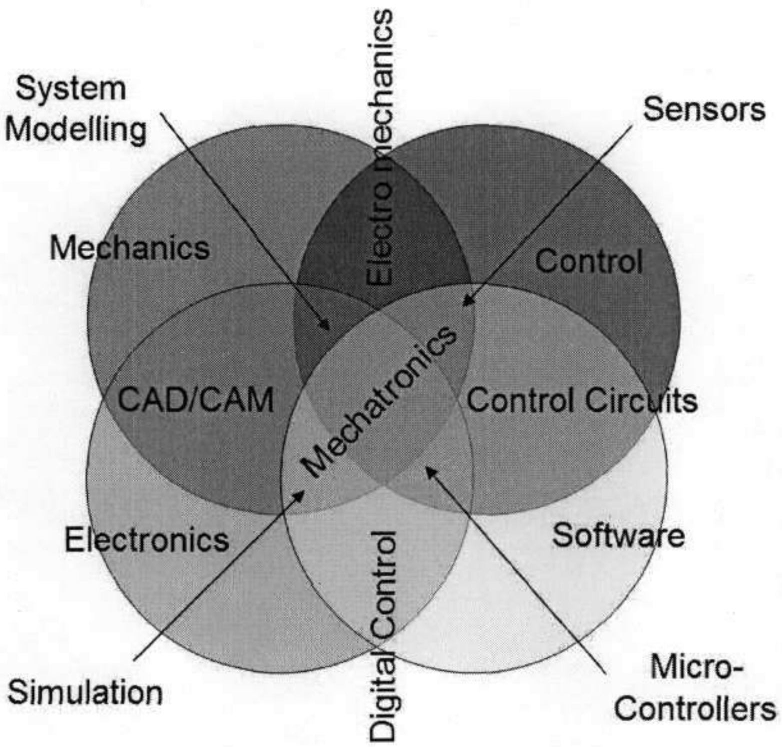


Figure 1 Graphical illustration of Mechatronics

(Adapted from [2])

Essentially, mechatronics adds intelligence to mechanical designs. With the rapid advance of electronic technology, designs that were once purely mechanical are now best accomplished with electronics or a combination of both. Traditional mechanical solutions in modern machinery are being improved on or replaced by mechatronic solutions. [2]

1.2 Motivation for the Study

There has been a renewed interest in parallel mechanisms, and currently researchers from all over the world are investigating or creating new parallel kinematics machines for industry. Serial type mechanisms have reached the limits on their advantages of speed, payload capacity, etc. Parallel mechanisms on the other hand can be designed to be faster, carry larger masses, and they can be designed to have greater accuracy. See Figure 6 for an illustration. The parallel robot that inspired this study was the Flex-Picker by ABB Automation. These machines, as well as many automated multi-axis machines, operate under overall 'open loop' control. The exact position in space of the end effecter or tool head is not sensed directly. With knowledge of the robot's initial position, the control system uses differential measurements from sensors on axes that track linear translation and rotations of shafts/gears, to track the position of the end effector. It then uses this information to plot a trajectory to future positions. A direct end effector sensing system would provide additional data for positioning. Errors in the control system brought on by errors in the actuator sensors may then be corrected, to achieve better positioning accuracy and repeatability. This however, is dependent on the resolution of the sensor system involved and the positioning capability of the robot.

The design of the robot and sensor system were specific to the agricultural and food processing/packaging industry. For this application robot speed is paramount, accuracy and repeatability is less stringent but within lie within certain tolerances, roughly in the millimetre range. Parallel robots are perfectly suited for such an application. The sensor system for such an application needs to be robust for the environment in which it is intended to work. This environment is subject to conditions of humidity, vibration and contaminants. An additional sensor system providing direct end effector position location adds sensor redundancy to the control system, thereby improving the ability of accurate positioning.

1.3 Scientific Contribution of the Dissertation

The scientific contribution of this dissertation lies with the fact that there hasn't been a parallel robot designed with an integrated direct end effector sensor system. Furthermore, the mathematical modelling of parallel robots uses intense matrix theory of Jacobians and Lagrangian formulae that are not easily followed. This dissertation aims to provide a simplified geometrical model of the parallel robot designed. It also provides closed form algebraic solutions to the forward and inverse kinematics for this Flex-Picker type PKM.

1.4 Project Objectives

The objectives of this project were:

- To research various types of parallel kinematics machines. To perform a study of the Delta type (Flex-Picker) PKM structure.
- Design and construct a scaled version of the Delta robot. Simulate the multi-DOF machine in a CAD package.
- Research, design and implement a sensor system that would align a machine's end effector with its base and be able to track its location in space. Develop, calibrate and test in 2D first. Then extrapolate the design to 3D.
- Design a control system for the robot which interprets data from the sensor system. Develop algorithms for movement control and data acquisition to and from sensors. Control the machine's movement by electronic hardware and software programming.
- Conduct a performance analysis of the design.

1.5 Project Specifications

1.5.1 Mechanical Specifications

Size: The robot should be a scaled version of a commercial system, for the purpose of a kinematics and controls study. The mounting framework of the robot should be 650 mm (length) x 300 mm (width) x 550 mm (height). The heavy inertial frame (relative to the size of the robot) should dampen any effect of vibration from the motors on the frame itself.

Workspace: The workspace of the end effector should cover 120 mm (length) x 120 mm (width) x 100 mm (height).

Positioning Accuracy: This scaled adaptations sole purpose is a study of parallel mechanism machine design. It will not be doing any work of pick and place or assembly. Accuracy therefore is of little consequence. An accuracy of 5 mm is more than sufficient.

Positioning repeatability: Likewise, repeatability is also not an issue. However a repeatability of 95% within a 5 mm radius of the intended position will be aimed for.

Robot Speed: Not applicable.

Robot Acceleration: Not applicable.

Payload Carrying Ability: Not applicable. As mentioned, this machine will not carry a payload.

1.5.2 End Effector Sensor Specifications

Sensor Resolution: This should be higher than the positioning capability of the robot. A resolution of 2 to 4 mm would be acceptable, as this was the positioning accuracy required by the food processing and packaging industry.

Sensor Repeatability: The repeatability of the sensor system should be higher than 97% with the robot positioned at its absolute maximum distance from the screen.

Sensitivity Distance: Must sense the end effector at a distance of 200 mm.

Sensitivity Area: 160 mm x 160 mm. This must be larger than the length x width of the workspace.

1.6 Research Publications

1. 22nd International Conference on CAD/CAM, Robotics and Factories of the Future, July 2006. Track: Advanced Control Systems. "Mechatronic Sensor System for Robotic and Automated Machines", by A. A. Shaik, Prof. G. Bright and Prof. W. L. Xu.

2. Incom'2006: 12th IFAC Symposium on Information Control Problems in Manufacturing, September 2006, Volume 1 - Track "Robotics and Factory of the Future". "Modular Sensor System for Flexi-Picker and Multi-Axis Automated Machines", by A. A. Shaik, Prof. G. Bright and Prof. W. L. Xu.

3. ACRA 2006: Australasian Conference on Robotics and Automation, December 2006. "Robotic Sensor System for Automated Machines", by A. A. Shaik, Prof. G. Bright and Prof. W. L. Xu.

4. ISAM 2007: 2007 IEEE Symposium on Assembly and Manufacturing, July 2007. "Sensor System for Multi Axis Automated Assembly and Manufacturing Machines", by A. A. Shaik, Prof. G. Bright and Prof. W. L. Xu.

5. AFRICON 2007: IEEE AFRICON Conference, September 2007. "Closed Loop Sensor System for Automated Machines", by A. A. Shaik, Prof. G. Bright and Prof. W. L. Xu.

6. IJISTA: International Journal of Intelligent Systems Technologies and Applications. "Parallel Robot Design Incorporating a Direct End Effector Sensing System", by A. A. Shaik, Prof. G. Bright and Prof. W. L. Xu. Publication pending.

1.7 Dissertation Outline

Chapter 1, Introduction: Introduces the topic of the dissertation, listing project objectives, specifications and publications.

Chapter 2, Parallel Mechanisms: Presents a history of the most influential parallel robots ever conceptualised or built. The Flex-Picker robot's capabilities are then highlighted.

Chapter 3, Direct End Effector Sensor System: Presents research on various position location technologies, discusses the sensor concept that was used and proves its resolution.

Chapter 4, Mechanical Design: Discusses the design of parallel robots and then presents the design of the modified delta mechanism. The system is modelled and the solutions to the forward and inverse kinematics are discussed.

Chapter 5, Electronic Hardware: The electronic components used in the design are discussed. The schematics and PCBs are then presented.

Chapter 6, Control Design: The system is characterised through a standard linear time invariant (LTI) modelling technique. Controllers are designed in the S and Z domains, and the discrete time controller implementation is illustrated.

Chapter 7, Software: Three software languages were used in the design of the system. These are discussed as well as the various software functions used to model or control the system.

Chapter 8, Calibration, Simulation Results and Prototyping: Discusses the mechanical and software calibration of the system. Simulation results of the forward and inverse kinematics, vibration, a designed trajectory and the control system are presented. The prototype is then illustrated and its performance discussed.

1.8 Chapter Summary

This chapter serves to introduce the reader to the project, which was the design of a parallel robot with an integrated direct end effector sensor system. It provides a motivation for the study, highlights the contribution of the dissertation, lists the project objectives and its specifications for both the sensor system and the parallel robot.

2 Parallel Mechanisms

2.1 A History of the most Influential Parallel Robots

Theoretical works on parallel mechanisms (leading up to parallel robots, or machines) date back to centuries ago, when mathematicians investigated polyhedra. However there is no clear evidence of a complete parallel manipulator until more recent times. One of the first recorded designs of a parallel mechanism (or machine) is accredited to James E. Gwinnett. He applied for a patent in 1928 for a motion platform for the entertainment industry which was based on a spherical parallel mechanism. It was visionary and was designed only a few years after the first colour motion picture and the first with sound. This is shown in Figure 2 a.

Ten Years later Willard L.V. Pollard invented a new industrial parallel robot for automated spray painting. It was a 5-DOF, 3 branch parallel robot that was never built. The first PKM industrial robot to be built was co-designed by Pollard's son, Willard L.G. Pollard Jr. On October 29, 1934, Willard Jr. filed a patent for a spray painting machine. The patent consisted of two parts: an electrical control system and a mechanical manipulator. The mechanical manipulator was a parallel robot based on a pantograph. Willard Jr.'s patent was issued on June 16, 1942. This machine is shown in Figure 2 b. [3, 4, 5]

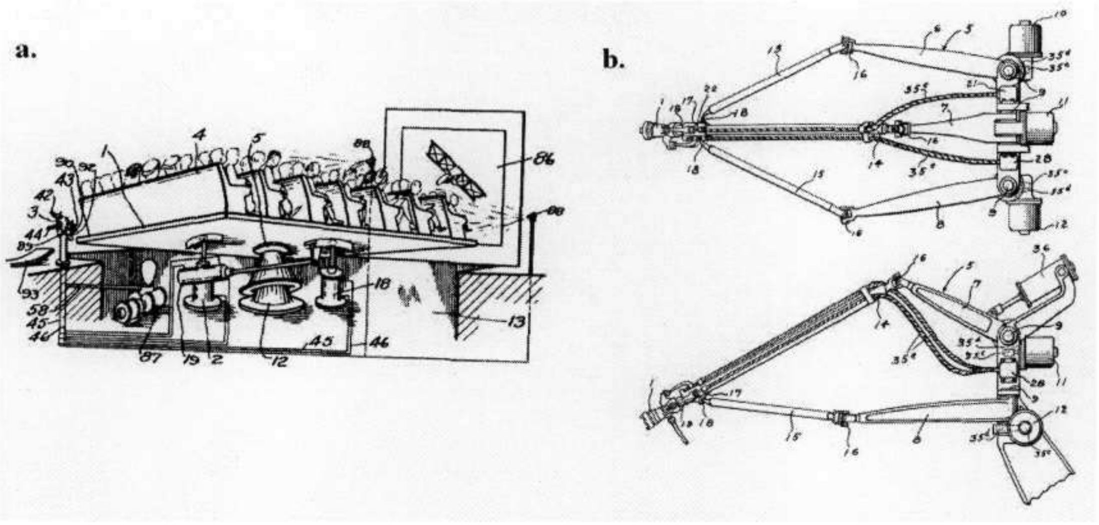


Figure 2 First Patented Parallel Mechanisms [3]

a. Possibly the first spatial parallel mechanism, patented in 1931

b. The first spatial industrial parallel robot, patented in 1942

In 1947 a new parallel mechanism was invented by Dr. Eric Gough, which would become the most popular, revolutionary parallel robot that would be replicated over a thousand times. It was the variable-length-strut octahedral hexapod (a polyhedron with 8 faces having 6 legs separating the base from the table). The universal tire-testing machine was invented to determine the properties of tires under combined loads, and was based on an earlier hexapod design as mentioned by Dr. Gough in his paper "Universal tire test machine" contained in the proceedings of FISITA (pp. 117-137, May 1962). Systems with six jacks (hexapods), with three vertical and three horizontal, have been so common that their origins were forgotten. Their popularity was due to the fact that for small variations, the jack adjustments would be simple and interpretable. These systems, or slightly modified versions, are known under the acronym MAST, i.e. Multi-Axis Simulation (or Shake) Table, and are still manufactured by numerous companies. These hexapods are shown in Figure 3. [3, 4, 5]

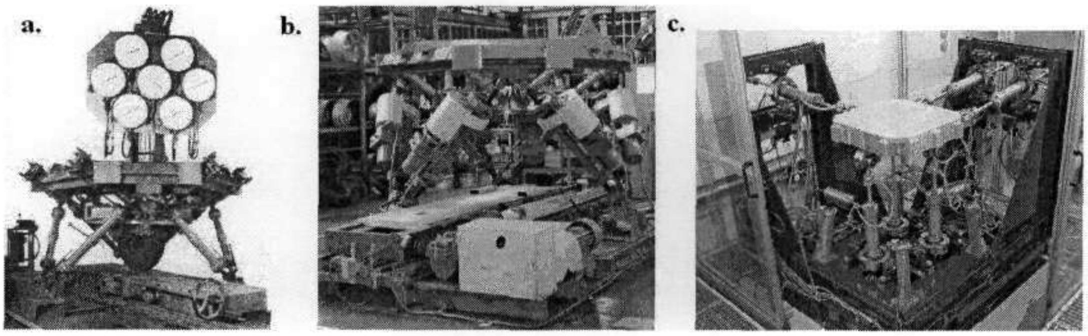


Figure 3 Hexapods [3]

a. The first octahedral hexapod, the original Gough platform of 1954
b. Tire testing machine in 2000, just before Dunlop started using another method
c. A typical MAST system, hexapods of this type have existed long before the Gough platform.

The distinguishing characteristic about the Gough platform was the arrangement of the six struts. Since large ranges of motion were needed, he selected the symmetrical arrangement of an octahedron. The machine was built in the early 1950s, was fully operational in 1954 and played an important role in the birth of rubber science.

In 1965, a paper written by D. Stewart entitled "A platform with 6 degrees of freedom" appeared in the proceedings of the British IMechE journal (Vol. 180, No. 15, pp. 371-385), where he described a 6-DOF motion platform for use as a flight simulator. The parallel mechanism, illustrated in Figure 4 a, was different from the octahedral hexapod which is oddly referred to as the "Stewart platform." Stewart's paper had a great impact on the subsequent development of the field of parallel kinematics. A number of uses were suggested for the hexapod, many of which were accurate predictions of the future. [3, 4, 5]

In 1962, US engineer Klaus Cappel was given the task of improving an existing conventional 6-DOF vibration system based on a hexapod, by the Franklin Institute Research Laboratories in Philadelphia. This MAST originally had four horizontal actuators positioned in a cyclic pattern. However, the redundancy of the seven-strut configuration was too complex to control and the resulting antagonistic forces eventually fractured the table. Mr. Cappel then came up with the same octahedral arrangement as the one designed by Dr. Gough. The corporate office of the Sikorsky Aircraft Division of United Technologies then made a request to the Franklin Institute for the design and construction of a 6-DOF helicopter flight simulator. Mr. Cappel produced his octahedral arrangement and applied for a patent on December 7, 1964. It was granted in 1971 by the US Patent and Trademark Office. At that time Mr. Cappel was unaware of Gough's invention (or of Stewart's paper which was not yet published). [3, 4, 5]

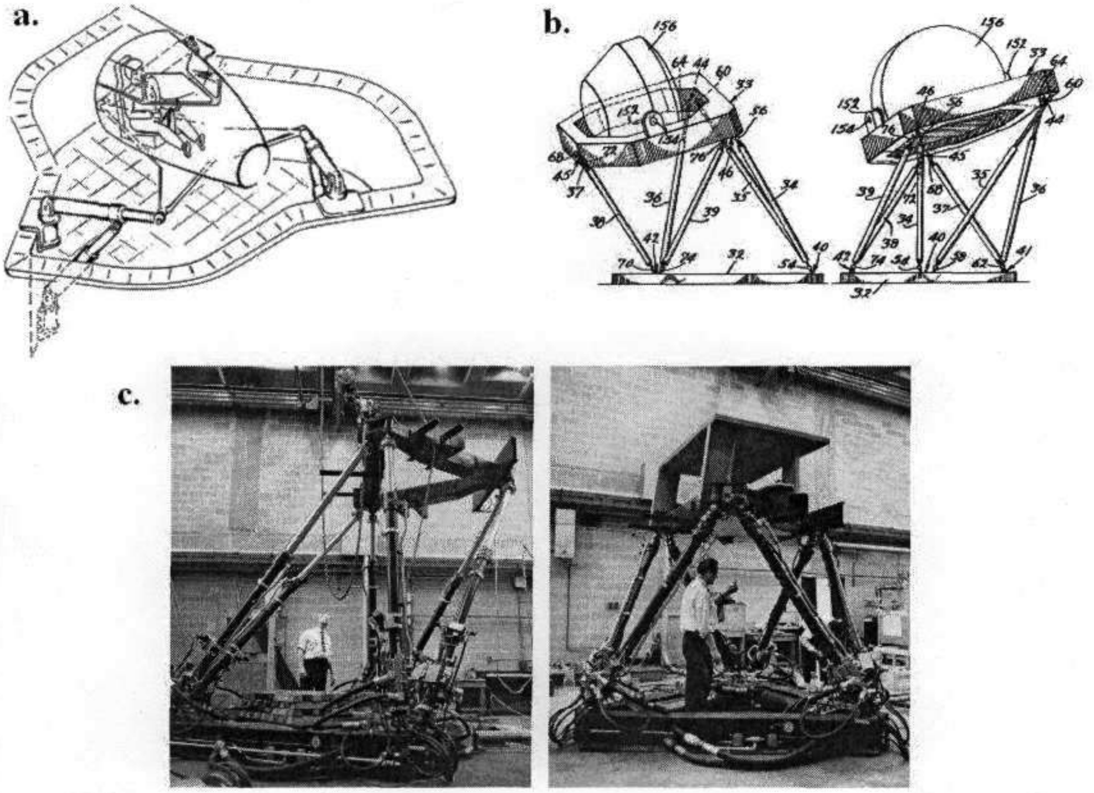


Figure 4 Flight Simulators [3]

a. Schematic of the one and only "Stewart platform"

b. Excerpt from the first patent on an octahedral hexapod issued in 1967

c. The first flight simulator based on an octahedral hexapod as in the mid 1960s

In 1987 Karl-Erik Neumann (founder of Neos Robotics) designed a new parallel kinematics robot, the Tricept (see Figure 5 c.). The major challenge for its control system was the lack of adequate microprocessor computational power. In 1992, Comau Pico launched the first multiprocessor controller which resolved this problem. The Tricept was designed to overcome the shortcomings of existing robots for the assembly of relays on switchboards. Its use was then extended as a machine tool for automotive and aerospace companies that wanted micron level repeatability, stiffer robots with greater power, and flexibility. [6]

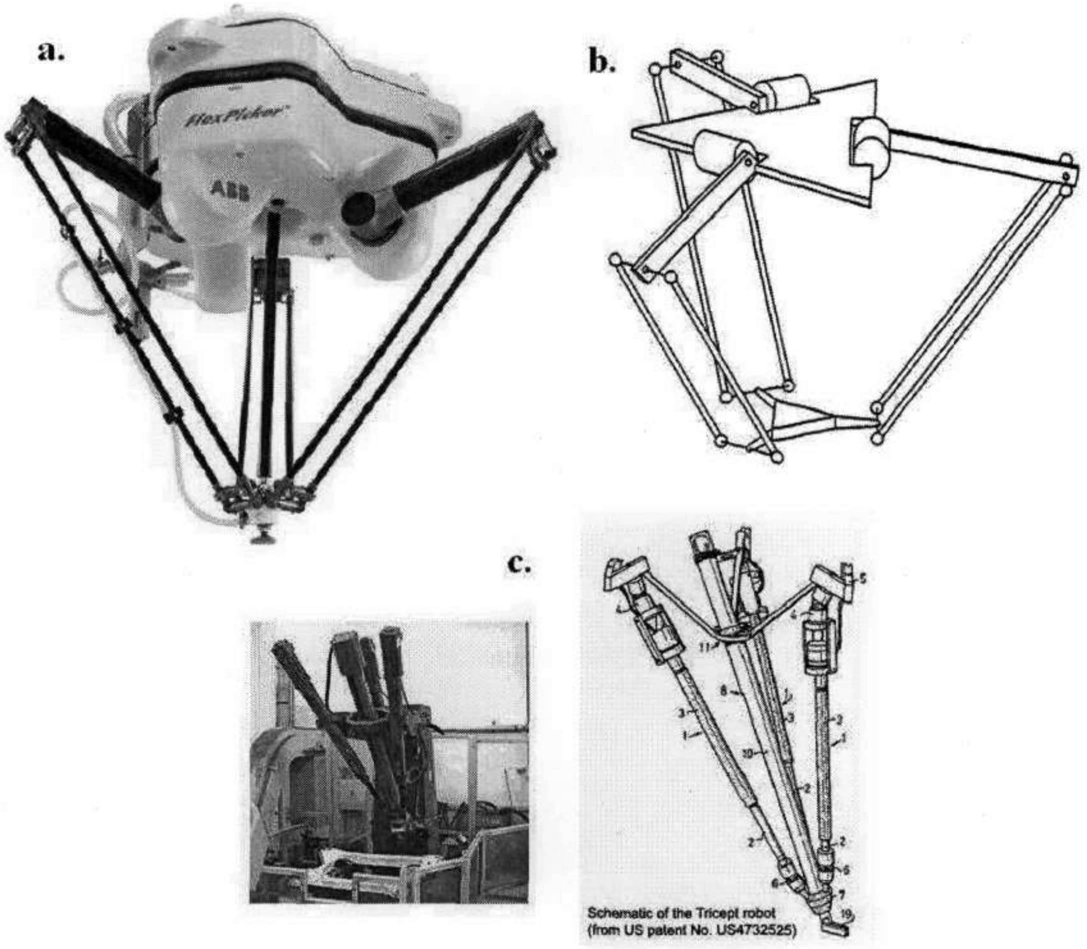

a.
FlexPicker
ABB
b.
c.
Schematic of the Tricept robot
(from US patent No. US4732525)

Figure 5 Popular PKMs

a. IRB 340 Flex-Picker [7]

b. Clavel's Delta Architecture [8]

c. Neos Tricept and Tricept design patent [6]

Another popular parallel robot is the Delta robot (see Figure 5, a, b.), invented by Prof. Reymond Clavel of École Polytechnique Fédérale de Lausanne. The Delta robot has three actuators controlling 3 translational DOFs of the mobile platform, with an additional linkage providing rotational movement of the end effector. This robot is used mostly as a pick-and-place facilitator; other rare applications include machining and assembly. The Delta robot unlike the Tricept, was licensed to various companies. Some machine tool manufacturers have built PKMs based on the Delta robot architecture, and have obtained patents for them. One such company is ABB Flexible Automation of New Berlin which produces the Flex-Picker Robots. [6, 8]

2.2 Comparison between Parallel and Serial Technology

The few parallel mechanisms addressed thus far present a fresh outlook from conventional mechanism design. Most robots used for industrial manufacturing have articulated arms equipped with serial technology. The increased need for automation and flexible production means new applications and higher performance requirements for industrial robots. Current serial robot technology is limited whereas parallel kinematics structures have the highest potential for improvement [9].

In a serial topology each actuator axis is in line relative to the preceding one in an open kinematics chain. In a purely parallel topology the actuator axes (one for each DOF) have a fixed arrangement and position in space. From the fixed base, a number of arms and links are coupled in parallel to the end effector, forming closed kinematics chains. Hybrid systems use a combined arrangement of parallel and serial mechanisms to extract the best features of each architecture. The result of the parallel design is a robot that has increased stability and arm rigidity. As there is less flexing of the arms there is high repeatability. Also the high structural stiffness of a closed-loop kinematics chain allows it to exert strong forces in its workspace [10]. The speed of displacement is often greater since the end effector has low inertia; due to the fact that the motors are generally positioned on a fixed base [11]. Depending on the exact configuration, the load can often be purely axial, and is always distributed through the legs [12]. With serial robots each link is required to support not only the load of the sample, but also the load of all the links and drive units preceding it. This means considerable inertia, thus limiting the robots' acceleration capability and dynamic performance [9]. Furthermore the end-of-arm flexing errors are cumulative. Both absolute accuracy and repeatability errors generated by each unit, together with manufacturing errors, gear backlash and hysteresis are amplified throughout the serial structure. In a parallel structure all the errors are averaged. The use of large displacement compliant joints to construct PKMs further reduces the errors mentioned above and can lead to sub-micron accuracy [13]. Additionally, PKMs have a reduced sensitivity to temperature, lower energy consumption, lower manufacturing cost and higher reliability.

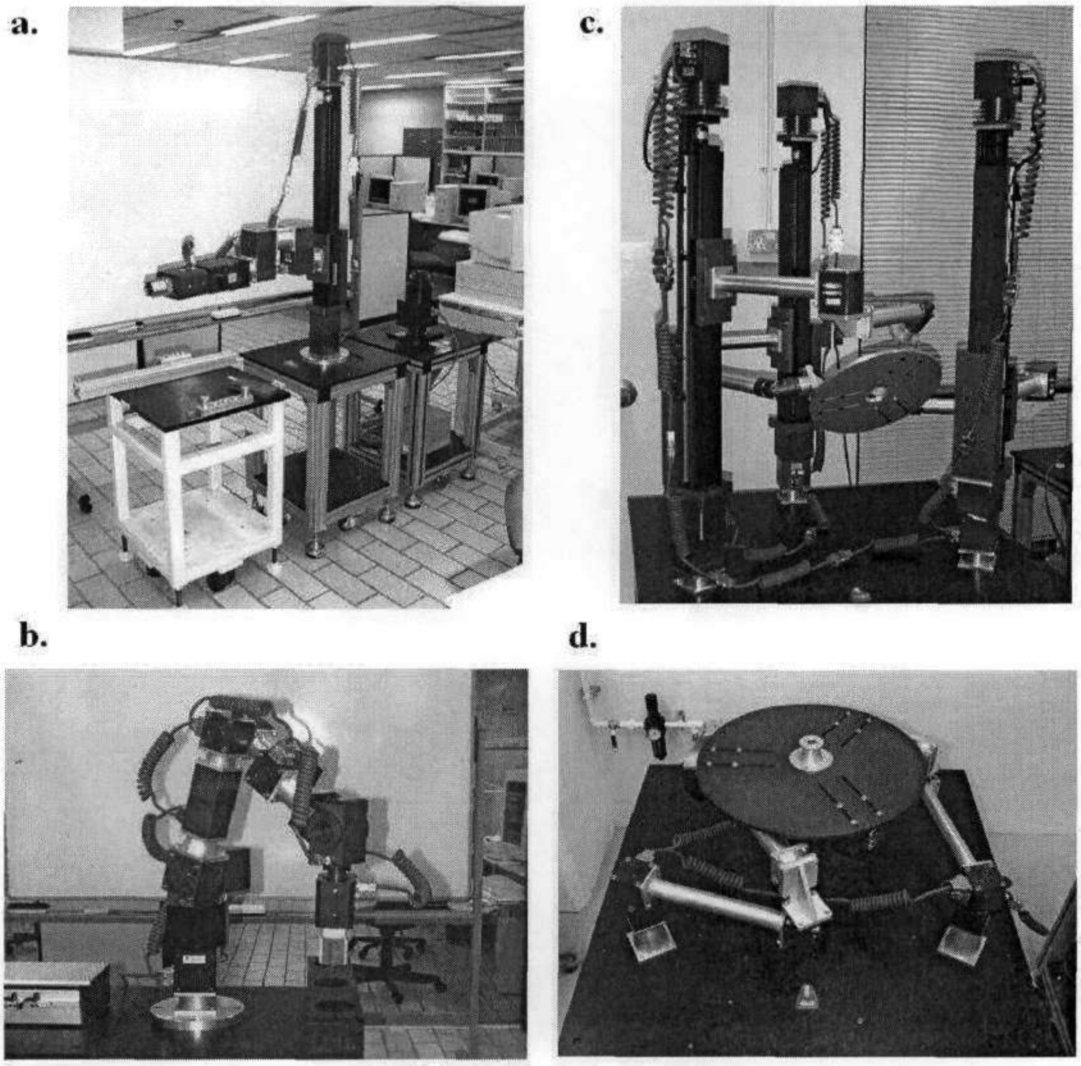


Figure 6 A comparison between parallel and serial machines [14]

a. 4-DOF SCARA Robot c. 3 RPRS PKM

b. 7-DOF Redundant Robot d. 3 RRR Planer PKM

PKMs offer good design variation and designers can stretch their creativity to conceptualize machines with varying architectures, more so than they could do with serial topologies.

The main disadvantage of parallel kinematics robots is that, in general, they have a larger footprint to workspace ratio as a result of the configuration of the axes. There are some exceptions such as the Tricept, but other devices such as those based on the hexapod PKM take up a sizable work area. Another drawback of PKMs is that their performance depends heavily on their geometry and optimal design has therefore become a key issue for their development [15]. The ratio between payload and machine moving mass is higher, and therefore the payload variations influence the machine behaviour remarkably. Control is also difficult as the kinematics and dynamic models are far more complex than those of serial machines. [10]

The most studied parallel mechanisms have 6 DOFs. They have a small useful workspace, are riddled with design difficulties and their direct kinematics is a very difficult problem. The problem of parallel mechanisms with 2 and 3 DOFs can be described with exact equations for motion and exact mathematical solutions relating position of the end effector to actuated variable magnitudes, i.e. they are closed form. Additionally, not all singularities of a 6-DOF parallel mechanism can be found readily, but these are identified easily for parallel mechanisms with 2 and 3 DOFs. For such reasons, parallel mechanisms with less than 6 DOFs have increasingly attracted more and more attention with respect to industrial applications. [16]

Table 1 A comparison between parallel and serial mechanisms

ADVANTAGE: ✓	PARALLEL MECHANISMS	SERIAL MECHANISMS
Higher Stability and Arm Rigidity	✓	
Greater Repeatability	✓	
Higher Stiffness	✓	
Greater Speed and Acceleration	✓	
Load Distribution among Actuators	✓	
Dynamic Behaviour Immune to Payload Variations		✓
Lower Energy Consumption	✓	
Lower Manufacturing Cost	✓	
Smaller Positioning Errors	✓	

Lower Sensitivity to Environmental Conditions	✓	
Larger Workspace to Footprint Ratio		✓
Geometry Independent Performance		✓
Simpler Control		✓
Simple Forward Kinematics		✓
Simple Inverse Kinematics	✓	
Predictable Dynamics		✓

2.3 The Flex-Picker Robot

The PKM designed is based on the Flex-Picker robot. The IRB 340 Flex-Picker system (see Figure 5 a) is a three or four axis robot designed for light assembly, material handling, and pick and place applications. This machine has three lightweight reinforced carbon fibre arms. Using a vacuum gripper, the IRB 340 is capable of 120 pick and place operations per minute, for objects with a mass up to one kilogram. It has a maximum acceleration of 10 g, a maximum velocity of 10 ms-1 and a maximum torque output of one N. It is suitable for manipulating light weight objects such as mechanical parts, electronic modules for personal computers and cell phones, pharmaceuticals, and food. Due to its off-line configuration ability, it is highly adaptable to product changeover. [7]

2.4 Chapter Summary

This chapter begins with a history of the most influential parallel robots ever designed. It then presents a comparison between serial and parallel technologies, pointing out the advantages and disadvantages of each. It ends with a look at the capabilities of the Flex-Picker robot, on which the PKM designed was based.

3 Direct End Effector Sensor System

3.1 An Overview of Current Location Sensing Technologies

There are numerous technologies available that can be used to locate objects in space. They are distinguished from one another by the use of different media, transducers and processing techniques. To effectively design a sensor system for object location it is imperative to at least discuss these technologies and methods available even though some of these may not be suitable for the task at hand. The common problems are resolution and the type of media used. Resolution is always an issue with measurement and in this particular case refers to the maximum positional error possible with the technology being discussed. With regard to the medium or transducer stimulant, some media may prove undesirable; for instance using an ultrasound triangulation system in a small closed environment. A discussion of these techniques/technologies now follows.

3.1.1 Global Positioning System (GPS)

The Global Positioning System (GPS) is the most significant recent advance in navigation and positioning technology. It was designed and built by the U.S. Department of Defence. It used to be known as the Navstar GPS and was first brainstormed at the Pentagon in 1973 as an error-proof navigation satellite system. The first operational GPS satellite was launched in 1978 and by the mid-1990s the system was fully operational with 27 satellites, 24 active and 3 backups. Each of these 1 500 kg solar-powered satellites circles the globe at about 19,300 km, making two complete orbits every day. Today anyone with a small receiver can use the system free of charge.

GPS satellites transmit signals to the receivers on the ground. These receivers are passive. The receivers require an unobstructed view of the satellites above, so they are used outdoors and perform poorly within forested areas or near tall buildings.

The operation of the GPS system is based on a mathematical principle called tri-lateration (see section 3.1.3.1 Lateration). Each GPS satellite transmits data that indicates its location and the current time. The satellites synchronize their operations so that these repeating signals are transmitted at the same instant. The signals move at the speed of light and arrive at a GPS receiver at slightly different times as some satellites are farther away than others. By multiplying this time by the speed of light an estimate of the distance to the satellites is determined, on the assumption that the signal traveled in a straight line. With one signal the receiver knows it is located somewhere on the surface of an imaginary sphere centred at the satellite. The receiver can provide a reasonable approximation of its position in 3D space when it has received at least four unique signals. Its location is the intersection of these 4 spheres. The orbits of the satellites were arranged so that at any time, there are at least four satellites visible at any point on the Earth. The system depends on a very accurate time reference. Atomic clocks are used on the satellites, but these cost around R 350k – R 700k. Hand held GPS receivers use simple quartz clocks, and it overcomes its timing inaccuracy by constantly resetting its time reference based on the signals it receives. There is only one time value that will make all the "signal spheres" intersect at 1 point, instead of having an "intersection space" of where the receiver could be.

The accuracy of a position determined with GPS depends on the type of receiver. Most hand-held GPS units have an accuracy of about 10-20 m (spherical radius), while other types of receivers use a method called Differential GPS to obtain higher accuracy.

[17, 18, 19]

3.1.1.1 Differential GPS (DGPS)

Three "signal spheres" will always intersect even if all the timing and data are inaccurate, however 4 spheres will not intersect at one point with inaccuracies. There are a number of errors that occur with GPS, these are:

- The signal slows down as it passes through the ionosphere and troposphere
- Signal multi-path
- Receiver clock errors
- Inaccuracies in the reported position of the satellites
- Low number of visible satellites
- Bad satellite geometry

DGPS uses two GPS receivers. Observations made by a known stationary location (base or reference) is used to correct the data received by a GPS unit at an unknown location (rover). As the base station knows its location exactly, it can determine satellite signal errors. This is done by measuring the ranges to each satellite using the received signals which are compared to the actual ranges calculated from its known location. These differential corrections for each tracked satellite are transmitted to all the roving GPS receivers in the area. The corrections are then applied to the calculations. DGPS generally achieves an accuracy of less than 1 meter. [20] Presents a cheap solution with errors of less than 5 cm.

[17, 19]

3.1.1.2 Assisted GPS (AGPS) / Indoor GPS

AGPS is a variant of GPS that utilizes an assistance server to aid in position determination. The assistance server has the ability to access information from a reference network and possesses computing power exceeding that of the GPS receiver. The receiver found in non-ideal locations for position fixing, communicates with the assistance server (via cellular communication) and by sharing tasks, the process position fixing is quicker and more efficient albeit more dependent on cellular coverage. Its intended application is in urban areas when the user is located in cities, under heavy tree cover, or indoors but not in underground car parks or tunnels.

Indoor GPS, or high sensitivity GPS, is a combination of AGPS and massive parallel correlation. Outdoor GPS applications tend to experience multi-path in only its most benign form, i.e. a reflection that is weaker than the direct line-of-sight signal. The situation is different indoors. The reflection can readily exceed the power of the direct signal, or the direct signal can disappear altogether.

Laser indoor GPS systems are also available. These systems are composed of three or more laser transmitters, and function in a similar manner to conventional GPS. The transmitter uses both laser and infrared light to create one-way position information of the relative azimuth (horizontal component of a direction) and elevation from the transmitter to the receiver. The receiver has photodiodes inside its module and senses the transmitted laser and infrared light signals. With the addition of a second transmitter of known location and orientation, the position of the receiver can be calculated in the base coordinate system. By adding two more transmitters, the system will have four laser transmitters having its accuracy maximized. As in satellite-based GPS, a one-way signal path is created from transmitters to the receiver, allowing an unlimited number of receivers to continuously and independently calculate positions whenever two or more transmitters are in view. This indoor GPS metrology system has millimetre accuracy and finds use in the manufacturing sector, tracking parts, materials, or people.

[18, 19, 21]

3.1.2 Bluetooth, WIFI and Cellular Networks

Bluetooth, WiFi and Cellular networks provide a means for location sensing.

Bluetooth devices form mini-cells, and with a sufficient number of Bluetooth cells (access points) installed, the position of a transmitter can be deduced by knowing the cell with which a device is communicating (or location base stations in a WiFi network), discussed in [22] for locating people in buildings. [23] Also takes into account signal strength from the access points. Since the transmitted signal energy decreases almost proportionally with the distance between stations and mobile terminals, this relation can be used to determine the distance from a particular node. The signal energy is measured by the mobile device and is transmitted to a central server that calculates its location. These Bluetooth networks have an accuracy ranging from 2 to 5 m; this is subject to the number of cells installed in the region and the spacing between them.

Cell of Origin (COO) is a mobile positioning technique for finding a caller's cell (the basic geographical coverage unit of a cellular telephone system) location. It works in the same way as the Bluetooth network just mentioned, however the accuracy may be as close as one hundred meters from the target (in an urban area) or as far off as thirty kilometres, the accuracy is dependent on the number of base stations in the area. For this reason, when precision is important COO is often used in conjunction with some other technology, such as GPS or Time of Arrival (TOA).

[24, 25]

3.1.3 Triangulation

This technique uses geometry to calculate position and is achieved by considering the properties of triangles to compute object locations. There are 2 subcategories of triangulation i.e. lateration and angulation.

3.1.3.1 Lateration

The term lateration is used for distance measurements. It computes the position of an object by measuring its distance from multiple reference positions. Calculating an object's position in 2D requires distance measurements from 3 non-collinear points. For 3D measurements, distances from 4 non-coplanar points are required. There are three general approaches to measuring the distances required by the lateration technique.

Direct - Direct measurement of distance. Direct distance measurements are simple to understand but may prove to be difficult or even impossible to obtain if the distances are rather large.

Time of Flight (TOF) - Measuring distance from an object to some point P using TOF means measuring the time it takes a signal to travel between the object and a point P at a known velocity. The problem is made more complex if the object itself is moving. It is possible to make the calculations if there is a known mathematical function for the acceleration. If however it varies arbitrarily then it is impossible to find a solution. Other factors have to be considered to get an accurate answer:

- For instance if the signal used is ultrasonic, and the air medium density varies over the distance then the speed of the signal itself will vary. The density of air at each point cannot be known, and there is no guaranteeing that it will vary according to some well behaved function. A best guess estimate must therefore be used, e.g. a function that varies density based on altitude. Temperature and humidity also influence air density and must be factored into the equation.

- Reflection is another problem as direct and reflected signals look identical. Bats and other creatures that use sonic vision statistically prune away reflected measurements by aggregating multiple receivers' measurements and observing the environment's reflective properties.

- Another issue in taking TOF measurements is the time reference. When only one measurement is needed, as with round-trip or radar reflections, timing is simple because the transmitting object is also the receiver and must maintain its own time with sufficient precision to compute the distance. In other systems (like GPS) where the receiver and transmitter are on different objects they must be synchronized precisely to get an accurate time measurement of distance.

Attenuation - The intensity of an emitted signal decreases as the distance from the emission source increases. The decrease relative to the original intensity is the attenuation. Given a function correlating attenuation and distance for a type of emission as well as the original strength of the emission, it is possible to estimate the distance from an object to some point P by measuring the strength of the emission when it reaches P. Signal propagation issues such as reflection, refraction, and multi-path cause the attenuation to correlate poorly with distance resulting in inaccurate and imprecise distance estimates.

[26, 27, 28]

3.1.3.2 Angulation

The term angulation is used for angular measurements, and these are used for determining the position of an object. In general, 2D angulation requires two angle measurements and one length measurement such as the distance between the reference points. In 3D, one length measurement, one azimuth measurement, and two angle measurements are needed to specify a position. Angulation implementations sometimes choose to designate a constant reference vector (e.g. magnetic north) as 0°. Phased antenna arrays are an excellent enabling technology for the angulation technique. Multiple antennas with known separation measure the time of arrival of a signal. Given the differences in arrival times and the geometry of the receiving array, it is then possible to compute the angle from which the emission originated. If there are enough elements in the array and large enough separations, the angulation calculation can be performed.

The VHF Omni-directional Ranging (VOR) aircraft navigation system uses a different implementation of the angulation technique. VOR stations are ground-based transmitters in known locations which repeatedly broadcast 2 simultaneous signal pulses. The first signal is an omni-directional reference containing the station's identity. The second signal is swept rapidly through 360° like the light from a lighthouse at a rate such that the signals are in phase at magnetic north and 180° out of phase to the south. By measuring the phase shift, aircraft listening to a VOR station can compute the angle formed by the direct vector to the VOR station and the vector from the VOR to magnetic north, with an accuracy of 1°. Aircraft location can be computed via angulation using 2 VOR stations.

[26, 27, 28]

3.1.4 Interferometers

Interferometry is the applied science of combining two or more waves, which are said to interfere with each other. The interference pattern is considered a state with amplitude and phase which depends on the amplitude and phase of all the contributing waves.

The interferometer can measure displacements to a resolution within a fraction of the wavelength of light. It has enabled Micro- and Nano-scale measurements of position or movement. An optical heterodyne interferometer recently designed at NASA's Jet Propulsion Laboratory can measure linear displacements with an error of 20 pm (pico, 10^{-12}). Many companies, such as Agilent Technologies and Zygo, provide laser interferometers for the purpose of high precision manufacturing.

Interferometers work by splitting a beam of light using a semi-transparent mirror into two separate beams that travel different paths along two arms. One of the beams is directed to a mirror located at a certain distance to provide a reference in measurements. The other beam reaches a mirror or reflector fixed on the moving object. This beam, being reflected, recombines and optically interferes with the reference beam (or beams) at the detector. The interference pattern, typically a set of alternating bright and dark stripes called fringes, displays subtle differences between the two travel paths. By analyzing these fringe patterns, the position of the moving target can be measured. For instance the fringe pattern is shifted by one fringe when one arm is stretched relative to the other by $\frac{1}{3000}$ mm. An important characteristic of interferometry is that only displacement is measured, not absolute position. The initial distance to the movable mirror is not measured, only the change in position of the mirrors with respect to each other can be determined. If the initial position is known, integrating the change in position over time will yield its current position. This is the principle by which all inertial measurement units work. [29]

3.1.5 Grid Encoders

An optical grid encoder offers yet another solution and is capable of 2D dynamic measurements. These grids are made by OPTRA and the Heidenhain Corporation. They have a coverage range up to 380 mm × 380 mm with high accuracy and excellent repeatability.

The grid encoder is composed of a grid plate with a waffle-type grating of closely spaced lines (4 μm signal period) and a non-contact scanning head which is able to measure translations in two directions. The optical grid plate is attached to an aluminium mounting base. This base is mounted in the plane to be measured (on an X–Y table for instance) and the scanning head is fixed perpendicular to the plate (e.g. on the Z axis attached to the spindle). This system measures the relative planar motion of the two bodies for any curvilinear path in the plane of the mounting base with a resolution of 4 nm and to within an accuracy of ± 2 μm. **[30]**

3.1.6 Imaging Methods

Imaging methods use cameras and sophisticated software to determine position location. [31] Discusses a low cost solution employing a camera and LCD screen to locate an object's coordinates in 2D with high accuracy. The LCD screen displays an image, a small circle or cross somewhere on its surface. The camera which is attached to the end effector then tries to align this shape in some way with the image it produces. It can also be used to determine 2D orientation.

3.1.6.1 Scene Analysis

The scene analysis location sensing technique uses features of a scene observed from a particular vantage point to draw conclusions about the location of the observer or of objects in the scene. Usually the observed scenes are simplified to obtain features that are easy to represent and compare. In static scene analysis, observed features are searched in a predefined dataset that maps them to object locations. In contrast, differential scene analysis tracks the difference between successive scenes to estimate location. Differences in the scenes will correspond to movements of the observer and if features in the scenes are known to be at specific positions, the observer can compute its own position relative to them. The advantage of scene analysis is that the location of objects can be inferred using passive observation and features that do not correspond to geometric angles or distances. The disadvantage of scene analysis is that the observer needs to have access to the features of the environment against which it will compare its observed scenes. Furthermore, changes to the environment in a way that alters the perceived features of the scenes may necessitate reconstruction of the predefined dataset or retrieval of an entirely new dataset. The scene itself can consist of visual images, such as frames captured by a wearable camera, or any other measurable physical phenomena such as the electromagnetic characteristics that occur when an object is at a particular position and orientation. [26, 32, 33]

3.1.6.2 Simultaneous Location and Mapping (SLAM)

SLAM is a technique used by robots and autonomous vehicles to build up a map within an unknown environment while at the same time keeping track of its current position from various sensors. If at the next iteration of map building the measured distance and direction traveled has a slight inaccuracy, then any features being added to the map will contain corresponding errors. If unchecked, these positional errors build cumulatively grossly distorting the map and the robot's ability to know its precise location. There are various techniques to compensate for this such as recognizing features that it has come across previously and re-skewing recent parts of the map to make sure the two instances of that feature become one. Some of the statistical techniques used in SLAM include Kalman filters, particle filters (a.k.a. Monte Carlo methods) and scan matching of range data. SLAM in the mobile robotics community generally refers to the process of creating geometrically accurate maps of the environment. SLAM has not yet been fully perfected, but it is starting to be employed in unmanned aerial vehicles, autonomous underwater vehicles, planetary rovers and newly emerging domestic robots. SLAM usually uses laser range finders or sonar sensors to build the map. However VSLAM (visual simultaneous localization and mapping) uses entirely visual means. [34, 35]

3.1.7 Other

Displacement measuring instruments utilizing eddy currents, capacitive and inductive properties exist, but are not as widely spread as the technologies mentioned.

Accelerometers and Gyroscopes are used to determine position and orientation. They are capable of measuring the change in acceleration of a body. An integration of this yields velocity and a second integration yields distance travelled. Keeping track of these parameters and having been given an initial position for the object, its current position can be inferred. This system is usually used in conjunction with other techniques/technologies such as GPS and radar.

3.1.8 Comparison of Technologies

Technology	Measurement Accuracy	Cost	Range	Size of measurement units / system	Other Disadvantages	Other Advantages
GPS	Within 10 m	Varies from R 800 - 4 000 for receiver alone	Globe	Receiver size 4cm x 4cm x 1cm, Compact processing unit.		3D Position determination
DGPS	Within 5 cm	Varies from R 1 600 - 8 000 for 2 receivers	100 m from base station	2 Receivers (4cm x 4cm x 1cm), 2 Compact processing units.		3D Position determination
Bluetooth Networks	*Ranges from 2 - 5 m*	*Depends on number of bluetooth cells -- R 7 k +*	*120 m radius for 3 cells*	Size of Bluetooth tranceivers (usb *type plug in devices) and* Processing unit (PC).		1D / 2D / 3D Position measurement; can *adjust range by using* more cells
Laser Triangulation	0.04% of range (2.4 microns to 6.6 mm)	R 15 k +	max 16.5 m	Smallest enclosure 6.5 cm x 5 cm x 2 cm --- largest enclosure 6.3 cm x 17 cm x 3 cm	One dimensional measurement; Cannot mount on end effector	Highly robust and reliable
Interferometers	12 pm - 10 nm	R 250 k +	max 10.6 m	6 cm x 25.5 cm x 6.35 cm	One dimensional measurement; positioning and setting mirrors for a 3D moving object would be extremely difficult	
Grid Encoders	± 2 um	R 250 k +	max area 1440 sq cm	XY plates range from 17.5 cm x 17. 5 cm to 38 cm x 38 cm	System cannot be scaled, maximum work area is fixed	2D metrology system
LCD Imaging System	Claimed within 0.05 mm	R 40 k + ... Large LCD screen and High resolution camera	177.8 cm	110 cm x 177.8 cm LCD Screen		
Image Processing Methods	Within 2 mm	R 100 k + ... Expensive Software and multiple high resolution cameras	20 m	Size of cameras and fixed mounted positions in environment	Complex software, real time control would be a problem	Completely passive, no transmitted signals

Figure 7 Comparison of location sensing technologies

3.2 Sensor Feedback System

3.2.1 Requirements

The sensor system should possess the following characteristics or capabilities:

- It should either locate the tool point's spatial coordinates directly,
- Or reduce the errors accumulated in an existing sensor system.
- It should integrate seamlessly with existing techniques for motion control.
- In its most primary function it should locate the tool head in 2D space,
- With some additions and modifications it should locate an object in 3D space.
- It must be modular,
- Robust,
- Fast,
- And error immune to work in a harsh industrial environment.

3.2.2 Sensor System Concept

The first decision to be made was choosing whether the sensing system should be passive or active. Passive systems, or imaging methods as mentioned in section 3.1.6, require lots of processing power, complex software and expensive cameras. Furthermore, with current technology the systems are not real time. An active sensor system is therefore needed, that is one that transmits a signal and then receives it with a sensor array at another location, and then computes position.

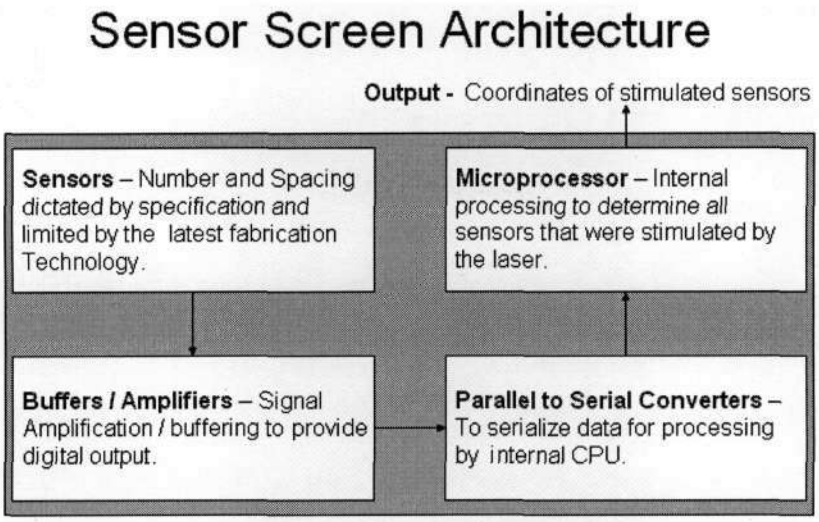
The first step to solving the problem of locating the end effector in space was to reduce the problem to a simpler case, solve it and then attempt a generalization. The problem of locating the end effector of a robot in real world space was first reduced to finding its position in a 2D plane with regard to a point reference. Once accomplished the general problem is solved by attaching two 2D planes at right angles. With such an arrangement 2 axes coincide and if the reference point of each plane coincides, the result is a 3 axis sensor system for position location in 3D space.

After consideration of the available physical quantities used when locating objects, a laser light stimulant was chosen for the task at hand. A laser light sensor can be conditioned to provide a digital output, that is, it provides only 2 voltage levels representing a digital 1 when the sensor is switched on and 0 when it is off. Most sensors used for tracking are analogue in nature and require digitization for use in digital systems, this digitization takes a finite time and the data created occupies a larger memory. If there are m sensors in a sensor array, with each sensor being represented by, for instance, an 8 bit digital value then there are $8m$ bytes of data that have to be processed. The varying value indicating light intensity is irrelevant, as all that is required is a value saying that that sensor has been stimulated by laser light. Furthermore analogue signals are compromised by atmospheric effects, temperature, humidity and unshielded noise from surrounding machinery. Triangulation utilizing radio, ultrasound or infrared waves is not suitable as multiple reflections from surrounding surfaces cause interference. They also require modulation and demodulation to distinguish the signals generated from those created by the environment.

The defining component of this sensor system is a grid of laser light detectors. The detectors need to have a narrow sensitivity wavelength bandwidth as well as viewing angle to prevent wrongful stimulation and spurious results. The designed sensor concept utilizes a direct approach, with a laser or set of lasers, attached to the end effector and the sensor grid (the sensor plane with sensors spaced equally in rows and columns) mounted directly above it, to the side or directly below. This was a natural choice as the coherent nature of laser light aids the task of finding the end effector in 2D if the laser beam remains perpendicular to the sensor plane at all times (this implies that the end effector must be *perfectly horizontal*). The end effector's location is the same as the sensor which is stimulated (in a 2D plane, depth has no meaning). It must be stressed that this sensor system requires only bit (1 or 0) information for each sensor. Each sensor is either stimulated (switched on) or not stimulated (switched off). A stimulated sensor indicates position on the plane as explained. This makes data processing and transfer far simpler and makes control easier. The resolution is limited to the spacing between sensors. If the spot light is smaller than the spacing between sensors, there will be a dead zone between sensors where beam tracking will be lost completely. The laser light detectors are phototransistors. Current fabrication techniques can accommodate hundreds of millions of transistors on a sliver of silicon. INTEL has claimed to have the capability of creating a 45 nm transistor (see INTEL website). This implies that on a 1 mm2 piece of silicon there is an upper limit of 493 827 160 transistors that can be etched on that surface. IBM states that it has produced a 6 nm transistor (see IBM website). From this it should be clear that extremely high densities of transistors can be achieved, but at high cost. Fabrication methods can be used to construct a detector screen with an exceptional and practical resolution. The current accuracy of the IRB 340 Flex Picker is 0.1 mm; the lower limit on screen resolution for absolute 2D positioning would then be 400 phototransistors per square mm, a screen with twice the resolution of the positioning accuracy of the robot. Resolution affects data output, a greater resolution implies more data per unit area (more sensors). A hybrid type system (combining this end effector sensor with conventional motor encoders and software position fixing) would involve a sensor grid with a comparatively smaller resolution. Each sensor provides a checkpoint. Knowing the exact spatial distance between these detectors provides the controller with a means to limit the errors incurred. Instead of accumulating errors from one extremity to the next, errors only exist between successive detectors.

The array of data has to be placed in a data format or byte structure to facilitate processing. This is made possible by parallel to serial data converters. As a numerical example consider first the workspace of the IRB 340 Flex Picker from ABB Automation. The specified workspace envelope is a cylinder with diameter 1130 mm and height of 250 mm. A 1150 mm x 1150 mm screen would be large enough to track the end effector in its specified workspace. Using the same resolution as before i.e. 400 detectors per mm2, there would be 529 000 000 detectors on the screen. This sensor grid consists if 23 000 rows and 23 000 columns. A 4 byte data format (2 bytes for the row and 2 bytes for the column) would be more than sufficient to indicate any single stimulated sensor to an external control system.

Figure 8 displays high level architecture of the detector screen.



Sensor Screen Architecture

Output - Coordinates of stimulated sensors

Sensors – Number and Spacing dictated by specification and limited by the latest fabrication Technology.

Microprocessor – Internal processing to determine all sensors that were stimulated by the laser.

Buffers / Amplifiers – Signal Amplification / buffering to provide digital output.

Parallel to Serial Converters – To serialize data for processing by internal CPU.

Figure 8 High level architecture of detector screen

3.2.3 Sensor System Resolution

The sensor system designed has a resolution of 2.5 mm. The detector screens have a resolution of 10 mm (spacing between successive sensors in rows and columns, see section 5.6 j as well as Figures 37 and 40). The improvement in resolution is due to the use of a laser module, one laser module per detector screen, at the end effector. Each laser module has 12 lasers with a particular arrangement to provide the sensor system with the resolution stated.

To explain how this is achieved an animated depiction of the laser projection onto the detector screen is illustrated in Figures 9, 10 and 11.

The blue circles represent the sensors. The red and green circles represent the lasers. The central laser with the blue bull's eye, henceforth know as BE, is the tracking point or end effector reference. The relative spacing of these lasers are shown in Figure 9. The 4 inner lasers lie on grid points with a grid spacing of 15 mm. These lasers improve the resolution of the detector system to 5 mm, and this laser grid will be known as G5. These lasers are represented by the BE and 3 green circles in Figure 9 b. The outer lasers lie on grid points with a grid spacing of 22.5 mm and improve the resolution of the sensor system to 2.5 mm, and this grid will be known as G2.5. Grid G2.5 has to be used in conjunction with grid G5 and is represented by BE and the red circles in Figure 9 b.

The black solid lines of Figure 9 b are reference lines. The dashed lines represent the 2.5 mm resolution. Figure 9 a shows the mechanical component used to mount the lasers (see Figures 16-18).

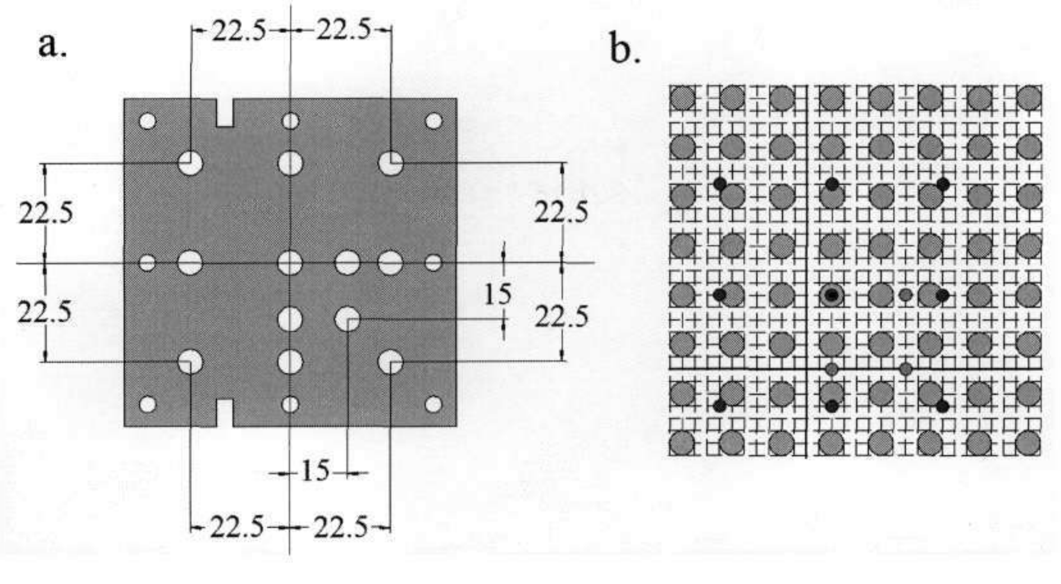


Figure 9 Laser grid

a. Laser guide indicating dimensions and laser positions

b. Laser grid projection onto detector plane, centred on s(i, j)

Suppose initially BE is at sensor $s(i,j)$, in Figure 9 b. If it moves 5 mm to the right it no longer lies on s(i, j), however the green circle on the same y grid line lies on sensor $s(i+2,j)$ (Figure 10 a). If BE were to move 5 mm to the left, that same green circle would lie on sensor $s(i+1,j)$ (Figure 10 b). So any horizontal 5 mm displacement can be tracked; this is the maximum distance the end effector would have to move horizontally before another sensor indicates position. Similarly 5 mm vertical displacements can be tracked (shown in Figure 10 c, d). Combined displacements can also be tracked. Suppose BE moves 5 mm to the right and 5 mm up, then the bottom left hand green circle lies on sensor $s(i+2,j-1)$ (Figure 10 e). If BE is moved 5 mm to the left and 5 mm down, then that same green circle lies on sensor $s(i+1,j-2)$ (Figure 10 f).

Figure 10 Depiction of laser grid projection onto detector plane for 5 mm resolution

a. BE moved 5 mm to the right of $s(i,j)$, $s(i+2,j)$ detects laser

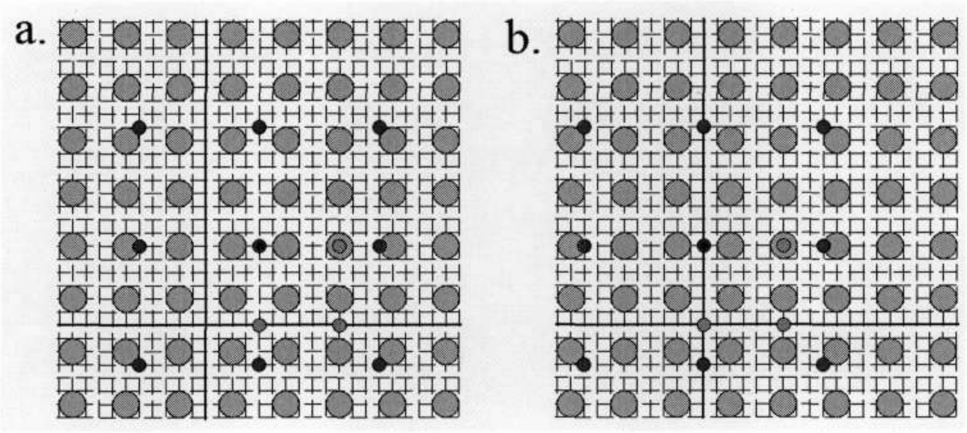
b. BE moved 5 mm to the left of $s(i,j)$, $s(i+1,j)$ detects laser

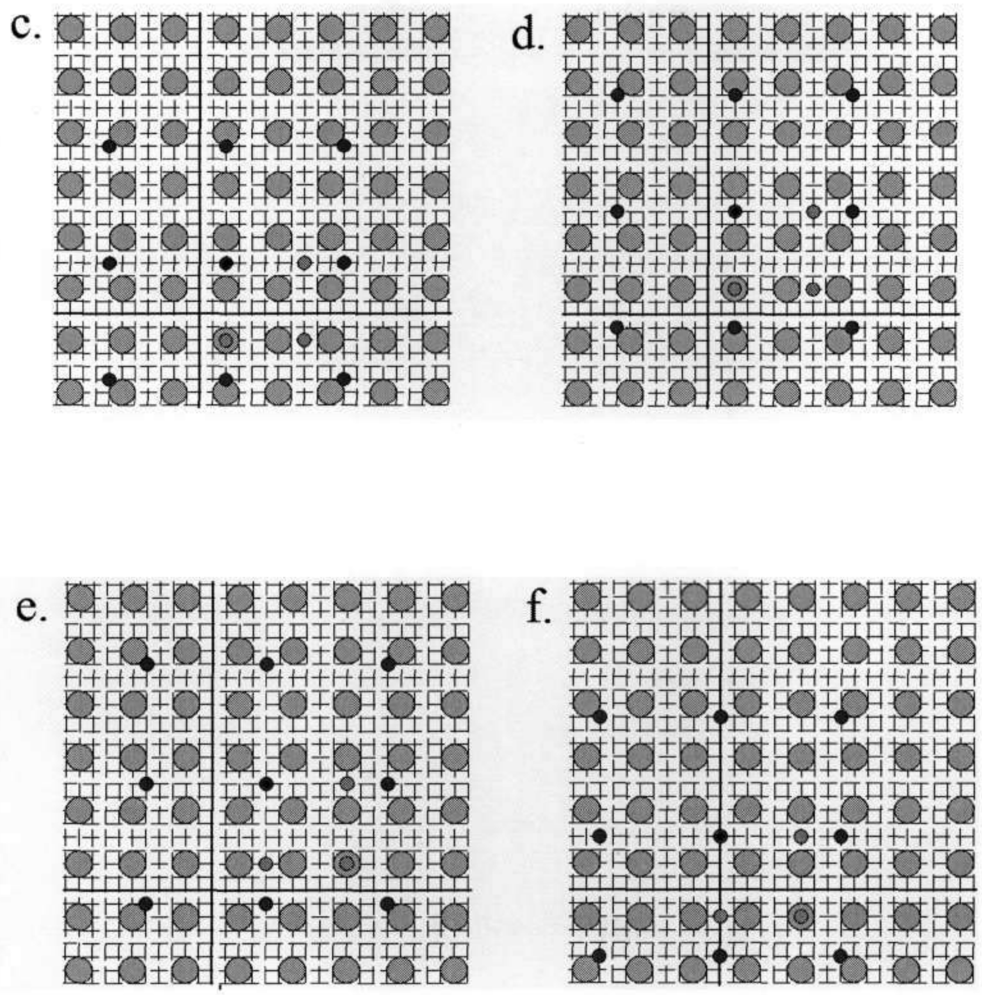
c. BE moved 5 mm down from $s(i,j)$, $s(i,j-2)$ detects laser

d. BE moved 5 mm up from $s(i,j)$, $s(i,j-1)$ detects laser

e. BE moved 5 mm up & to the right of $s(i,j)$, $s(i+2,j-1)$ detects laser

f. BE moved 5 mm down & to the left of $s(i,j)$, $s(i+1,j-2)$ detects laser





The 2.5 mm resolution is proved as follows. Start of at position in Figure 9 b. Move 2.5 mm to the right. The laser sharing the same y grid line to the far left of BE then lies on sensor $s(i-2,j)$ (Figure 11 a). If BE is moved 2.5 mm down then the laser at the top left hand corner lies on sensor $s(i-2,j+2)$ (Figure 11 b). If BE is now moved 2.5 mm to the left the laser sharing the same x grid line directly above BE moves onto sensor $s(i,j+2)$ (Figure 11 c). The pattern can now be seen clearly. As BE moves about the 2.5 mm grid surrounding sensor $s(i,j)$, the lasers on G2.5 hit sensors lying on the 20 mm grid surrounding sensor $s(i,j)$ (i.e. $s(i \pm m, j \pm n)$ where m, n = 0 or 2 as shown in Figure 11 a to g). So any 2.5 mm vertical, horizontal or combined displacement can be sensed. Similarly displacements of 7.5 mm can be tracked however the designed PKM is not expected to have such a large error.

Figure 11 Depiction of laser grid projection onto detector plane for 2.5 mm resolution

a. BE moved 2.5 mm to the right of $s(i,j)$, $s(i-2,j)$ detects laser

b. BE then moved 2.5 mm down, $s(i-2,j+2)$ detects laser

c. BE moved 2.5 mm to the left, $s(i,j+2)$ detects laser

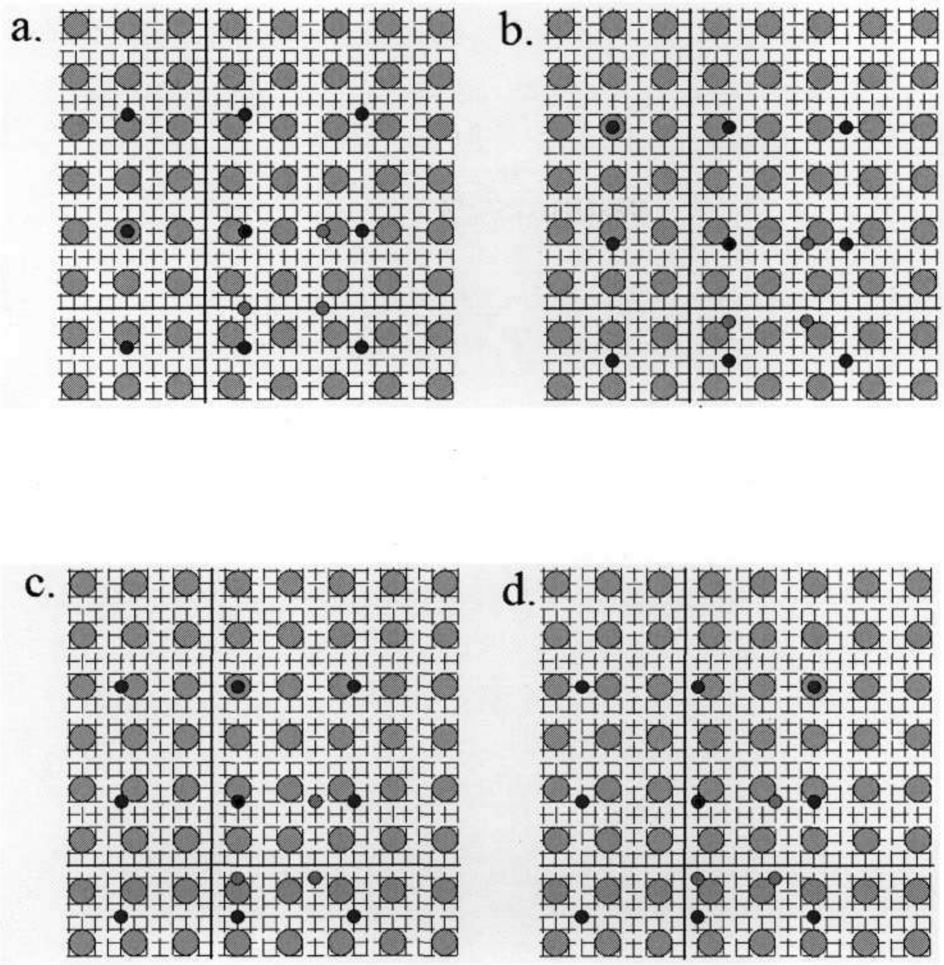
d. BE moved 2.5 mm to the left, $s(i+2,j+2)$ detects laser

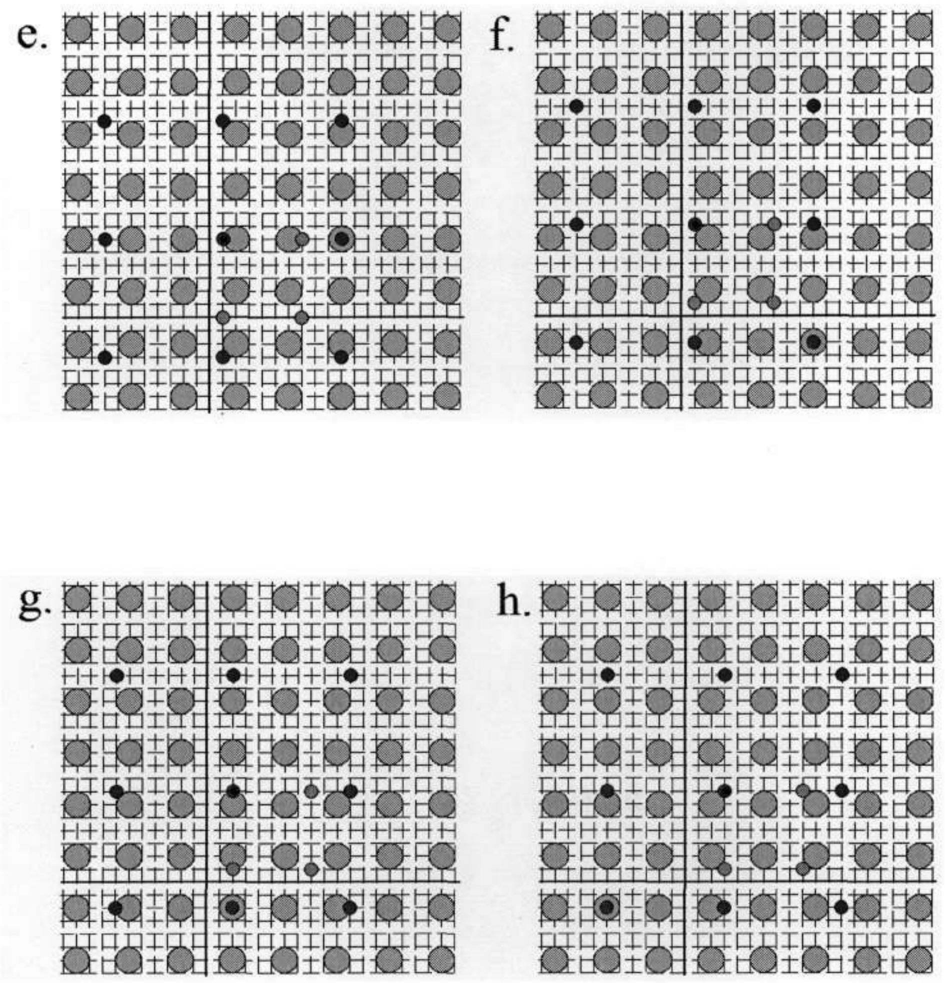
e. BE moved 2.5 mm up, $s(i+2,j)$ detects laser

f. BE moved 2.5 mm up, $s(i+2,j-2)$ detects laser

g. BE moved 2.5 mm to the right, $s(i,j-2)$ detects laser

g. BE moved 2.5 mm to the right, $s(i-2,j-2)$ detects laser





The meaning of stimulated sensors does not refer to the exact position of the reference point BE but rather displacements from that point. For this machine there are sensors that monitor the angular positions of the legs, these are the potentiometers of the servo motors used (sections 5.3 and 5.4). Those measurements together with these end effector displacement error readings provide improved end effector tracking. The errors from positioning are not expected to exceed 5 mm and this is also one of the machine design specifications.

3.2.4 Advantages and Disadvantages of Sensor Concept

Advantages:

- Detector screen of any practical size can be built from modular components.
- The screens do not have to be mounted vertical or horizontal, as long as they are perpendicular to each other for 3D object tracking.

- Resolution is independent of screen size, and the detector screens can be scaled up maintaining the exact resolution designed.
- Data is purely digital as the sensors are either on (1) or off (0).
- It facilitates 2D or 3D tracking.

Disadvantages:

- For 3D tracking 2 screens are needed, and the system becomes twice as expensive.
- The end effector must remain perpendicular to the detector screens.
- This sensor must fit the space and mounting constraints imposed by the manufacturing system.

3.3 Chapter Summary

This chapter discussed current technologies used for position location. Most of the equipment used for each technology was unsuitable for the task at hand for one of more of the following reasons: the systems were too expensive, slow, inaccurate or bulky. A tabulated comparison of the technologies was given which highlighted accuracies, ranges, advantages and disadvantages. From this investigation an idea stemmed from two sources i.e. laser triangulation and John Ziegert's idea [31] of an imaging screen. The sensor concept was then discussed and its resolution capability proved.

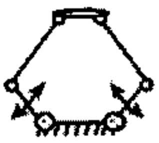
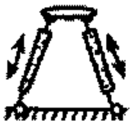
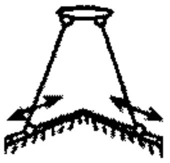
4 Mechanical Design

4.1 Design of Parallel Kinematic Machines

Although robots are usually designed to perform a large variety of tasks it is not realistic to believe that a single robot will be sufficiently flexible and able to manage any task. On the other hand the end user may wish to perform a set of specific tasks with stringent requirements. For this reason a fundamental step in the design of robotic systems is determining the most appropriate mechanical structure of the robot when given the task requirements, such as desired workspace, accuracy, load, and stiffness.

There are three basic types of parallel actuation mechanisms applicable for robot arms, i.e. prismatic, rotary, or fixed linear actuation types, including their modifications. Table 2 shows the comparison of these three mechanisms in terms of their basic characteristics. [36]

Table 2 Comparison of PKM attributes based on actuators used in its design [36]

Type	Rotary 	Prismatic 	Fixed Linear 
Output Force	Small	Large	Large
Actuator Location	Basement	Moving Part	Basement
Moving Mass	Small	Large	Small
Speed	Very Fast	-	Fast
Rigidity	Fair	Good	Good
Workspace	Large	Small	Large
Structure	Simple and compact	Large moving parts	Large basement

Overall the rotary actuation type provides the best characteristics for more general applications even though the resulting rigidity isn't as high as the other types.

The mechanical design of robots may be split into two processes:

- **Structural Design** – It involves the general arrangement of the mechanical structure such as the type and number of joints and how they are connected.
- **Dimensional Synthesis** – This determines the length of the links, the axis and exact location of the joints, and the necessary maximal joint forces/torques.

The performance of a robot is drastically dependent on both syntheses. A comparison between two different structures may only be made after a careful dimensional synthesis. This is more so for closed loop parallel robots. [37]

4.1.1 Structural Design

a. Machine Topology

Machine topology describes the number and type of joints, as well as the number of branches in the structure. It can be described compactly if the machine is symmetric. For instance in Figure 6 c, the parallel robot is described as 3-DOF 3 RPRS, meaning that the PKM has 3 degrees of freedom with 3 branches and each branch from base to end effector has a rotational joint, followed by a prismatic joint, another rotational joint and finally a spherical joint. In Figure 6 d, the parallel robot is described as 3-DOF 3 RRR planer, meaning there are 3 branches and each branch has 3 rotational joints. Its motion is restricted to the plane; that is it has 3-DOF with 2 translational and 1 rotational. Clavel's Delta robot may be described as 3-DOF 3 RUU, with U representing a universal joint. ABB's Flex-Picker may be described as 3-DOF 3 RSS, as it has spherical joints on its "knee" and "ankle". The hexapod may be described as 6-DOF 6 SPU, where P denotes a prismatic actuated joint. As an additional example Figure 12 illustrates 2 more parallel mechanisms. Their structures are completely different. They have different branches, different joints, and different actuator positions. This provides an indication as to the various number of PKM designs possible.

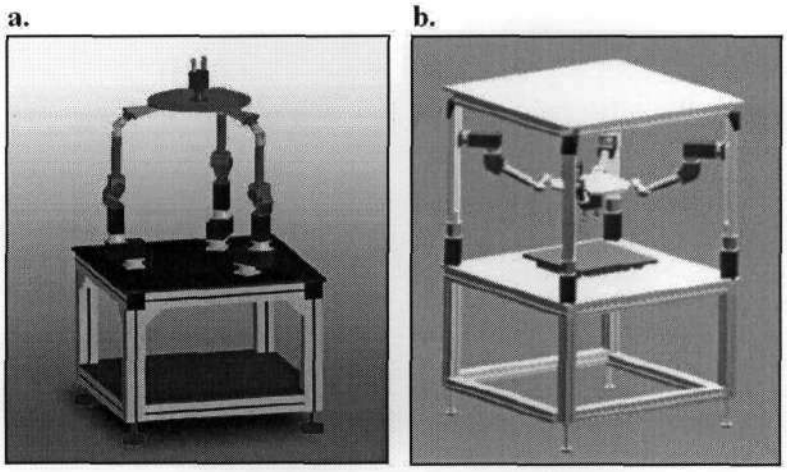


Figure 12 A comparison of machine topologies [14]

a. 4-DOF 3 RRRS
b. 6-DOF 4 PRRS

b. Actuator Positioning

The arrangement of the actuators affects the way the robot moves. This also determines the position of singularities (see section 4.3.2. h). See Figure 13 a.

4.1.2 Dimensional Synthesis

a. Dimensioning

Choosing the size of robot arms is a matter of finding a good compromise between weight, stiffness, and its ability to reach the entire workplace. Depending on robot sizing, an optimal choice of motors and transmission ratios of reducers should be addressed. [11] See Figure 13 b.

b. System Modelling

Once the dimensioning is complete, models for both the forward and inverse kinematics (section 4.3.2) and dynamics (section 4.3.2 k.) may be obtained. These models are then used for motion control.

As this machine accomplishes no pick and place operations, a rigorous and complete dynamic analysis is unnecessary. The complete kinematics model with closed form solutions will be explained, as one is used in the control system. Dynamic modelling is discussed in section 4.3.2 k but is not used in the control system of the robot. Theory of plant estimation from linear control theory was used to model the system, and these equations inherently capture the dynamic behaviour of the system.

The effectiveness of a kinematics model can be described by three terms, i.e.

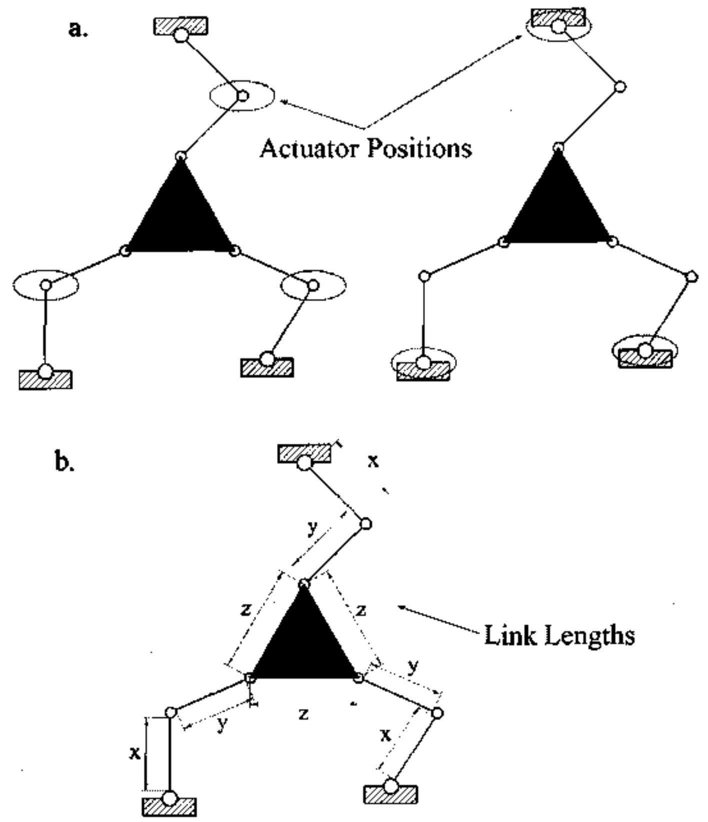


Figure 13 Actuator positioning and dimensioning

(Adapted from [14])

a. Figure depicting the choice of actuator positions

b. Figure depicting the link length design parameter

- **Accuracy** – Is defined as the difference between the actual position in space and position calculated using the kinematics model.
- **Repeatability** – It is the difference between actual positions when repeatedly sent to the same position coordinates. It includes hysteresis of joints, thermal elongation of links etc, but does not include bad model design or wrongly estimated parameters.
- **Resolution** - The size of the smallest positional step.

The order of importance is accuracy, repeatability and then resolution. The parameters that are used to model a robot are split into two categories, as the following list indicates [14]:

Non-Geometrical Model Parameters:

- Compliance and stiffness
- Gear backlash
- Encoder resolution
- Temperature related expansion
- Linkage wobble

Geometrical Model Parameters:

- Structure
- Angles between links
- Links dimensions
- Zero positions of links
- Mechanical Design

4.1.3 Design Considerations

A general approach to the design of PKMs should cover the following issues [38]:

- Determination of the reachable workspace,
- Kinematics stiffness described by several local and global manipulability measures,
- Relation of driving and actuator forces,
- Overall elastic stiffness of the structure,
- Static stability analysis.

Kinematically, an n-DOF non redundant PKM also implies that each leg should also be an n-DOF serial kinematics chain, regardless of the number of legs. To simplify design and development efforts, there are a few additional considerations [39]:

- Symmetric Design - Each leg is identical to the others. Hence, each leg should have the same number of active joints. As the total number of (1-DOF) active joints in a 6-DOF non redundant PKM is six, the number of legs for a symmetric design can be six (1-DOF actuated joint per leg), three (two 1-DOF actuated joints per leg), or two (three 1-DOF actuated joints per leg).
- Types of joints - Four types of commonly used joints are considered, i.e., 1-DOF revolute (R), 1-DOF prismatic (P), 2-DOF universal (U), and 3-DOF spherical (S) joints. Among

them, the spherical and universal joints are meant as passive joints, the prismatic joints are meant as active joints (they are ineffective as passive joints), and revolute joints can be used as either passive or active joints.

- Active joints are placed close to the based so as to reduce the moment of inertia and increase the loading capacity and motion acceleration.
- Passive 3-DOF spherical joints are used to reduce the number of passive joints and make the design compact.
- At most one (active) prismatic joint can be employed in each of the legs due to its heavy and bulky mechanical structure.
- Designing for Decoupled Motion Axes (DMA). This gives the robot simple kinematics for easy analysis, design, trajectory planning, and motion control. [39]

Figure 14 illustrates a graphical summary of the steps used in the parallel mechanism design process.

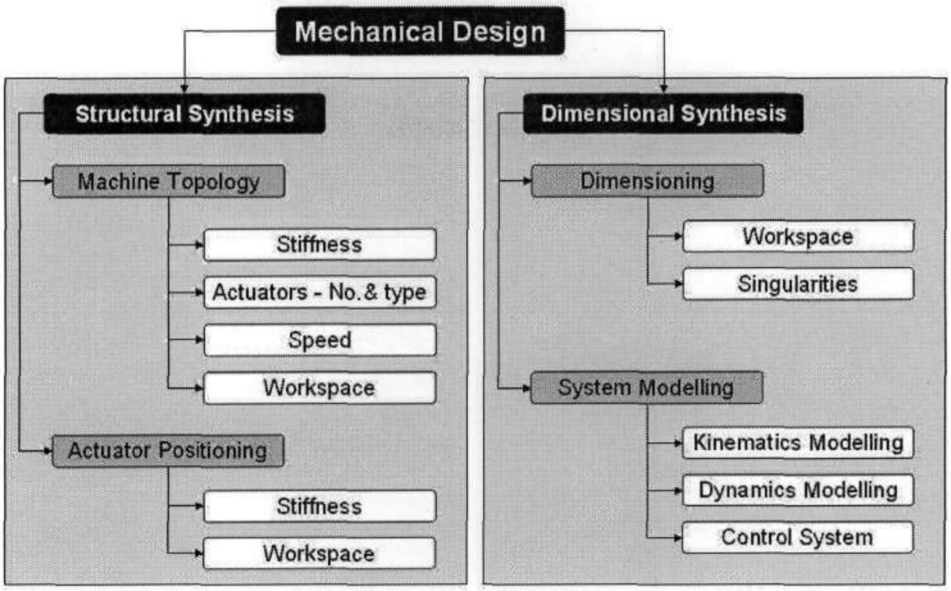


Figure 14 Design criteria used in constructing PKMs

4.2 Structural Design of the Modified Delta Robot

The mechanical structure is based on that of a Flex-Picker pick and place parallel kinematics industrial robot, and is a scaled adaptation.

The design consists of 4 articulated legs; 4 servo motors (as used in model helicopters); a plate end effector with attached lasers; ball-cup joints and a mounting frame. The entire mechanical structure is 600 mm in length, 400 mm wide and 500 mm high. Figures 15 and 16 illustrate the parts and some assemblies. Figures 17 and 18 illustrate the complete assembly with detector boards in various views.

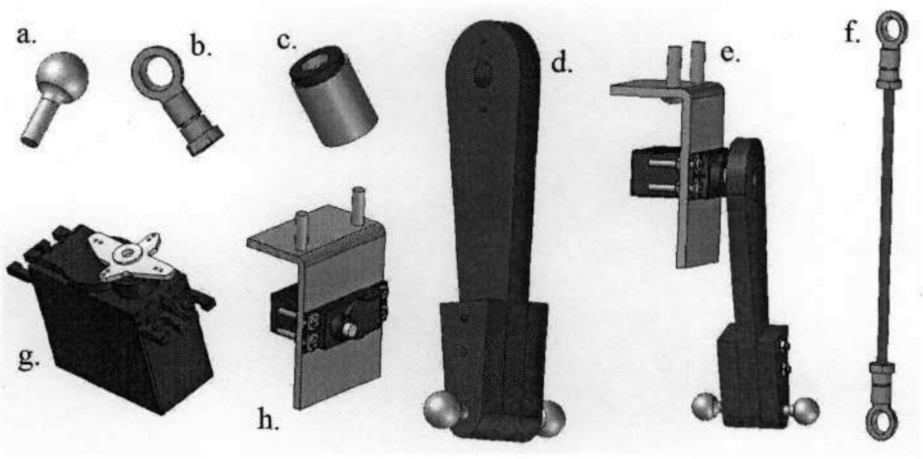


Figure 15 Significant Mechanical Parts

a. Ball from ball in socket bearing
b. Socket/Ball cup
c. Laser
d. Upper leg
e. Servo motor with upper leg attached and mounting bracket
f. Lower leg component
g. Servo motor
h. Servo motor with mounting bracket

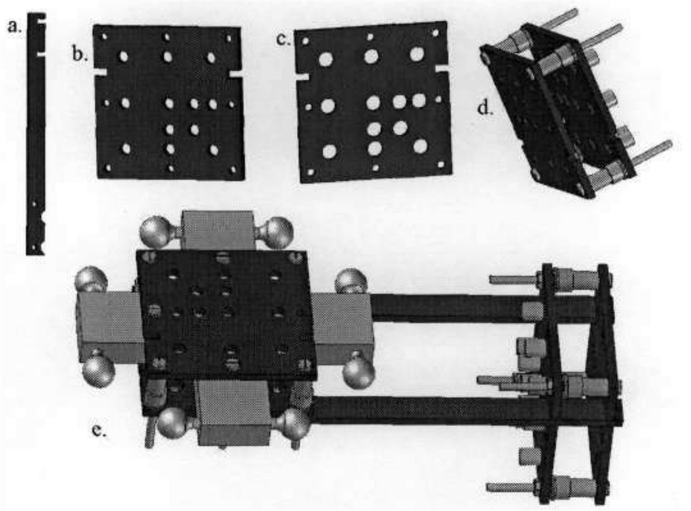


Figure 16 End effector

a. Vertical laser mounting arm
b. Multiple laser guide
c. Multiple laser mounting
d. Laser mounting with guide attached
e. Laser mountings and guides attached to end effector

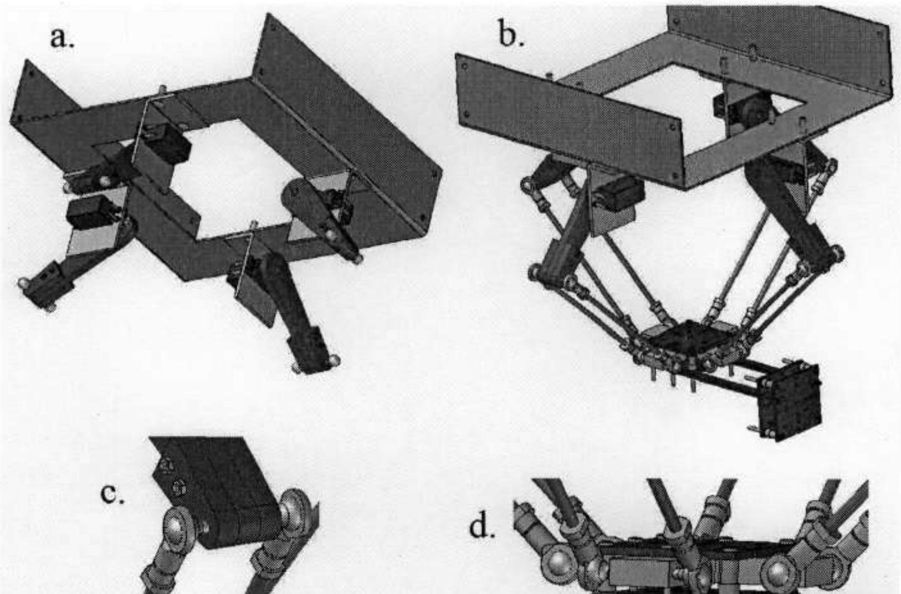
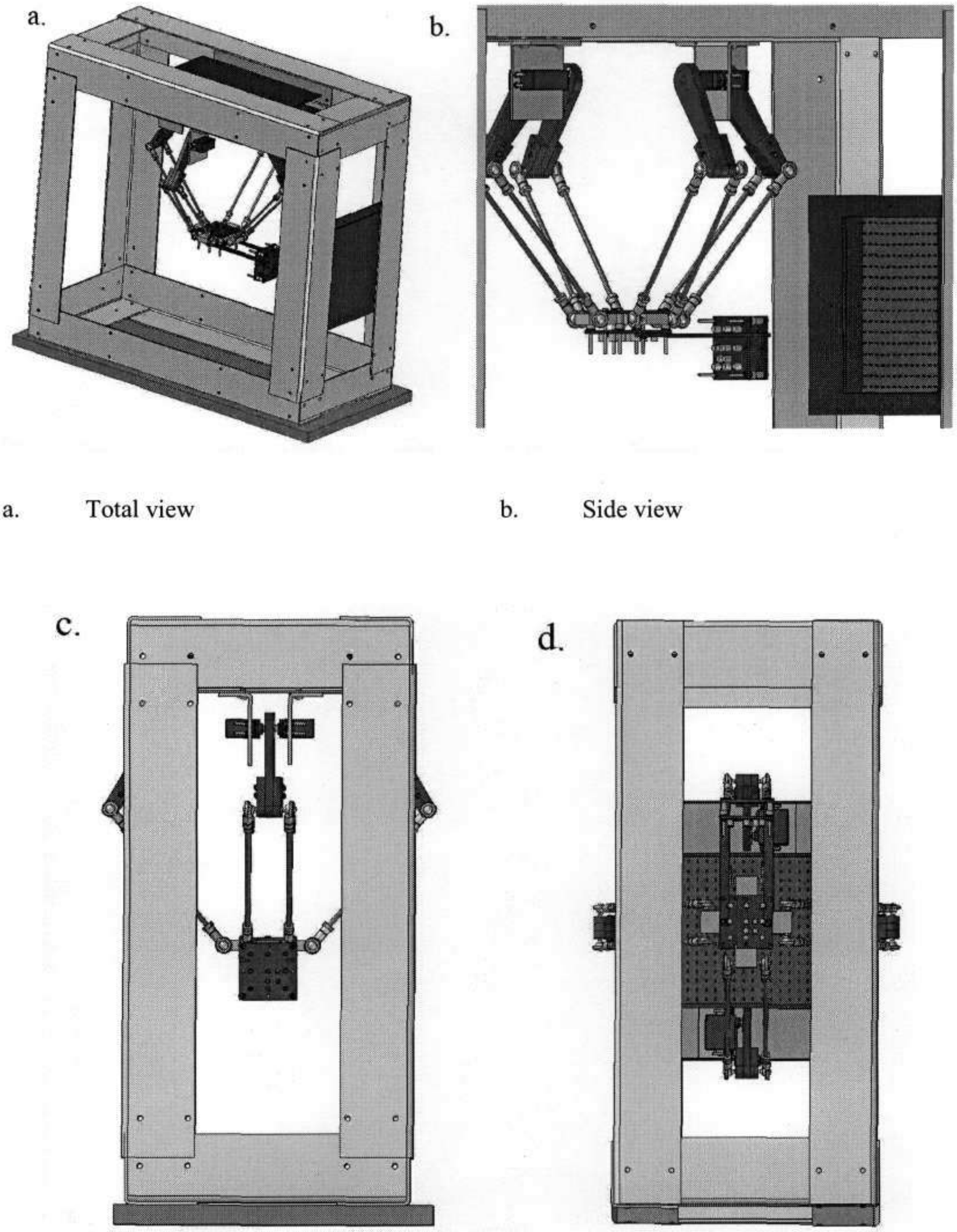


Figure 17 Mounting of servo motors and assembly of arms

a. Servo motors and upper arms mounted on inertial frame
b. Lower arms and end effector attached to upper arms
c. Knee joint
d. Ankle joint

Figure 18 Complete assembly

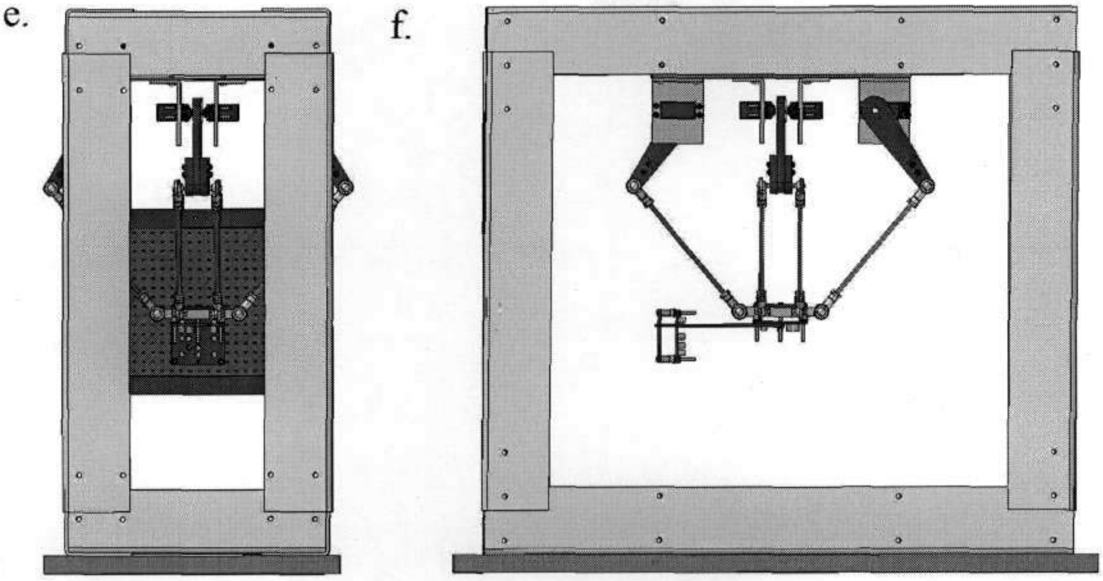


a. Total view

b. Side view

c. Front view

d. Bottom view of end effector and horizontal detector screen (hidden base)



e. Back view of end effector and vertical detector screen

f. Left side view

It must be noted that the lower leg components are held together via 2 springs (not shown), one just below the 'knee' and the other just above the 'ankle' for each leg (see Figure 19 for these joint labels). The ball cup joints give a large degree of freedom. These were made from ball in socket bearings. The upper legs swing from side to side whereas the lower legs can move up, down, left and right and can rotate about the 'knee' by sequencing sets of its basic motion (induced by rotating pairs of servos each to particular angles). The laser can move about a volume of space, which is roughly a hemisphere below the sensitivity area, the square cut-out on the servo mounting frame in Figure 17 a.

The frame work was made from angled aluminium (long bar of L shape aluminium). The lower legs and laser mounting were made from a stiff hard plastic. The lower legs were made from stainless steel rods used to construct model helicopters and aeroplanes. The ball and socket joints were made of steel. These materials gave the Flex-Picker model sufficient stiffness for all movement during testing.

Machine Topology

As this PKM is symmetric it may be described as 3-DOF 4 RSS (3 degrees of freedom, all translational with 4 branches containing a rotational actuator and 2 spherical joints).

Actuator Positioning

The actuators are located on the same plane and are arranged to form a cross. They are positioned such that the plane of rotation of each servo lies coincident with that of the servo directly opposite it, and the adjacent rotation planes are at 90° to each other.

4.3 Dimensional Synthesis

4.3.1 Dimensioning

Table 3 Link lengths

Lengths		Spacing	
Upper Arm	100 mm	Between lower arms	30 mm
Lower arm	179 mm	Between opposing servos	200 mm
End effector (Cross)	100 mm		

4.3.2 System Modelling

The system model is illustrated graphically in Figures 19 to 22.

Parallel structures suffer from some weaknesses which have to be analyzed and taken into account when designing the mechanisms. These are:

- The existence of critical points in the workspace where the mechanism loses the ability to change its position in a prescribed manner or to react to a given load. These are known as points of singular configuration.
- A limited work volume in comparison with that of serial manipulators.
- The increased computational effort necessary to control a parallel manipulator. [13, 40]

The kinematics geometry of multi-DOF robotic manipulators must be analysed to determine the positions and orientations of all the members of the mechanism. This is to avoid the pitfalls mentioned above as the device goes through its motions. This position analysis can be formulated, but is difficult to solve for certain machine configurations. The difficulty arises due to the fact that the kinematics analysis depends on solutions to sets of nonlinear equations. There are two types of kinematics problems for every robotic manipulator, i.e. the forward and inverse kinematics. [41]

For both problems the kinematics structure with its parameters are defined. (**Structure** - the number of links, the types of joints, the connectivity graph; **Parameters** - each link's twist and length, and the fixed lengths or angles between neighbouring links)

The forward kinematics (FK) problem in addition to the above has a full set of actuation parameters and aims to determine the position and orientation of the end effector. (**Actuation Parameters** - the actively controlled joint variables: angles for revolute joints and linear displacement for prismatic joints)

In the inverse kinematics (IK) problem the situation is the opposite, here the end effector's position and orientation are given and the objective is to find the set of actuation parameters that will satisfy the kinematics configuration. This leaves a set of nonlinear equations that have to be solved to obtain the actuated variables. This nonlinearity expresses the fact that generally there are multiple sets of values for the actuated variables that will produce exactly the same end effector pose, i.e. multiple solutions for a single end effector position. [41, 42]

It is the IK problem that is of interest to the control systems designer of any robotic manipulator. The control system needs to move the end effector to specific points in its workspace to carry out a task. These points are known. The requirement is the set of actuation values that would follow a trajectory from its current location, avoiding any obstacles in its path to the point of interest.

The equations that describe the direct and inverse problems are the same. The IK is a first example of the geometrical duality between serial and parallel manipulators. The IK for a parallel manipulator (with an arbitrary number of legs) has a unique solution (if each serial leg has a unique solution), and can be calculated immediately. These are properties of the FK of the general serial manipulator. [42]

a. Forward Kinematics (FK) Problem

For serial mechanisms the FK problem can be solved without any difficulty. The relative position and orientation of each link is dependent on the previous link, and so it can be obtained as the result of vector addition, matrix multiplication, or some analogous deterministic operations. For parallel mechanisms, the relative position and orientation of some links depend on more than one other link, so the FK problem leads to a set of nonlinear equations. Hence, for parallel mechanical systems, the FK problem (i.e., where all the actuator values are given and the requirement is the end effector's pose) is more difficult to solve than its IK. [41]

Solution Methods

Many solutions to the forward kinematics problem for parallel mechanisms deal with particular architectures or small classes of architectures, sometimes under hypothesis of geometrical symmetries. The lack of general explicit solutions occurs even in the simpler case of pure translational motion of the end effector. [43]

A variety of solution methods have been developed for solving the sets of nonlinear polynomial equations that arise in the inverse problem for series chains and the direct problem for parallel systems. The methods that have proven to be the most useful have been based on polynomial continuation, elimination methods or Gröbner bases. [41]

Polynomial continuation is a numerical method that is useful in solving problems for actual numerical values and running numerical experiments. It does not offer any direct assistance in general studies involving parameters on a symbolic level. For kinematics analyses it is necessary to have solution methods that give all possible solutions to a particular set of nonlinear equations. While numerical methods, such as Newton-Raphson, converge to a single solution, the polynomial continuation method is a numerical procedure that can find all solutions to a given problem. "The idea is that small changes in the coefficients of a system lead to small changes in the solutions. Using this idea and having a system whose initial solutions are known, it is possible to gradually transform the system to find the solutions that are required. During this transformation, all solutions are tracked, so that in the end all the solutions to the final system are found." [41]

The advantages of polynomial continuation are its ability to solve very large systems, and the fact that the procedure itself need not be modified for different polynomial systems. Also it virtually guarantees that all solutions to a system will be found, assuming there are no numerical anomalies.

The disadvantage of polynomial continuation is mainly speed. For many kinematics problems, the number of paths that must be tracked can be large enough that the continuation calculation is too slow for real-time control or other applications where speed is important. [41]

Elimination methods, sometimes called resultant methods, are based on an algebraic formulation that allows for the elimination of a large number of variables in one single step, and reduces a set of nonlinear equations to a single polynomial in one unknown. They also allow for studies of solution properties on a symbolic rather than a numerical basis and require much more algebraic manipulation than continuation methods. A basic example illustrates the construction procedure more adequately, see source [41] for a detailed example.

If an elimination based solution method can be found for a particular problem, it normally leads to much faster computation times than polynomial continuation or Gröbner base methods. The main disadvantage of elimination based methods is finding an appropriate multivariate eliminant for a particular problem.

Numerous authors have used eliminant methods to solve the direct kinematics problems for Stewart-Gough platforms with some special geometric constraints, such as concentric spherical joints. All the algorithms to solve the general case of the Stewart-Gough platform's direct kinematics problem and obtain a 40th-degree univariate polynomial, use elimination. [41]

Gröbner bases is an iterative algebraic variable elimination technique for solving sets of nonlinear equations. Gröbner bases has recently proven to be very useful in conjunction with elimination methods. The basic elimination procedure resembles Gaussian elimination in that it produces a triangular system of equations. For the Gröbner bases technique, the last equation is a univariate polynomial, and each subsequent equation adds at most one new variable, although the equation may not be linear in that variable. The univariate polynomial may be solved to find all possible values of one unknown, and the other equations will yield the values of the other variables for each solution. The choice of ordering for the polynomial terms in nonlinear equations is not obvious, however a lexicographic ordering will always lead to triangular Gröbner bases.

The disadvantage of the Gröbner bases technique is the computation time needed. Also, the complexity of a given problem is unpredictable. Nevertheless, the technique has proven useful in kinematics analysis, most notably in confirming the number of solutions for the general case of the Stewart-Gough platform's FK. It also aided in determining the characteristic polynomial, as well as in predicting the upper bound on the number of solutions for those special cases where platform legs are required to share pivot locations. [41]

b. Geometric Kinematics Model

The methods to solve the forward and inverse kinematics using complex matrix algebra and Jacobians are difficult to understand. A geometric model, shown in Figure 19, was devised and using common math and trigonometric functions, the forward and inverse kinematics problems were established. These were then solved with rigorous algebraic manipulation of the variables, in those functions, to obtain closed form solutions.

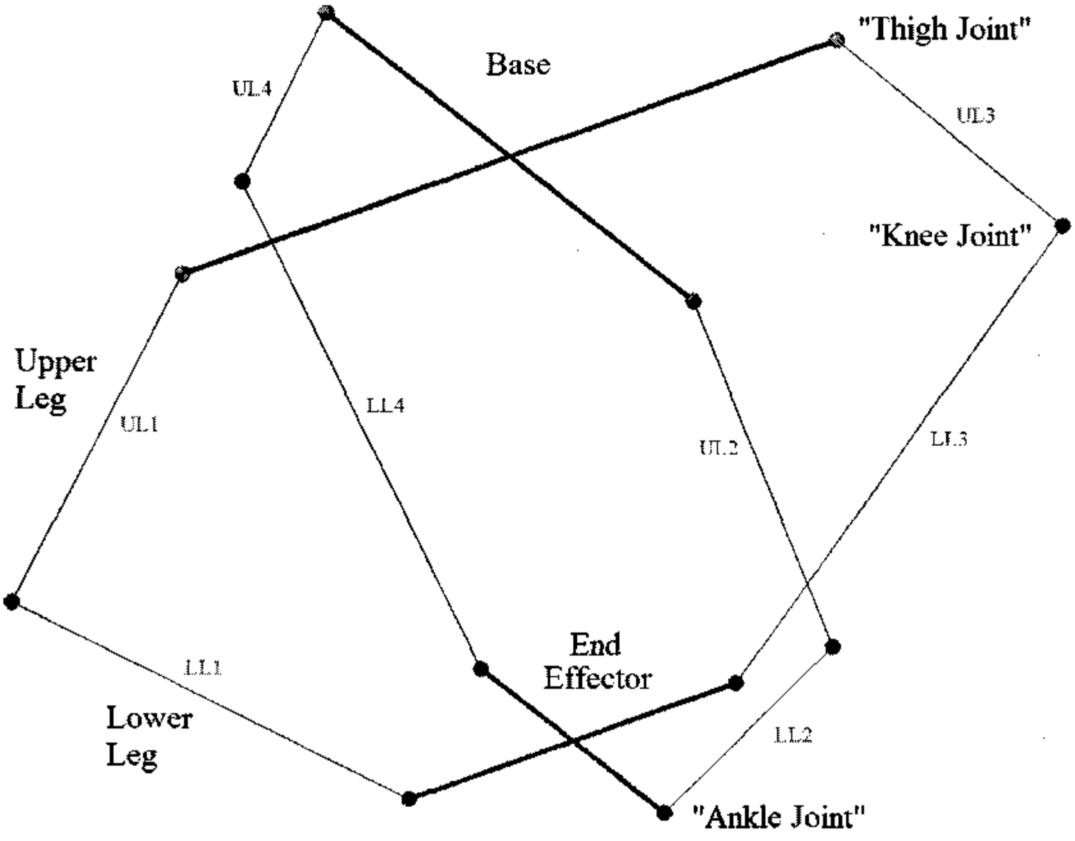


Figure 19 Simplified Geometric Kinematics Model

This geometric model differs from the mechanical system shown in Figure 18. The parallelograms of the lower arms have been collapsed to single lines joining the "knee" to the "ankle". This simplification is acceptable as the parallelograms that compose the lower arms completely restrain the orientation of the end effector; as a result the end effector plane $\{ee\}$ remains parallel to the base plane $\{bs\}$ at all instances of its motion.

This parallel plane constraint is taken into account in the model with a few points as listed ...

- The origin of $\{ee\}$ is the "dead" centre of the end effector.
- The coordinates of the "ankle" joints for each leg i (i = 1,...,4) on the end effector are known relative to the origin of $\{ee\}$. These are fixed distances from the origin of $\{ee\}$ determined at design.
- The origin of $\{bs\}$ is the "dead" centre of the base.
- The coordinates of the "thigh" joints for each leg i (i = 1,...,4) on the base are known relative to the origin of $\{bs\}$. These are fixed distances from the origin of $\{bs\}$ determined at design.
- The upper legs are restrained to have rotational motion about a plane, i.e. Upper legs 1 and 3 move in plane y = 0, the rotation axes of T_1 and T_3 are perpendicular to the plane y = 0. Upper legs 2 and 4 move in the plane x = 0, the rotation axes of T_2 and T_4 are perpendicular to the plane x = 0.
- The lower legs have complete spatial motion.

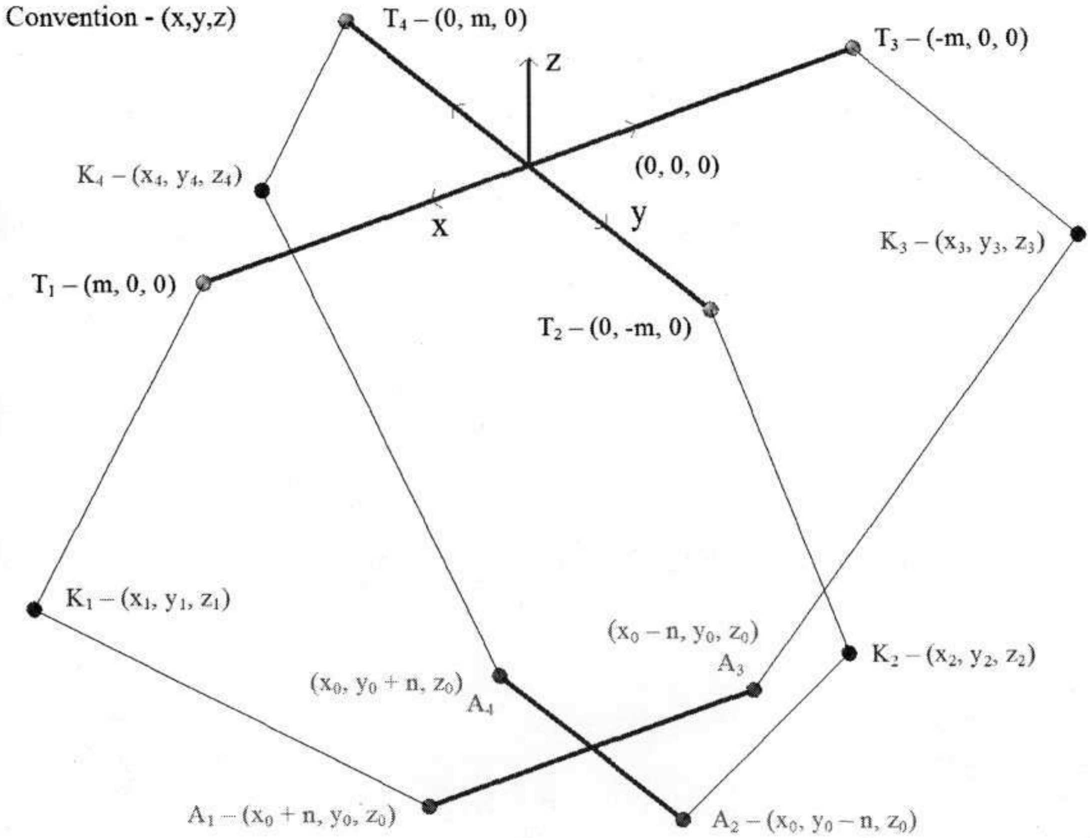


Figure 20 Illustration of joint labels and coordinates

The coordinates of the critical points are:

- $\{bs\}$ origin - $(0; 0; 0)$
- origin of $\{ee\}$ relative to $\{bs\}$ is $(x_0 ; y_0; z_0)$

Leg 1: $T_1 - (m;0;0)$	$K_1 - (x_1; y_1; z_1)$	$A_1 - (x_0 + n; y_0; z_0)$
Leg 2: $T_2 - (0;-m;0)$	$K_2 - (x_2; y_2; z_2)$	$A_2 - (x_0; y_0 - n; z_0)$
Leg 3: $T_3 - (-m;0;0)$	$K_3 - (x_3; y_3; z_3)$	$A_3 - (x_0 - n; y_0; z_0)$
Leg 4: $T_4 - (0;m;0)$	$K_4 - (x_4; y_4; z_4)$	$A_4 - (x_0; y_0 + n; z_0)$

- m And n are design constraints, indicating the relative displacement of the revolute and spherical joints from the centres of $\{bs\}$ and $\{ee\}$ along the x and y axes

- All sphere radii have magnitude R_0 (see following sub-section)
- All circle radii have magnitude R_1 (see following sub-section)
- From the dimensional synthesis of section 4.3.1 Dimensioning, the variables mentioned above have values:

  m = 10 cm; n = 4.45 cm; R_0 = 17.9 cm; R_1 = 10 cm.

c. Geometric Approach to Solving the FK

The FK problem provides the actuation angles for each motor and then requires the end effector position in space. This problem is easier to solve with serial kinematics manipulators but far more difficult for PKMs, and the difficulty depends on the complexity of the machines legs.

For this mechanical design, seeing as the actuation angles are known for the forward kinematics, the coordinates of the knee joints can be readily calculated.

To solve the FK some knowledge of the mechanical constraints of the system, have to be used. In particular the fact that the end effector plane $\{ee\}$ will always remain parallel to the base plane $\{bs\}$ is critical, this reduces the number of unknowns from 6 (having any position and orientation in space) to just 3 (only position).

The knee joint coordinates are known, each of x_i, y_i and z_i $(i = 1...4)$ can be calculated.

- $K_1 - (x_1; y_1; z_1)$, $y_1 = 0$
- $K_2 - (x_2; y_2; z_2)$, $x_2 = 0$
- $K_3 - (x_3; y_3; z_3)$, $y_3 = 0$
- $K_4 - (x_4; y_4; z_4)$, $x_4 = 0$

The ankle joint coordinates are unknown, x_0, y_0 and z_0 have to be solved.

- $A_1 - (x_0 + 4.45; y_0; z_0)$
- $A_2 - (x_0; y_0 - 4.45; z_0)$

- $A_3 - (x_0 - 4.45; y_0; z_0)$
- $A_4 - (x_0; y_0 + 4.45; z_0)$

A_i must lie on a sphere centred at K_i, for i = 1,:,4. The equation of a sphere is given by $(x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2 = R^2$, for a point a $(x_a; y_a; z_a)$ lying on a sphere of radius R with centre b $(x_b; y_b; z_b)$. This is shown in Figures 21 and 22.

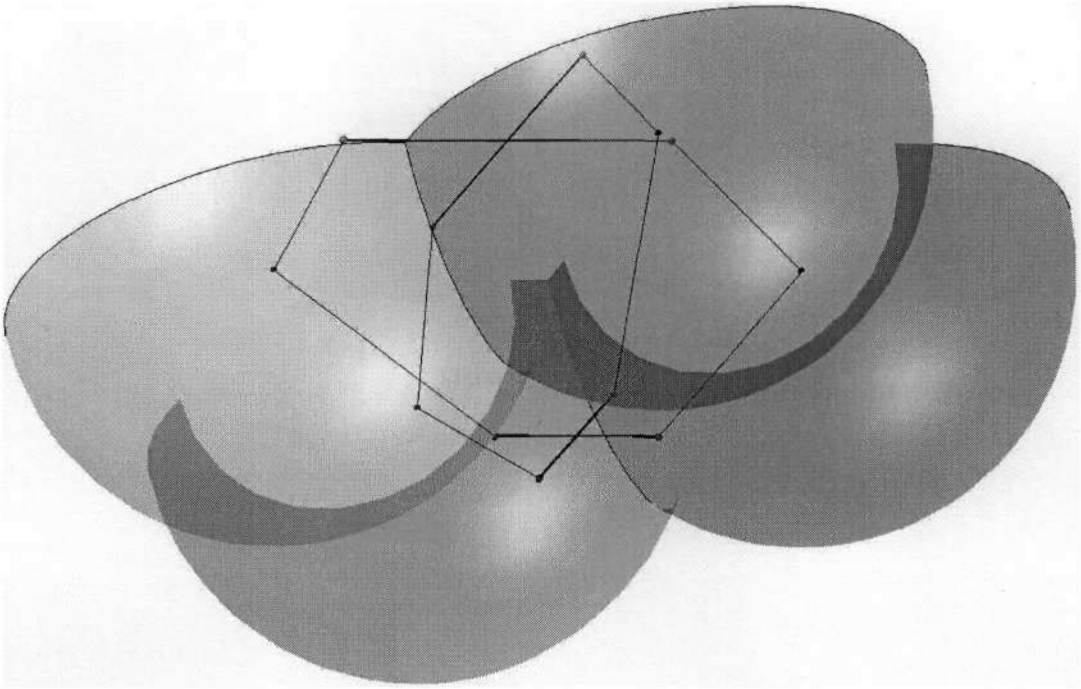


Figure 21 Four hemispheres each centred on a knee joint

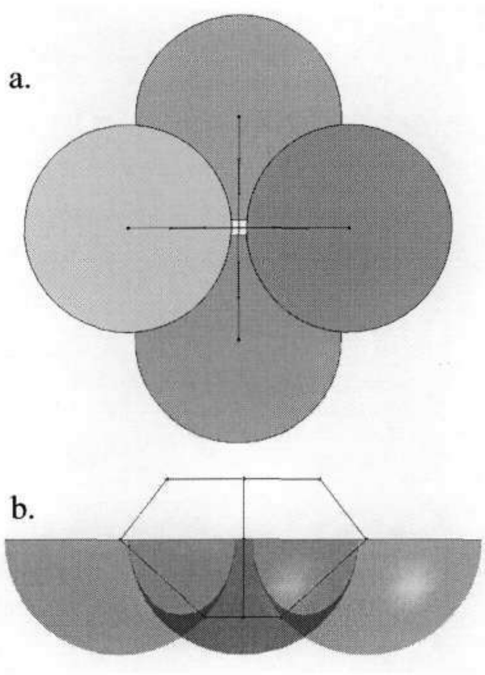


Figure 22 Top and side views of spheres used to solve FK

a. Top view b. Side view

Solving for (x_0, y_0, z_0) requires rigorous algebraic manipulation of these sphere equations. It is however much easier to understand when compared to traditional methods as discussed in a previous section.

Solving for x_0:

$$(x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2 = R^2$$

Sphere equation for leg 1:

$$((x_0 + 4.45) - x_1)^2 + (y_0 - y_1)^2 + (z_0 - z_1)^2 = 17.9^2 = 320.41, \qquad y_1 = 0$$

$$\Rightarrow \quad (x_0^2 + 8.9x_0 + 19.803 - 2x_1x_0 - 8.9x_1 + x_1^2) + y_0^2 + (z_0^2 - 2z_1z_0 + z_1^2) = 320.41$$

Rearranging the equation and factoring yields:

$$\Rightarrow \quad x_0(8.9 - 2x_1) - 2z_1z_0 + (-8.9x_1 + x_1^2 + z_1^2) = 300.607 - x_0^2 - y_0^2 - z_0^2$$

Making the following substitutions:

$$C_0 = 300.607 - x_0{}^2 - y_0{}^2 - z_0{}^2$$
$$C_1 = 8.9 - 2x_1$$
$$C_2 = -8.9x_1 + x_1{}^2 + z_1{}^2$$

C_1 And C_2 are constants whereas C_0 is variable. Hence the first equation reduces to:

$$C_0 = C_l x_0 - 2z_l z_0 + C_2 \qquad \ldots \qquad (4.1)$$

Sphere equation for leg 3:

$$((x_0 - 4.45) - x_3)^2 + (y_0 - y_3)^2 + (z_0 - z_3)^2 = 17.9^2 = 320.41, \qquad y_3 = 0$$
$$\Rightarrow \quad (x_0{}^2 - 8.9x_0 + 19.803 - 2x_3x_0 + 8.9x_3 + x_3{}^2) + y_0{}^2 + (z_0{}^2 - 2z_3z_0 + z_3{}^2) = 320.41$$

Rearranging the equation and factoring yields:

$$\Rightarrow \quad x_0(-8.9 - 2x_3) - 2z_3z_0 + (8.9x_3 + x_3{}^2 + z_3{}^2) = 300.607 - x_0{}^2 - y_0{}^2 - z_0{}^2$$

Making the following substitutions:

$$C_3 = -8.9 - 2x_3$$
$$C_4 = 8.9x_3 + x_3{}^2 + z_3{}^2$$

C_3 And C_4 are constants. Hence the second equation reduces to:

$$C_0 = C_3x_0 - 2z_3z_0 + C_4 \qquad \ldots \qquad (4.2)$$

Clearly equation (4.1) equals equation (4.2).

$$\mathrm{C_1x_0 - 2z_1z_0 + C_2 = C_3x_0 - 2z_3z_0 + C_4 = C_0}$$

Solving for $\mathrm{x_0}$ in terms of $\mathrm{z_0}$ yields:

$$\Rightarrow \quad (C_1 - C_3)x_0 = 2(z_1 - z_3)z_0 + (C_4 - C_2)$$
$$\Rightarrow \quad x_0 = \frac{2(z_1 - z_3)z_0 + (C_4 - C_2)}{C_1 - C_3} \qquad \ldots \qquad (4.3)$$

Solving for y_0:

Sphere equation for leg 2:

$$(x_0 - x_2)^2 + ((y_0 - 4.45) - y_2)^2 + (z_0 - z_2)^2 \;=\; 17.9^2 \;=\; 320.41, \qquad x_2 = 0$$

$$\Rightarrow \qquad x_0^{\,2} + \left(y_0^{\,2} - 8.9y_0 + 19.803 - 2y_2y_0 + 8.9y_2 + y_2^{\,2}\right) + \left(z_0^{\,2} - 2z_2z_0 + z_2^{\,2}\right) \;=\; 320.41$$

Rearranging the equation and factoring yields:

$$\Rightarrow \qquad y_0(-8.9 - 2y_2) - 2z_2z_0 + \left(8.9y_2 + y_2^{\,2} + z_2^{\,2}\right) \;=\; 300.607 - x_0^{\,2} - y_0^{\,2} - z_0^{\,2}$$

Making the following substitutions:

$$C_5 \;=\; -8.9 - 2y_2$$

$$C_6 \;=\; 8.9y_2 + y_2^{\,2} + z_2^{\,2}$$

C_5 And C_6 are constants. Hence the third equation reduces to:

$$C_0 = C_5y_0 - 2z_2z_0 + C_6 \qquad \ldots \qquad (4.4)$$

Sphere equation for leg 4:

$$(x_0 - x_4)^2 + ((y_0 + 4.45) - y_4)^2 + (z_0 - z_4)^2 \;=\; 17.9^2 \;=\; 320.41, \qquad x_4 = 0$$

$$\Rightarrow \qquad x_0^{\,2} + \left(y_0^{\,2} + 8.9y_0 + 19.803 - 2y_4y_0 + 8.9y_4 + y_4^{\,2}\right) + \left(z_0^{\,2} - 2z_4z_0 + z_4^{\,2}\right) \;=\; 320.41$$

Rearranging the equation and factoring yields:

$$\Rightarrow \qquad y_0(8.9 - 2y_4) - 2z_4z_0 + \left(-8.9y_4 + y_4^{\,2} + z_4^{\,2}\right) = 300.607 - x_0^{\,2} - y_0^{\,2} - z_0^{\,2}$$

Making the following substitutions:

$$C_7 \;=\; 8.9 - 2y_4$$

$$C_8 \;=\; -8.9y_4 + y_4^{\,2} + z_4^{\,2}$$

C_7 And C_8 are constants. Hence the fourth equation reduces to:

$$C_0 = C_7 y_0 - 2z_4 z_0 + C_8 \qquad \ldots \qquad (4.5)$$

Equation (4.4) equals equation (4.5).

$$C_5 y_0 - 2z_2 z_0 + C_6 \;=\; C_7 y_0 - 2z_4 z_0 + C_8 \;=\; C_0$$

Solving for y_0 in terms of z_0 yields:

$$\Rightarrow \quad (C_5 - C_7) y_0 \;=\; 2(z_2 - z_4) z_0 + (C_8 - C_6)$$
$$\Rightarrow \quad y_0 \;=\; \frac{2(z_2 - z_4) z_0 + (C_8 - C_6)}{C_5 - C_7} \qquad \ldots \qquad (4.6)$$

Both x_0 and y_0 are expressed in terms of z_0 :

$$x_0 \;=\; \frac{2(z_1 - z_3)}{C_1 - C_3} z_0 + \frac{(C_4 - C_2)}{C_1 - C_3}$$
$$y_0 \;=\; \frac{2(z_2 - z_4)}{C_5 - C_7} z_0 + \frac{(C_8 - C_6)}{C_5 - C_7}$$

Make the following substitutions:

$$m_1 \;=\; \frac{2(z_1 - z_3)}{C_1 - C_3}; \qquad n_1 \;=\; \frac{(C_4 - C_2)}{C_1 - C_3}$$
$$m_2 \;=\; \frac{2(z_2 - z_4)}{C_5 - C_7}; \qquad n_2 \;=\; \frac{(C_8 - C_6)}{C_5 - C_7}$$

$$x_0 \;=\; m_1 z_0 + n_1$$
$$y_0 \;=\; m_2 z_0 + n_2 \qquad \ldots \qquad (4.7)$$

Substitute equation set (4.7) into the sphere equation for leg 1:

$$((x_0 + 4.45) - x_1)^2 + (y_0 - y_1)^2 + (z_0 - z_1)^2 \;=\; 17.9^2 \;=\; 320.41, \;\; y_1 = 0$$
$$\Rightarrow \quad \left(x_0^{\,2} + 8.9x_0 + 19.803 - 2x_1 x_0 - 8.9x_1 + x_1^{\,2}\right) + y_0^{\,2} + \left(z_0^{\,2} - 2z_1 z_0 + z_1^{\,2}\right) \;=\; 320.41$$

$$\Rightarrow \left((m_1 z_0 + n_1)^2 + 8.9(m_1 z_0 + n_1) + 19.803 - 2x_1(m_1 z_0 + n_1) - 8.9x_1 + x_1^2\right) + (m_2 z_0 + n_2)^2 + \ldots$$
$$\left(z_0^2 - 2z_1 z_0 + z_1^2\right)$$

$$\Rightarrow m_1^2 z_0^2 + n_1^2 + 19.803 + 2m_1 z_0 n_1 + 8.9m_1 z_0 + 8.9n_1 - 2x_1 m_1 z_0 - 2x_1 n_1 - 8.9x_1 + x_1^2 + m_2^2 z_0^2 + \ldots$$
$$2m_2 z_0 n_2 + n_2^2 + z_0^2 - 2z_1 z_0 + z_1^2 \quad = \quad 320.41$$

Rearranging and simplifying:

$$\Rightarrow \left(m_1^2 + m_2^2 + 1\right) z_0^2 + (2m_1 n_1 + 8.9m_1 - 2x_1 m_1 + 2m_2 n_2 - 2z_1) z_0 + \ldots$$
$$\left(n_1^2 + 8.9n_1 - 2x_1 n_1 - 8.9x_1 + x_1^2 + n_2^2 + z_1^2 - 300.607\right) \; = \; 0$$

This is a quadratic in z_0, i.e. $a z_0^2 + b z_0 + c \; = \; 0$.

Therefore $z_0 \; = \; \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$, with:

$a \; = \; m_1^2 + m_2^2 + 1;$

$b \; = \; 2m_1 n_1 + 8.9m_1 - 2x_1 m_1 + 2m_2 n_2 - 2z_1$ and:

$c \; = \; n_1^2 + 8.9n_1 - 2x_1 n_1 - 8.9x_1 + x_1^2 + n_2^2 + z_1^2 - 300.607$.

There are 2 sets of solutions for $(x_0; y_0; z_0)$, and both are real as there are 2 possible values for z_0. The correct configuration or solution for the end effector coordinates has z_0 more negative than the z coordinates of the knee joints.

d. Calculating knee coordinates from the actuation angle

Leg 1: Resolving (x_1, y_1, z_1):

The first thing to note is that the knee coordinate (x_1, y_1, z_1) lies on a circle centred at the "thigh joint" for leg 1, this is indicated by the red dashed line of Figure 23. It also lies on the straight line passing through that "thigh joint" with a gradient given by the angle θ_1 with regard to the fixed frame of reference, indicated the thick solid black line of Figure 23. The positions of the motors were mirrored about a centre line in the mechanical design to make the machine symmetric. The coordinate systems and references for each leg are the same, however, to make calculations uniform. The angle θ_1 in the calculations (indicated by the red counter clockwise arrow) differs from the actuation angle θ_{R1} (indicated by the green arrow –

range 0° to 180°, clockwise). The angle θ_1 is with respect to the coordinate system angular frame of reference, and θ_{R1} is the rotation angle with respect to the servo and its mounting (solid green line). A transform is used to obtain θ_1 from θ_{R1} (θ_1 is used in the FK to determine end effector position coordinates). As the servo motors have a 180° limit on their rotation, the limits on θ_1 are from 45° to 225° (clockwise).

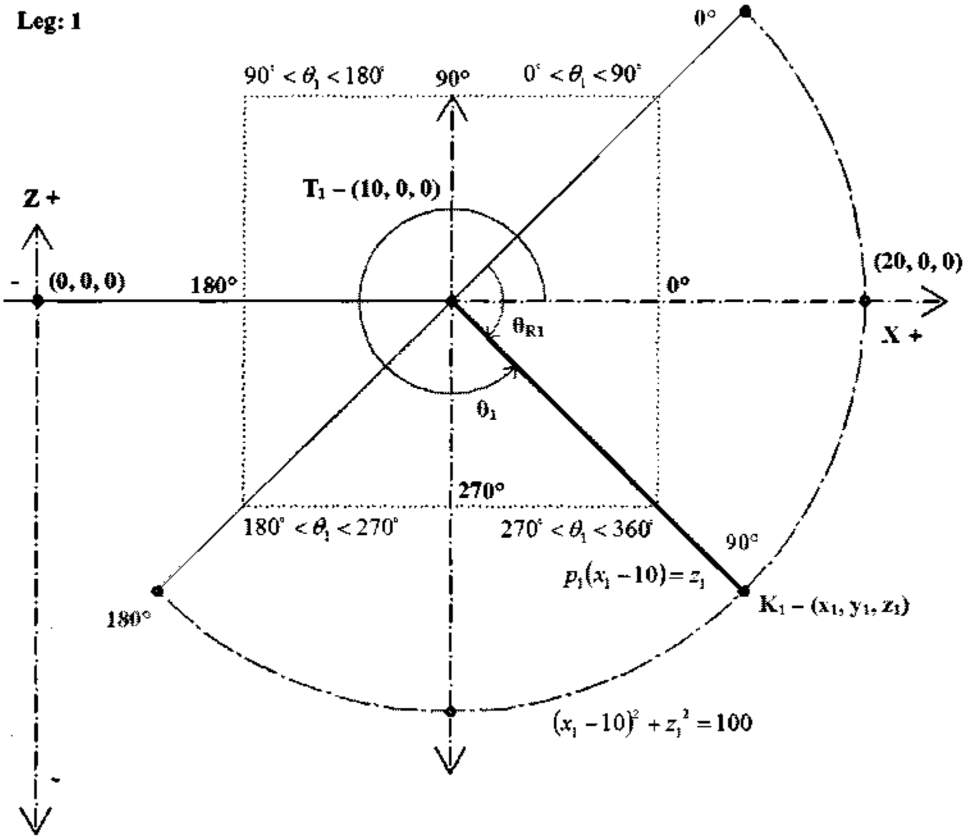


Figure 23 **Illustration of leg 1 coordinate frame and angular conventions**

The transform used to obtain θ_1 is:

$$\theta_1 = (405° - \theta_{R1}) \bmod 360; \quad 0° \le \theta_{R1} \le 180°, \; 225° \le \theta_1 \le 360° \;\cup\; 0° \le \theta_1 \le 45°$$

A few example calculations:

For $\theta_{R1} = 0°$: $\theta_1 = 405° \bmod 360 = 45°$

For $\theta_{R1} = 90°$: $\theta_1 = (405° - 90°) \bmod 360 = 315° \bmod 360 = 315°$

For $\theta_{R1} = 180°$: $\theta_1 = (405° - 180°) \bmod 360 = 225° \bmod 360 = 225°$

This transform is also used to obtain θ_4 from θ_{R4}. The transform for θ_2 and θ_3 is different and is given by $\theta_i = 135° + \theta_{Ri}$, $i = 2, 3$.

Once the actuation angles are obtained in the global reference system, the "knee" coordinates for each leg can be evaluated. This is illustrated for leg 1.

Equation of straight line (upper leg): $p_1(x_1 - 10) = z_1$ (when $x_1 = 10, z_1 = 0$ and p_1 is the gradient of the line, i.e. $p_1 = \tan\theta_1$.)

Equation of circle: $(x_1 - 10)^2 + z_1^2 = 100$

$$= \qquad x_1^2 - 20x_1 + 100 + z_1^2 = 100$$

$$\Rightarrow \qquad x_1^2 - 20x_1 + z_1^2 = 0$$

$$\Rightarrow \qquad x_1^2 - 20x_1 + (\tan\theta_1(x_1 - 10))^2 = 0$$

$$\Rightarrow \qquad x_1^2 - 20x_1 + x_i^2\tan^2\theta_i - 20x_i\tan^2\theta_i + 100\tan^2\theta_1 = 0$$

$$\Rightarrow \qquad x_1^2(1 + \tan^2\theta_1) - 20x_1(1 + \tan^2\theta_1) + 100\tan^2\theta_1 = 0$$

This is a quadratic in x_1, and using the binomial formula to resolve x_1 yields:

$$\Rightarrow \qquad x_1 = \frac{20(1 + \tan^2\theta_1) \pm \sqrt{400(1 + \tan^2\theta_1)^2 - 400\tan^2\theta_1(1 + \tan^2\theta_1)}}{2(1 + \tan^2\theta_1)}$$

$$= 10 \pm \frac{10\sqrt{1 + 2\tan^2\theta_1 + \tan^4\theta_1 - \tan^2\theta_1 - \tan^4\theta_1}}{1 + \tan^2\theta_1}$$

$$= 10 \pm \frac{10}{\sqrt{1 + \tan^2\theta_1}}, \quad \forall \ \theta_1 \neq 90° + k.180°, \quad k \in \mathrm{N}_0 \qquad \ldots \qquad (4.8)$$

This formula provides 2 solutions for x_1, the correct solution is determined from the quadrant of θ_1. If θ_1 is in the first or fourth quadrant $x_1 > 10$ and if θ_1 is in the third quadrant $x_1 < 10$.

See Appendix A for the calculation of knee coordinates for legs 2, 3 and 4, leg coordinate frame figures and their corresponding angular transforms ($\theta_{Ri} \Rightarrow \theta_i$, i = 2, 3, 4).

e. The Inverse Kinematics (IK) Problem

The IK problem can be challenging for serial mechanisms and an example of this can be illustrated using the planar, three revolute joint mechanism shown in Figure 24 a. Given the lengths of all the links, i.e. the structural parameters, and given a specified value for the end effector pose, i.e. the position and orientation of the end effector, the problem is to determine the angles θ_1, θ_2 and θ_3 to get the manipulator into that desired position. In even the most complex series manipulators, the situation is analogous.

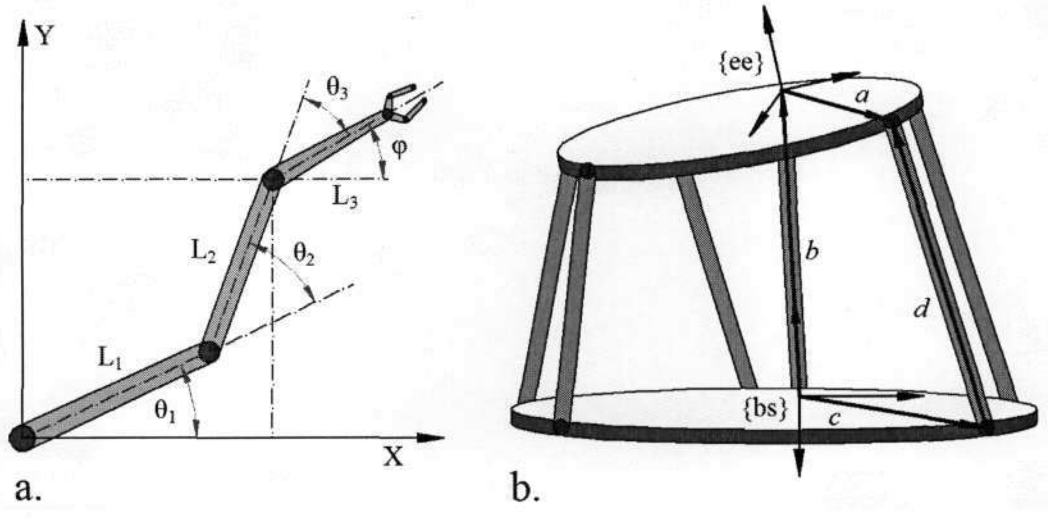


Figure 24 Inverse Kinematics Problem

a. A planar 3-revolute joint mechanism

b. Generic kinematics model for parallel manipulators

For parallel structures, the level of difficulty of the IK problem depends entirely on the complexity of the legs. If the legs are simple, as when each leg consists of two links connected by a prismatic joint, then the inverse kinematics is simple. On the other hand, if the legs are complex, as when each leg is a series chain of five links, each connected to its neighbour by revolute or spherical joints, the problem becomes quite complex. In general most parallel devices are built with simple legs, and therefore it is usual for the inverse problem to be quite simple and straightforward. [41]

f. Solving the IK using the Looping Method

The link frame conventions and transformations defined for serial kinematics chains apply without change to each of the legs in a parallel robot. The only difference with the serial case is the definition used for the connection of all legs to the base and the end effector platforms. Figure 24 b shows the kinematics model that will be used as a generic example. It is not fully generic, in that the base and end effector are arbitrarily chosen to be planar, and only prismatic legs are used with spherical joints at both ends. It is sufficient for illustrating the kinematics loops and loop equations used in solving the IK.

The platforms are rigid bodies, which are represented by the reference frames $\{bs\}$ (base, plane at z = 0) and $\{ee\}$ (end effector, plane at z = z_0), respectively. Parallel manipulators are closed loop mechanisms, by virtue of the fact that by selecting any critical point such as a joint position one can traverse a set of links and joints passing each one only once and return to the joint or point of origin. This is illustrated in Figure 24 b. It will be shown using conventional coordinate and vector notation, as indicated below [42, 44, 45, 46]:

- In Figure 24 b$\{bs\}$ serves as the immobile world reference frame, and $\{ee\}$ the mobile frame of interest,
- The vector a can be written as $P^{EE,i_{EE}}$; it is the vector from the origin of $\{ee\}$ to the connection of the i^{th} leg on the end effector platform,
- The vector d (also denoted by l_i) is a non-unit direction vector along the i^{th} leg, and its length $|l_i|$, equals the current length of the leg,
- The vector c is the vector from the origin of $\{bs\}$ to the connection point of the i^{th} leg on the base platform; it is denoted by the notation $P^{BS,i_{BS}}$,
- The vector b can be written as $P^{BS,EE}$, it connects the origin of $\{bs\}$ to the origin of $\{ee\}$.

For each leg i, the following position closure constraint is always satisfied:

$$P^{BS,i_{BS}} + l_i = P^{BS,EE} + P^{EE,i_{EE}} \quad \forall\, i = 1,\ldots,6 \qquad (4.9)$$

Or by lettered enumeration $c + d = a + b$.

In this equation, $P^{BS,i_{BS}}$ and $P^{EE,i_{EE}}$ are known design constants, so their coordinates are known with respect to $\{bs\}$ and $\{ee\}$, respectively. l_i Is time-varying and usually only its magnitude is measurable, not its direction. $P^{BS,EE}$ Changes with the position and orientation of the end effector platform with respect to the base platform. The matrix $\overline{q} = (\overline{q}_1 \ldots \overline{q}_6)^T$ (containing vectors $\overline{q}_i$) denotes the end effector joint positions of the parallel manipulator. These can be used to calculate leg lengths of prismatic joints, or angles of actuated revolute joints, as the matrix containing the vectors of all base or reference joints is known from design. [42, 44, 45, 46]

The IK is solved as follows:

Step 1: Equation (4.9) immediately yields the vector l_i, since all other vectors in the position closure equation are known when ${}^{EE}_{BS}T$ is known. ${}^{EE}_{BS}T$ is the transform that would yield the current position and orientation of $\{ee\}$ when applied to an initial frame $\{ee\}$ that is coincident to $\{bs\}$. This transform includes a rotation matrix used for orientation and a translation vector used for positioning. In terms of coordinates with respect to the base reference frame $\{bs\}$, this equation gives:

$$\begin{aligned} {}_{BS}l_i &= {}_{BS}P^{BS,EE} + {}_{BS}P^{EE,i_{EE}} - {}_{BS}P^{BS,i_{BS}} \\ &= {}_{BS}P^{BS,EE} + {}^{EE}_{BS}R\,{}_{EE}P^{EE,i_{EE}} - {}_{BS}P^{BS,i_{BS}} \qquad \ldots \qquad (4.10) \end{aligned}$$

${}_{BS}P^{BS,EE}$ (Translation vector) and ${}^{EE}_{BS}R$ (rotation matrix) come from the input ${}^{EE}_{BS}T$, as mentioned. ${}_{EE}P^{EE,i_{EE}}$ And ${}_{BS}P^{BS,i_{BS}}$ are known constant magnitude vectors determined during the design of the manipulator. [42, 44, 45, 46]

Step 2: The length $|l_i|$ is the Euclidean norm:

$$|l_i| = \sqrt{(l_{i,x})^2 + (l_{i,y})^2 + (l_{i,z})^2} \qquad \ldots \qquad (4.11)$$

For the hexapod (or Stewart-Gough design), this length immediately gives the desired position q_i of the actuated prismatic joint for leg i. Other designs require more mathematics to arrive at the values for the actuated joint variables.

g. IK for the Delta Modification using the Geometry Method

The geometric kinematics model used is shown in Figures 19 and 20. This was explained in section 4.3.2 b Geometric Kinematics Model.

In order to determine the rotation values for the actuated revolute joints, from a geometric point of view, position the end effector as desired within its workspace, having $\{ee\}$ parallel to $\{bs\}$. At this point the coordinates of the "ankle" joints on $\{ee\}$ are known. Construct spheres with radii equal to the length of the lower arm centred at the "ankle" joint. Construct circles with radii equal to the length of the upper arm centred on the "thigh" joint. Now for each leg i ($i = 1,\ldots,4$), the intersection of the circle and the sphere result in the coordinates of the "knee" joints. There are 2 intersection points that occur on each leg, from these only the outer coordinates are taken as the required solutions [44, 45]. See Figures 25 and 26 for illustrations.

The Cartesian equation of a sphere centred at the point (a,b,c) with radius R_0 is given by $(x-a)^2+(y-b)^2+(z-c)^2=R_0^{\ 2}$. The Cartesian equation of a circle centred at the point (d,e) with radius R_1 is given by $(x-d)^2+(y-e)^2=R_1^{\ 2}$.

At first glance it may seem that there are 3 variables to solve for at each knee joint, however due to the fact that the circle is completely planar one of those three variables is known and only 2 have to be solved. Essentially there are 2 equations and 2 unknowns and this ensures that this system is solvable, although it requires substantial algebraic manipulation.

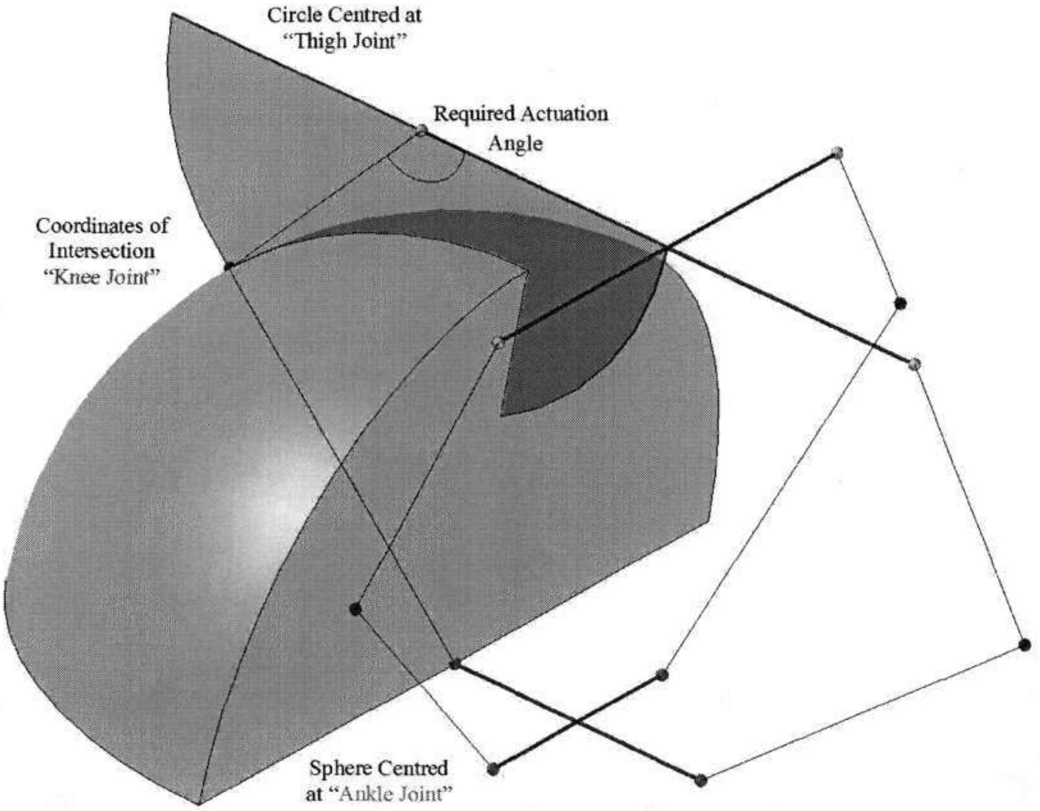


Figure 25 Illustration of sphere-circle intersection

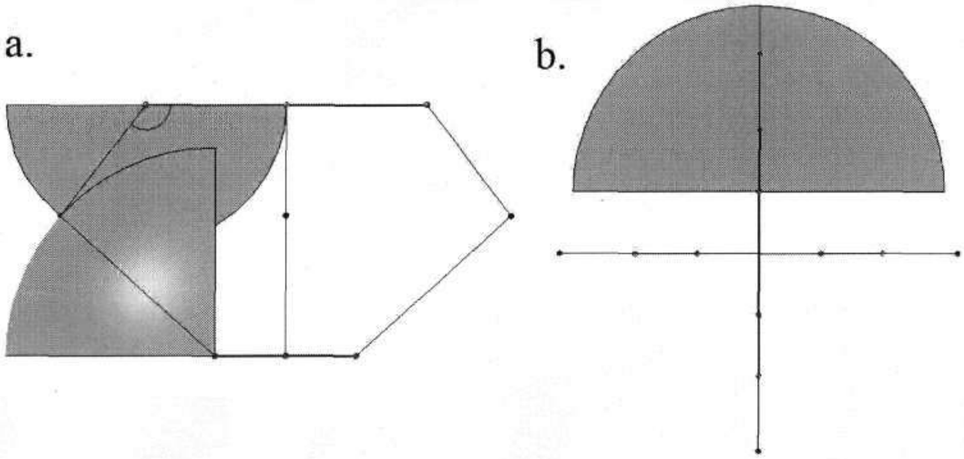


Figure 26 Illustration of sphere-circle intersection

a. Side View

b. Top view

Figure 20 illustrates the joint labels and coordinates for each leg. Leg i contains "thigh" joint T_i, "knee" joint K_i and "ankle" joint A_i for i = 1,…,4.

Now the leg equations are:

Leg 1:

Sphere...

$$(x_1-(x_0+n))^2+(y_1-y_0)^2+(z_1-z_0)^2=R_0^{\ 2}$$

y_1 is known and is 0, this yields...

$$\Rightarrow \qquad (x_1-(x_0+n))^2+(y_0)^2+(z_1-z_0)^2=R_0^{\ 2}$$

Circle...

$(x_1-m)^2+(z_1)^2=R_1^{\ 2}$

Leg 2:

Sphere...

$$(x_2-x_0)^2+(y_2-(y_0-n))^2+(z_2-z_0)^2=R_0^{\ 2}$$

x_2 is known and is 0, this yields...

$$\Rightarrow \qquad (x_0)^2+(y_2-(y_0-n))^2+(z_2-z_0)^2=R_0^{\ 2}$$

Circle...

$$(y_2-(-m))^2+(z_2)^2=R_1^{\ 2}$$

$$\Rightarrow \qquad (y_2+m)^2+(z_2)^2=R_1^{\ 2}$$

Leg 3:

Sphere...

$$(x_3-(x_0-n))^2+(y_3-y_0)^2+(z_3-z_0)^2=R_0^{\ 2}$$

y_3 is known and is 0, this yields...

$$\Rightarrow \qquad (x_3-(x_0-n))^2+(y_0)^2+(z_3-z_0)^2=R_0^{\ 2}$$

Circle...

$$(x_3-(-m))^2+(z_3)^2=R_1^{\,2}$$

$$\Rightarrow \qquad (x_3+m)^2+(z_3)^2=R_1^{\,2}$$

Leg 4:

Sphere...

$$(x_4-x_0)^2+(y_4-(y_0+n))^2+(z_4-z_0)^2=R_0^{\,2}$$

x_4 is known and is 0, this yields...

$$\Rightarrow \qquad (x_0)^2+(y_4-(y_0+n))^2+(z_4-z_0)^2=R_0^{\,2}$$

Circle...

$$(y_4-m)^2+(z_4)^2=R_1^{\,2}$$

In each of the circle equations the z coordinate can be made the subject of the formula. This allows for the removal of the z variable from the sphere equation through manipulation and substitution. In this case the result would be a quadratic in the remaining variable.

Once the coordinates of the "knee" joints are established, the actuation angles can be calculated as the gradient of each leg can now be found. Inverse trigonometric formulae are then used to obtain the angles.

Solving the Leg Equations for Leg 1:

From the circle equation:

$$(x_1-m)^2+(z_1)^2=R_1^{\,2} \quad \text{where } m=10 \text{ and } R_1=10.$$

$$\Rightarrow \qquad (x_1-10)^2+(z_1)^2 = 10^2$$

$$= \qquad x_1^{\,2}-20x_1+100+z_1^{\,2} = 100$$

$$\Rightarrow \qquad x_1^{\,2}-20x_1+z_1^{\,2}=0$$

$$\Rightarrow \qquad z_1^{\,2}=20x_1-x_1^{\,2} \qquad \ldots \qquad (4.12)$$

This places a restriction on x_1 as $z_1^{\,2}$ is always non-negative, so $20x_1-x_1^{\,2}\geq 0$, which implies that $0 \leq x_1 \leq 20$.

From the sphere equation:

$$\Rightarrow \quad (x_1 - (x_0 + n))^2 + (y_0)^2 + (z_1 - z_0)^2 = R_0^{\,2}$$

$$= \quad (x_1 - x_0 - n)^2 + (y_0)^2 + (z_1 - z_0)^2 = R_0^{\,2} \text{ where } n = 4.45 \text{ and } R_0 = 17.9.$$

$$\Rightarrow \quad (x_1 - x_0 - 4.45)^2 + (y_0)^2 + (z_1 - z_0)^2 = 17.9^2 = 320.14$$

$$= \quad x_1^{\,2} + x_0^{\,2} + 19.803 - 2x_1x_0 - 8.9x_1 + 8.9x_0 + y_0^{\,2} + z_1^{\,2} - 2z_1z_0 + z_0^{\,2} = 320.14 \quad \ldots \quad (4.13)$$

Substituting z_1 from equation (4.12)into (4.13) above yields:

$$\Rightarrow \quad x_1^{\,2} + x_0^{\,2} - 2x_1x_0 - 8.9x_1 + 8.9x_0 + y_0^{\,2} + \left(20x_1 - x_1^{\,2}\right) - 2z_1z_0 + z_0^{\,2}$$

$$= \quad 320.14 \text{ -}19.803 = 300.607$$

$$= \quad x_1(11.1 - 2x_0) - 2z_1z_0 + \left(x_0^{\,2} + 8.9x_0 + y_0^{\,2} + z_0^{\,2}\right) = 300.607$$

Rearrange and make z_1 the subject of the formula:

$$\Rightarrow \quad z_1 = \frac{x_1(11.1 - 2x_0) + \left(x_0^{\,2} + 8.9x_0 + y_0^{\,2} + z_0^{\,2} - 300.607\right)}{2z_0} \quad \ldots \quad (4.14)$$

The only unknowns here are z_1 and x_1, the rest are known. Collecting terms and making the following substitution to ease readability results in:

$$c_1 = \frac{11.1 - 2x_0}{2z_0} \text{ and } c_2 = \frac{x_0^{\,2} + 8.9x_0 + y_0^{\,2} + z_0^{\,2} - 300.607}{\mathbf{2z_0}}$$

$$\Rightarrow \quad z_1 = c_1x_1 + c_2 \quad \ldots \quad (4.15)$$

Squaring both sides of (4.15):

$$\Rightarrow \quad z_1^{\,2} = c_1^{\,2}x_1^{\,2} + 2c_1c_2x_1 + c_2^{\,2}$$

However from equation (4.12) $z_1^{\,2}$ equals $20x_1 - x_1^{\,2}$, which implies:

$$c_1^{\,2}x_1^{\,2} + 2c_1c_2x_1 + c_2^{\,2} = 20x_1 - x_1^{\,2}$$

$$\Rightarrow \quad c_1^{\,2}x_1^{\,2} + x_1^{\,2} + 2c_1c_2x_1 - 20x_1 + c_2^{\,2} = \left(c_1^{\,2} + 1\right)x_1^{\,2} + (2c_1c_2 - 20)x_1 + c_2^{\,2} = 0$$

This is a quadratic in x_1 and using the binomial formula $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ yields both solutions i.e.:

$$x_1 = \frac{-(2c_1c_2 - 20) \pm \sqrt{(2c_1c_2 - 20)^2 - 4(c_1^{\ 2} + 1)c_2^{\ 2}}}{2(c_1^{\ 2} + 1)}$$

$$= \frac{-2c_1c_2 + 20 \pm \sqrt{4c_1^{\ 2}c_2^{\ 2} - 80c_1c_2 + 400 - 4c_1^{\ 2}c_2^{\ 2} - 4c_2^{\ 2}}}{2c_1^{\ 2} + 2}$$

$$= \frac{-2c_1c_2 + 20 \pm \sqrt{400 - 80c_1c_2 - 4c_2^{\ 2}}}{2c_1^{\ 2} + 2} = \frac{-2c_1c_2 + 20 \pm 2\sqrt{100 - 20c_1c_2 - c_2^{\ 2}}}{2c_1^{\ 2} + 2}$$

$$= \frac{-c_1c_2 + 10 \pm \sqrt{100 - 20c_1c_2 - c_2^{\ 2}}}{c_1^{\ 2} + 1}$$

For real solutions to exist the condition $100 - 20c_1c_2 - c_2^{\ 2} \geq 0$ must hold. Since x_1 is now known, having taken into account the restriction of equation (4.12), z_1 can be found by taking the square root of both sides of said equation. The inverse kinematics leg equations for legs 2, 3 and 4 can be found in Appendix B.

There are 2 real solutions giving 2 real configurations of each leg, for a given end effector position. The correct leg configuration must be selected, to acquire the correct angle for actuation. The wrong solution for actuation would mean that the leg is folded inwards instead of outwards, and would imply that it must have passed through a singularity condition, i.e. when the leg is completely folded or completely extended, see section 4.3.2 h for an explanation of singularities.

There are a few possible graphical illustrations for configurations of the leg and their planar projections (XZ or YZ plane) are illustrated in Figure 27. The cases illustrated by C and D are impossible; that is having both solutions lie on the same side of the line $\mathrm{T_iA_i}$. This can be seen immediately as $\mathrm{m} = \mathrm{n}$ and $\mathrm{p} = \mathrm{o}$, and the formation of 2 isosceles triangles. Hence each set of solutions for the knee coordinates, for each leg, must lie on either of line $\mathrm{T_iA_i}$.

The correct solution for legs 1 and 4 lie to the right of line $\mathrm{T_iA_i}$ (i=1, 4), and the correct solutions for legs 2 and 3 lie to the left of line $\mathrm{T_iA_i}$ (i=2, 3) with respect to the geometric model's coordinate system (Figure 27). The equation for line $\mathrm{T_iA_i}$ is found; it is a straight line function of variable x or y. The x or y coordinates, depending on the leg, of the knee joint solution sets are then substituted into this equation yielding 2 z coordinate test values. These values are then checked, with the test just described, to determine if they lie to the left or right of line T_iA_i for leg i and the correct solution is chosen. If the z coordinate equals the 'test' value, that is when both solutions coincide, then there is a singularity (see section 4.3.2 h.) at this end effector position.

Once the correct leg configurations have been determined, the angles for the upper legs can be calculated. To obtain θ_1 three cases need to be resolved (see Figure 27 e). A general explanation will be given for θ_i. θ_i Can lie in 1 of 4 quadrants depending on *i* (for $i = 1,..,4$, see Figure 27 e and f), and its value is given by:

θ_i Lies in first quadrant: $\theta_i = \frac{180°}{\pi}.\arctan\left(\frac{z_i - z_{li}}{q_i - q_{li}}\right)$

θ_i Lies in second quadrant: $\theta_i = 180° - \frac{180°}{\pi}.\arctan\left(\frac{z_i - z_{li}}{q_i - q_{li}}\right)$

θ_i Lies in third quadrant: $\theta_i = 180° + \frac{180°}{\pi}.\arctan\left(\frac{z_i - z_{li}}{q_i - q_{li}}\right)$

θ_i Lies in fourth quadrant: $\theta_i = 360° + \frac{180°}{\pi}.\arctan\left(\frac{z_i - z_{li}}{q_i - q_{li}}\right)$

Where $z_{li} = 0 \; \forall \; i$; $q_i = x_1, y_2, x_3, y_4$; $q_{li} = 10, -10, -10, 10$ for i=1,...4 respectively. These cases are due to the fact that *arctan* yields a principle argument in the range $-\frac{\pi}{2} < \theta_i < \frac{\pi}{2}$. The case when θ_i = 270° ($-\frac{\pi}{2}$) can be seen through inspection and occurs when the centre line of the upper leg lies directly on the negative z axis passing through that particular T_i.

The transforms used to obtain the rotation angles θ_{Ri} are inverse functions of those shown in the FK, when θ_{Ri} was used to obtain θ_i. Interestingly, the inverse function for legs 1 and 4 in the IK is the same as the original function used in the FK, with variables swapped.

For $i = 1, 4$: $\theta_{Ri} = (405° - \theta_i) \bmod 360$

For $i = 2, 3$: $\theta_{Ri} = \theta_i - 135°$

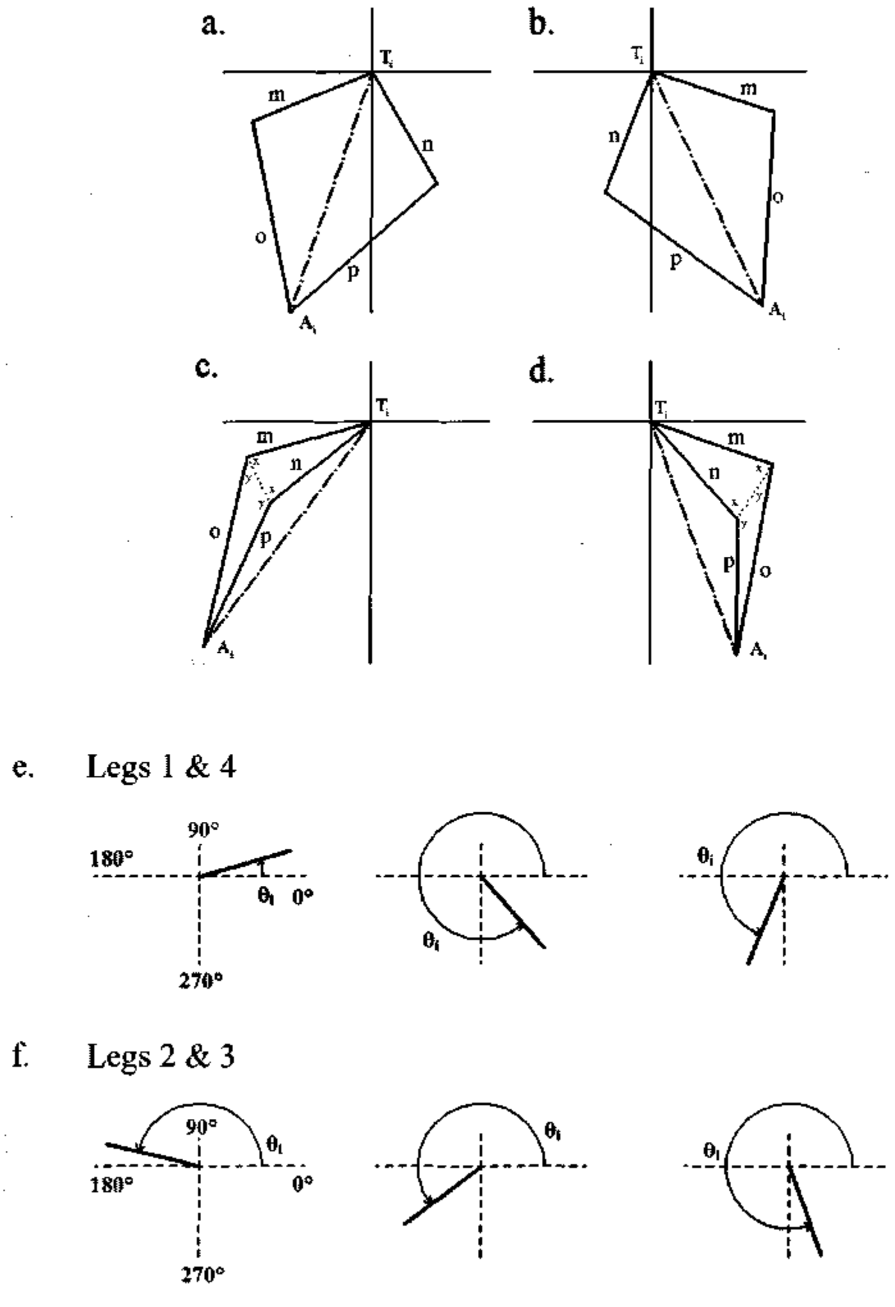


Figure 27 Choosing the correct leg configuration

Leg configuration of projection onto the XZ or YZ plane

a. A_i Lies in the 3rd quadrant, solutions on either side of line A_iT_i

b. A_i Lies in the 4th quadrant, solutions on either side of line A_iT_i

c. A_i Lies in the 3rd quadrant, solutions on same side of line A_iT_i

d. A_i Lies in the 4th quadrant, solutions on same side of line A_iT_i

Leg positions

e. Legs 1 & 4 lie in quadrants 1, 3 and 4

f. Legs 2 & 3 lie in quadrants 2, 3 and 4

h. Singularities

Singularities are an important consideration in the design of parallel manipulators. It was mentioned earlier, but a full explanation is necessary and will now be given.

There are 2 classes of singularities, i.e. architectural and configuration.

Architectural Singularities

An architectural singularity is caused by the design of the manipulator, and not by a specific combination of actuation values for rotation or prismatic joint variables. [42]

Configuration singularities

This singularity would occur for specific values of the actuated joint variables. One example of this is when either the base or the end effector platform is coplanar with one or more legs. In this configuration, the manipulator cannot resist forces orthogonal to the plane of the base or the end effector. As a design rule-of-thumb, it is better not to use planar base or end effector platforms, because that planar relationship introduces partial dependence between several coordinates and increases the possibility of singularities. [42]

There are 3 subclasses that result from configuration space singularities, i.e. inverse, forward and combined.

i. Inverse Singularity

The inverse singularity refers to a specific robot configuration in which the moving platform loses one or more degrees of freedom, instantaneously. [39]

ii. Forward Singularity

The forward singularity configuration refers to a specific robot configuration in which the moving platform gains one or more degrees of freedom, instantaneously. In other words, if all of the active joints are completely locked, the moving platform will still possess infinitesimal motion in certain directions. [39]

iii. Combined Singularity

The combined singularity configuration refers to a specific robot configuration in which the moving platform simultaneously gains and loses one or more degrees of freedom. Hence, the combined singularity occurs if and only if both forward and inverse singularities occur simultaneously. The combined singularity is subject to strict conditions and can be avoided by proper dimension design of the mechanisms. [39]

The importance of singularities, from an engineering perspective, arises for several reasons:

- **Loss of freedom**. A kinematics configuration may result in joints locking in their position. This represents a loss of freedom of one or more degrees.
- **Workspace**. When a manipulator is at a boundary point of its workspace it is at a singular point of its kinematics mapping, though the converse is not the case. Knowledge of these singular points indicates where the manipulator can and cannot move.
- **Loss of control**. A variety of control systems are used for manipulators. Rate control systems require the end effector to traverse a path at a fixed rate and therefore to determine the required joint velocities by means of the inverse of the derivative of the (known) forward kinematics. Near a singularity, this matrix is ill-conditioned and either the control algorithm fails or the joint velocities and accelerations may become unsustainably great. Conversely, force control algorithms, well adapted for parallel manipulators, may result in intolerable joint forces or torques near singularities of the projection onto the joint space.
- **Mechanical advantage**. Near a singular configuration, large movement of joint variables may result in small motion of the end effector. Therefore there is mechanical advantage that may be realised as a load-bearing capacity or as fine control of the end effector. Another aspect of this is in the design of mechanisms possessing trajectories with specific singularity characteristics. In traditional 1-DOF mechanisms (such as the planar 4–bar) a cusp singularity provides 'dwell', where the trajectory is close to stationary for a period of time allowing some process steps in a production to be performed. [38, 47]

i. Delta Mechanism Singularities / Designed PKM Singularities

The PKM designed is a modified delta type mechanism. Since each leg still consists of 2 links and the structure is similar, the singularity conditions for this mechanism are the same as that of a normal Delta PKM (or Flex Picker robot).

The Delta PKM is relatively free of singularities. The ones that occur are readily anticipated, i.e. when a leg is fully extended or completely folded. Due to symmetry these conditions may arise simultaneously in all three legs of the Delta PKM [39, 46].

For the PKM designed the singularities occur when both the lower leg and the upper leg have the same gradient, in the plane of the upper leg rotation. This condition may arise in all 4 legs simultaneously, and this happens when all legs are completely extended or folded (when the "thigh", "knee" and "ankle" joints of the legs are collinear). When calculating the IK this gradient condition must be checked for each leg, and this procedure was outlined in section 4.3.2 g.

Understanding the intrinsic nature of the various types of singularities and their relations with the kinematics parameters and the configuration spaces is of ultimate importance in design, planning and control of the system. [48]

Singularities can never be eliminated, but, as in the case of serial robots, they can be cleverly exploited. Due to their linear dependency on coordinates they can be positioned so that they are either outside of the useful workspace of the robot, or are easier to control. [42]

The approach for this system was to avoid any and all singularities, as those that occur are inverse singularities.

j. Idealized Work Envelope Calculation and Visualisation

The work envelope of the PKM was calculated to provide a visualisation of the workspace. There are 2 possible methods for completing this calculation: using the forward kinematics or the inverse kinematics. Using the forward kinematics there would be 4 nested "for loops", one for each angle (upper leg). Each angle is varied and the forward kinematics calculated for the set of angles, giving the end effector coordinates. This allows calculation of the workspace as well as determination of all singularities within the workspace. However, placing the data in a format acceptable to MATLAB for creating the visualisation is difficult.

The second method using the inverse kinematics solves the difficulty of the previous solution but does not provide singularity information. It, however, was used as it simplified the problem of determining the workspace envelope. There is an intuitive guess for the limits on the XYZ positioning capability. The algorithm starts at the extreme Z positions (a Z min and Z max that cannot be reached for any pair of XY coordinates) in space and calculates the inverse kinematics (solving for the angles of the upper legs for each pair of XY sets). It then works its way inwards to the boundary of the workspace (increasing Z min and decreasing Z max, finding the first Z min and the first Z max that provides actual solutions to the inverse kinematics, for each XY pair). The XY boundary is also set just outside the positioning ability of the robot and the values of X and Y varied (with suitable step) to cover the boundary. This method produces the workspace envelope shown in Figure 28.

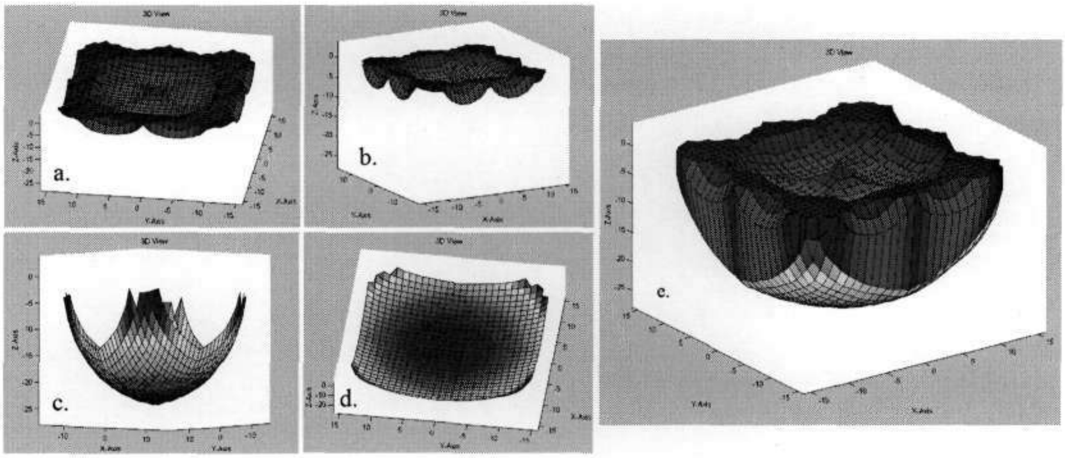


Figure 28 Different views of workspace, top half, bottom half and total.

k. Dynamic Modelling to determine maximum Servo Motor Loading

The dynamic modelling of robotic systems involves the study of motion with regard to the forces that cause it as well as external forces that are applied to the system during parts of that motion. For pick and place robots the force of the added weight of the object that was picked up would affect the machine's dynamics if the ratio of object mass to the system's moving mass is high. This applies to parallel pick and place machines like the Flex-Picker.

Complete dynamic modelling, resolving all forces on all links, of PKMs is sometimes not possible due to the multiple arm structure and the multiple dependencies of the arms on each other. For these systems an approximation is sometimes the only possibility to model some of the dynamics involved. [30]

The first step to establishing the dynamic relations in a mathematical model is to find the centre of masses (COMs) for each of the links in the system including the end effector. These are then superimposed on the geometric kinematics model established earlier.

The COMs for each component was found using the SE CAD software package. The densities of each part are saved in the material properties of the part file. Once the design for the part is complete, the physical properties of COM and centre of volume (COV) are calculated by the CAD package and saved with the design, as this property does not change unless the design is altered. For a composite part like the end effector in this PKM design, its COM is calculated from the COMs of is composing parts. The COMs of all moving parts are shown in Figure 29.

All that is required are the COMs for the end effector, the lower legs and the upper legs. These are illustrated in Figure 29, which are represented by significant points on the geometric model.

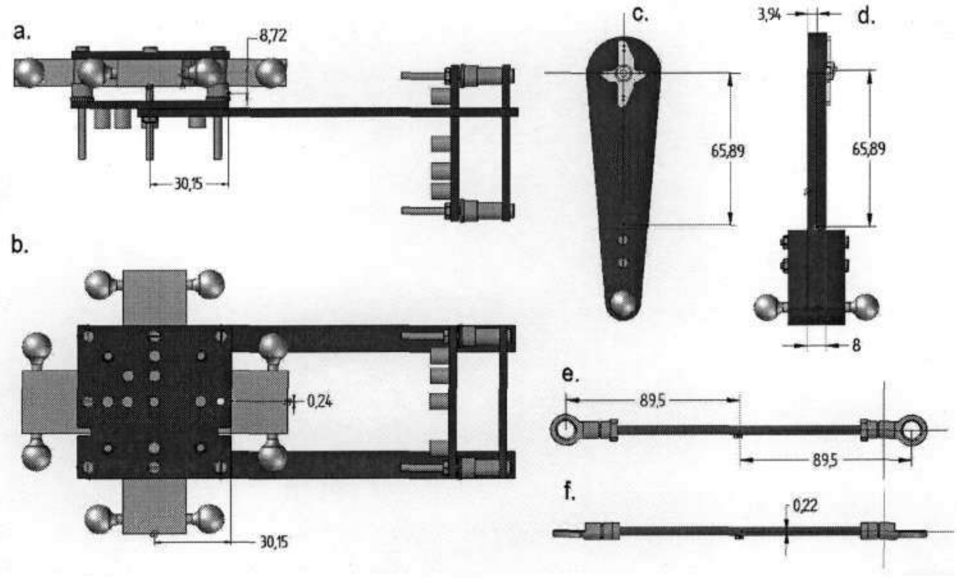


Figure 29 Centres of mass on major components of the moving system

The COM coordinates for the upper leg and lower leg were simplified in the dynamics model that was developed. For the upper leg it is the length from the axis of rotation to the point of the COM, which is 65.89 mm. For the lower leg it can be reduced to the mid point of the line from the "ankle" joint to the "knee" joint, which is 89.5 mm from either point.

The COMs are superimposed on the geometric model and is illustrated in Figure 30. These COMs can be calculated in the kinematics model, since the relative positions from joints do not change.

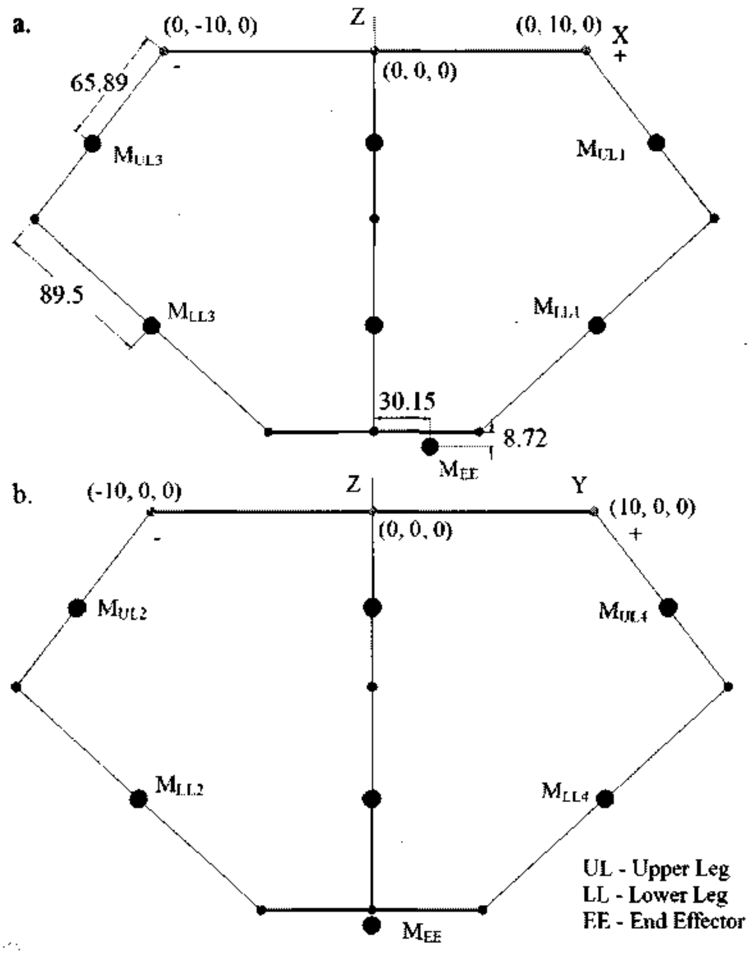


Figure 30 A depiction of the COMs of each link superimposed on the geometric model

The next step in approximating the system is to split the parallel structure into 4 serial parts. This is done at the end effector where the mass M_{EE} is carved up into 4 parts with a certain portion concentrated at each "ankle" joint. This is obtained from a look at the end effector when it is stationary. For static equilibrium the sum of the forces and torques must be zero. See Figure 31.

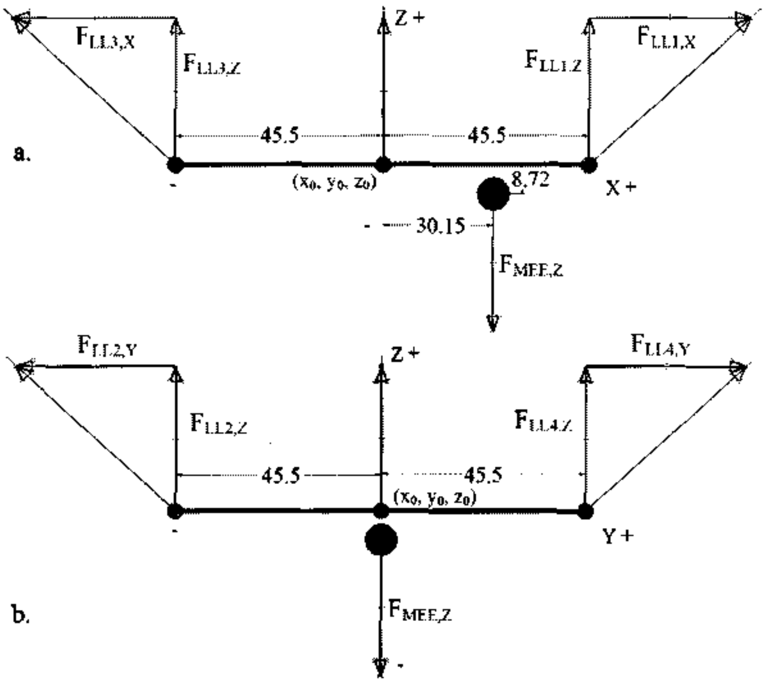


Figure 31 Illustration of torques about x_0 and y_0 on the plane $z = z_0$

To find the equivalent mass at each "ankle" joint, the following sets of equations must be solved, with torques positive in the clockwise direction:

$$\sum_{i=1}^{4} F_{LLi,Z} + F_{MEE,Z} = 0 \qquad \ldots \quad (4.16)$$

Torques about the Y-axis, the line x = x_0 and z = z_0.

$$45.5F_{LL3,Z} + 30.15F_{MEE,Z} - 45.5F_{LL1,Z} = 0 \qquad \ldots \quad (4.17)$$

Torques about the X-axis, the line y = y_0 and z = z_0.

$$45.5F_{LL2,Z} - 45.5F_{LL4,Z} = 0$$
$$\Leftrightarrow F_{LL2,Z} = F_{LL4,Z} \qquad \ldots \quad (4.18)$$

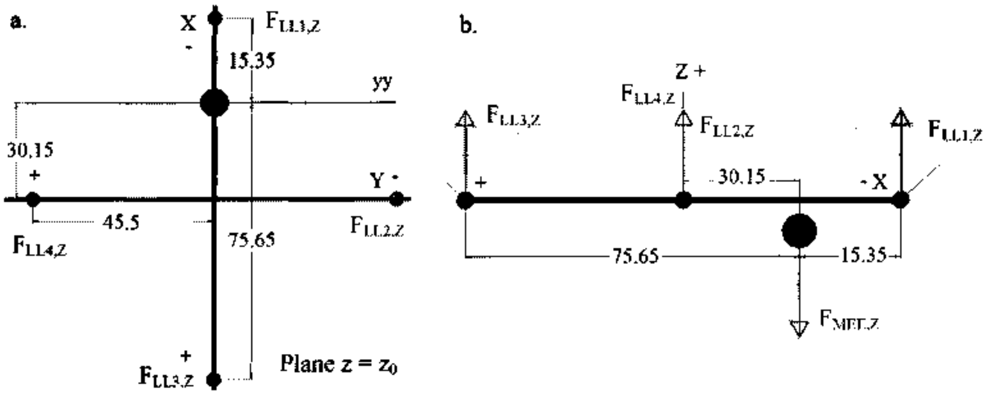


Figure 32 Illustration of torques about line yy ($x_0 = 30.15$, $z = z_0$)

The masses of the upper leg, lower leg and end effector are 0.055 kg, 0.0825 kg and 0.2413 kg respectively.

The torques about the line yy, given by $x_0 = 30.15$ and $z = z_0$:

$$F_{MEE,Z}(0)+15.35F_{LL1,Z} -2\times 30.15F_{LL2,Z} -75.65F_{LL3,Z} = 0 \qquad \ldots \quad (4.19)$$

(4.16), (4.17) and (4.19) may be written as follows:

$$F_{LL1,Z} + 2F_{LL2,Z} + F_{LL3,Z} = M_{EE}g \qquad \ldots \quad (4.20)$$

$$F_{LL1,Z} + 0F_{LL2,Z} - F_{LL3,Z} = \frac{30.15}{45.5}M_{EE}g = 0.6626M_{EE}g \qquad \ldots \quad (4.21)$$

$$F_{LL1,Z} - 3.9283F_{LL2,Z} - 4.9283F_{LL3,Z} = 0 \qquad \ldots \quad (4.22)$$

Or in matrix form:

$$\begin{bmatrix} 1 & 2 & 1 \\ 1 & 0 & -1 \\ 1 & -3.9283 & -4.9283 \end{bmatrix} \begin{bmatrix} F_{LL1,Z} \\ F_{LL2,Z} \\ F_{LL3,Z} \end{bmatrix} = \begin{bmatrix} M_{EE}g \\ 0.6626M_{EE}g \\ 0 \end{bmatrix} \qquad \ldots \quad (4.23)$$

= **A.F** = **C**

This system cannot be solved as **A** is not invertible, that is all equations are not linearly independent.

The reason for this is that the mechanical system is over determined. If one of the lower leg pairs is removed the mechanical system will still be functional, capable of performing its 3DOF. The additional leg was added to improve positioning accuracy, control and load carrying capability.

To model the worst case situation, set $F_{LL3,Z} = 0$. This yields $F_{LL1,Z} = 0.6626 M_{EE} g$ and $F_{LL2,Z} = 0.1687 M_{EE} g$. Hence the worst case partial mass of the end effector at the ankle joint is $0.6626 M_{EE}$ = 0.1599 kg.

Each 2 link arm is now treated serially. Standard techniques for dynamic modelling of serial manipulators may be applied, but a holistic picture of the machine must be maintained. Each position of this equivalent serial arm is obtained by solving the inverse kinematics of the parallel machine. To gauge the effect of this resulting serial arm on the motor, these 3 masses are combined to form one equivalent mass at the end of the upper leg, or "knee joint". The dynamics of the system may then be treated as that of a pendulum. This simplification is illustrated in Figure 33.

The effect of masses M_1 and M_2 on the upper leg (see Figure 33):

$$\begin{aligned} F &= F_{M1} + F_{M2} \\ F_{M1} &= M_1.g.\sin\theta_{2A} \\ F_{M2} &= M_2.g.\sin\theta_{2A} \end{aligned}$$

$$F_a = F.\cos(\theta_R - \theta_{2A} - 270°)$$

Hence the worst case equation for static torque of all masses on the motor is given by:

$$\begin{aligned} T \;&= F_{M3}.d_1 + F_a.d_2 \\ &= d_1.M_3.g.\sin(\theta_{RT} - 270°) + d_2.(M_1 + M_2).g.\sin\theta_{2A}.\cos(\theta_R - \theta_{2A} - 270°) \\ &= d_1.M_3.g.\cos(\theta_{RT}) - d_2.(M_1 + M_2).g.\sin\theta_{2A}.\sin(\theta_R - \theta_{2A}) \end{aligned}$$

Using standard trigonometry rules in particular the equality,

$\sin A.\sin B = \dfrac{\cos(A-B) - \cos(A+B)}{2}$, yields:

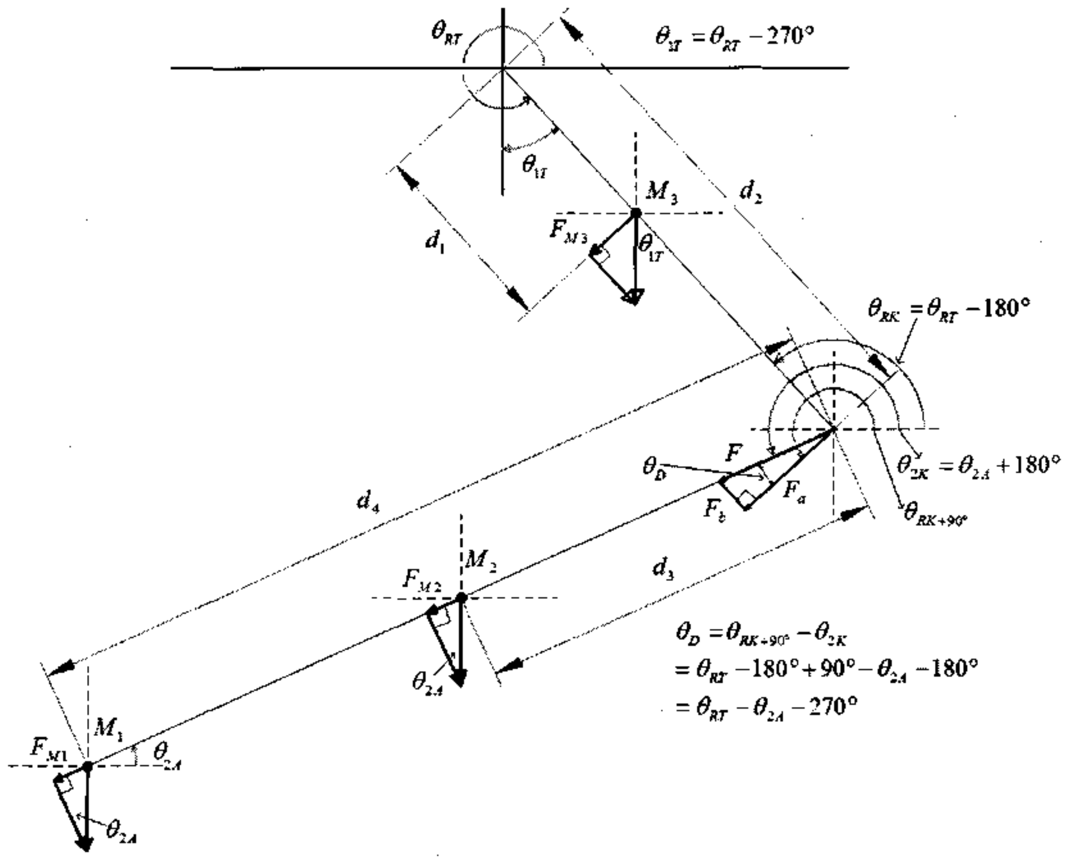

θ_{RT}
$\theta_{1T} = \theta_{RT} - 270°$
θ_{1T}
M_3
d_2
d_1
F_{M3}
θ_{1T}
$\theta_{RK} = \theta_{RT} - 180°$
θ_D
F
F_a
F_b
$\theta_{2K} = \theta_{2A} + 180°$
$\theta_{RK+90°}$
d_4
M_2
F_{M2}
d_3
$\theta_D = \theta_{RK+90°} - \theta_{2K}$
$= \theta_{RT} - 180° + 90° - \theta_{2A} - 180°$
$= \theta_{RT} - \theta_{2A} - 270°$
θ_{2A}
M_1
F_{M1}
θ_{2A}
θ_{2A}

Figure 33 Planar projection of serial leg equivalent for each leg of the PKM

$$\sin\theta_{2A}.\sin(\theta_{RT} - \theta_{2A}) = \frac{1}{2}\cos(2\theta_{2A} - \theta_{RT}) - \frac{1}{2}\cos(\theta_{RT})$$

Substitute this in the previous equation:

$$\begin{aligned} T &= d_1.M_3.g.\cos(\theta_{RT}) - d_2.(M_1 + M_2).g.\sin\theta_{2A}.\sin(\theta_R - \theta_{2A}) \\ &= d_1.M_3.g.\cos(\theta_{RT}) - \frac{1}{2}.d_2.(M_1 + M_2).g.\cos(2\theta_{2A} - \theta_{RT}) + \frac{1}{2}.d_2.(M_1 + M_2).g.\cos(\theta_{RT}) \end{aligned}$$

... (4.24)

Clearly, from Figure 33, the maximum static torque on the motor occurs when the upper leg is completely horizontal ($\theta_{RT} = 360°$) and the lower leg completely vertical ($\theta_{2A} = 90°$, this also forces the lower leg planer projection to equal its maximum possible length, that is the full length of the lower leg, i.e. 179 mm).

$$
\begin{aligned}
T &= d_1.M_3.g.\cos(360°)-\frac{1}{2}.d_2.(M_1+M_2).g.\cos(2(90°)-360°)+\frac{1}{2}.d_2.(M_1+M_2).g.\cos(360°)\\
&= d_1.M_3.g-\frac{1}{2}.d_2.(M_1+M_2).g.(-1)+\frac{1}{2}.d_2.(M_1+M_2).g\\
&= d_1.M_3.g+.d_2.(M_1+M_2).g\\
&= (0.06589)(0.055)(9.81)+(0.1)(0.2413+0.0825)(9.81)\\
&= 0.0356+0.3177 \quad = \quad 0.3533\ Nm
\end{aligned}
$$

The maximum mass at the "knee" joint is given by $\frac{T}{d_2.g}$.

$$M_{Max}=\frac{0.3533}{(0.1)(9.81)}=0.3601\ kg$$

The leg may now be modelled as a pendulum with a mass of 0.3601 kg at a distance of 0.1 m from the actuated rotational joint of the upper leg. This provides a conservative model for the PKM and eases calculation. The purpose of this model is to determine the maximum speed and acceleration at which the upper legs can be moved without the end effector overshooting its intended position; that is to keep the reactive torque applied by the linkage less than that of the maximum torque rating of the motor. This provides a modelling approach for the control of the robot from a theoretical point.

The model in the control system however uses control theory of plant estimation. This is described in chapter 6 Control Design.

4.4 Chapter Summary

Chapter 4 describes the mechanical design and the design process for PKMs. Fully illustrated CAD drawings of the mechanical structure is presented. This chapter also provides a combined geometric and algebraic method to solve both the forward and inverse kinematics as well as providing an illustration of the workspace envelope. The solutions to the FK and the IK are closed form, and these can be solved rapidly by the control software. Singularities of PKMs are mentioned, and those for the machine designed are indicated as well as how they are determined in the IK and avoided. Mass and dynamic modelling is also presented and the mechanical system reduced to 4 pendulums.

5 Electronic Hardware

5.1 Processor

The processor used for the electronic control system was the ATmega128 micro controller from Atmel. It is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications. Some of its features include [49]:

- 133 Powerful Instructions – Most Single Clock Cycle Execution
- 32 x 8 General Purpose Working Registers + Peripheral Control Registers
- Fully Static Operation
- Up to 16 MIPS Throughput at 16 MHz
- 128K Bytes of In-System Reprogrammable Flash
- 4K Bytes EEPROM
- 4K Bytes Internal SRAM
- Up to 64K Bytes Optional External Memory Space
- Two 8-bit Timer/Counters
- Two 16-bit Timer/Counters
- Real Time Counter with Separate Oscillator
- Two 8-bit PWM Channels
- PWM Channels with Programmable Resolution from 2 to 16 Bits
- 8-channel, 10-bit ADC
- Dual Programmable Serial USARTs
- Powerful multiplier supporting signed or unsigned multiplication and fractional format

The peripheral features that were used most extensively were the timers and the USART (universal synchronous asynchronous receiver transmitter).

The 16 bit timers were used to generate the PWM control signals for the servo motors.

A dedicated hardware USART in this microcontroller gives it the ability to communicate serially with any other processor possessing a USART. It is capable of both synchronous and asynchronous communication. The asynchronous feature was used as the microcontroller communicates with a PC via its RS232 serial port.

One ATmega128 was used. Its functions were:

- Communication with host PC
- Control of data converters
- Search and process acquired data
- Servo motor digital controller implementation
- Read ADCs
- Generate 4 PWM signals for servo control

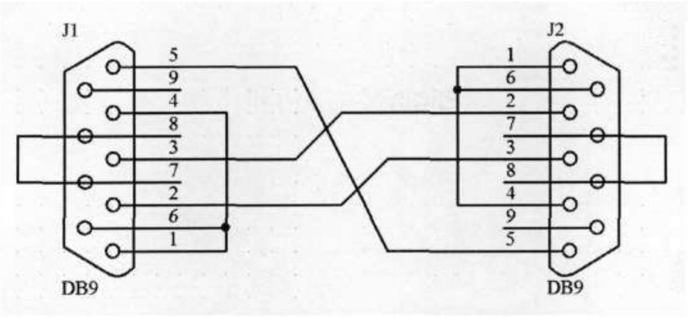
5.2 Communication

The controller is used to transfer the reference numbers of those sensors that are stimulated to the PC through its USART transceiver and the PCs RS-232 serial port.

One protocol for serial communication is the RS-232C standard, which stands for Recommend Standard number 232, C being the latest revision of the standard. The serial ports on most computers use a subset of the RS-232C standard. The full standard specifies a 25 pin "D" connector of which 22 pins are used. Most of these pins are not needed for normal PC communications, and most new PCs are equipped with male D type connectors having only 9 pins.

To use the RS232 port a null modem configuration of the communications line was implemented. Null modem cables cross the transmit & receive, DTR & DSR & CD and RTS & CTS lines in the cable. This configuration allows communication when there is no need for data flow control. Figure 34 shows the wiring diagram for the implementation.

Figure 34 Null modem with loop back handshaking



Connector 1	Connector 2	Function
2	3	RX ← TX
3	2	TX → RX
5	5	Signal Ground
1 & 4 & 6	----	DTR & CD & DSR
----	1 & 4 & 6	DTR & CD & DSR
7 & 8	----	RTS & CTS
----	7 & 8	RTS & CTS

5.2.1 Line Voltage Conversion

The voltage levels specified in the RS-232 standard for the serial port are -10 V for logic 1 and +10 for logic 0. These are different when compared to the microcontroller circuits that are powered by a 5 V source. In order for the controller and the PC to communicate a voltage conversion must occur. For this purpose a MAX 232 conversion chip is used. It has two internal charge pumps which convert the voltages as required. [50]

5.3 Servo Motors

A servo motor is one that can place its rotary shaft to specific angular positions depending on the reception and value of a particular coded signal. As long as that coded signal exists on the input line, the servo will maintain the shaft's angular position. Servos are used in radio controlled airplanes and helicopters to position elevators, rudders and blades. They are also used in radio controlled cars, puppets and robots. [51, 52]

5.3.1 The Inner-workings of a Servo

The servo motor is composed of a DC motor, a feedback potentiometer, control circuitry, a plastic casing and a gear box. The potentiometer allows the control circuitry to monitor the current angle of the servo motor. If the shaft is at the correct angle, then the motor shuts off. If the circuit finds that the angle is not correct, it will rotate the motor in the right direction until the desired angle is reached. The output shaft of the servo can move about 180°. Usually, there is a 210° range, but this varies by manufacturer and it is not capable of any further movement due to a mechanical stop built on the main output gear.

The amount of power applied to the motor is proportional to the angular distance it needs to travel. If the shaft needs to rotate a large angular distance, the motor will run at full speed. If it needs to turn only a small amount, the motor will run at a fraction of its full speed. The control wire is used to communicate the angle, which is determined by the duration of a pulse that is applied to the control wire, called Pulse Coded Modulation or Pulse Width Modulation.

The parameters for this pulse are its minimum/maximum values and its repetition rate. Given the rotation constraints of the servo, neutral is defined to be the position where the servo has exactly the same amount of potential rotation in the clockwise direction as it does in the counter clockwise direction. Different servos have different constraints on their rotation but they all have a neutral position, and that position is always around 1.5 ms (pulse width). Angular positioning is achieved through linear interpolation of pulse width, between the extreme positions of 0° and 180°. The resolution however is limited by the digital control system in use: 8 bit timers can achieve rotational resolution of 0.706° and 16 bit timers can achieve resolution of 0.003°. When a pulse is sent to a servo that is less than 1.5 ms the servo rotates to a position and holds its output shaft some number of degrees counter clockwise from the neutral point. When the pulse width is wider than 1.5 ms the opposite occurs. The minimum and maximum pulse width, that will command the servo to turn to a valid position are functions of each servo. Different brands, and even different servos of the same brand, will have different maximum and minimum pulse widths. Generally, for all servos, the minimum pulse is about 1 ms and the maximum pulse about 2 ms, and it has to be refreshed every 20 ms.

The maximum amount of force the servo can exert is its torque rating. Another parameter that varies from servo to servo is the turn rate. This is the time it takes for the servo to change from one position to another. [51, 52]

Servos Used

Four JR-591 servos were used in the design. They each have a mass of 350 grams and a torque rating of 5.1 kg.cm. Also its rated speed is 60°/0.21s, i.e. it rotates 60° in 0.21s. They were selected for reasons of cost, torque and size. The servo's torque was sufficient to carry and hold all masses attached to its shaft which was the main requirement. This motor loading was described in section 4.3.2 k. The maximum static torque applied by the system on the servo motor is 3.601 kg.cm (or 0.3601 kg at 10 cm, as was derived). This is well within the torque rating of the motor. The control design ensures that the dynamic reactive torque applied by all masses and links attached to the servos are less than its maximum rating. See chapter 6 on the control design.

5.4 Analogue To Digital Converters

To acquire direct feedback on the angular position of the servomotor (hence each upper leg), a modification was made to each servo, tapping directly into the analogue voltage on its potentiometer. This analogue voltage provides the servo with angular position feedback. The analogue signals were fed to 4 independent 8 bit TLC548CP analogue to digital converters (ADCs) [53]. The high input impedance of these ADCs do not affect the control circuitry of the servos. The reason for using 4 ADCs will be explained. Firstly the rotation of the servos were limited and calibrated, to make the mechanical system uniform as the servos are not identical and have slight variations. A mechanical calibration and limiting tool was made to prevent the upper leg from exceeding its maximum and minimum positions (see section 8.1.1 on calibration). The voltages at each of these maximum and minimum positions were different for each servo. The positive reference voltage on the ADC for each servo was tuned to the voltage appearing at the maximum position. The negative reference was tuned to the value appearing at the minimum position. Each ADC converts the rotation range (140°) exactly to a value between 0 and 255. Even though each servo may have a different voltage at for instance 120°, the digital value of each output of each ADC would read the same. This was the purpose of having separate ADCs with separate references.

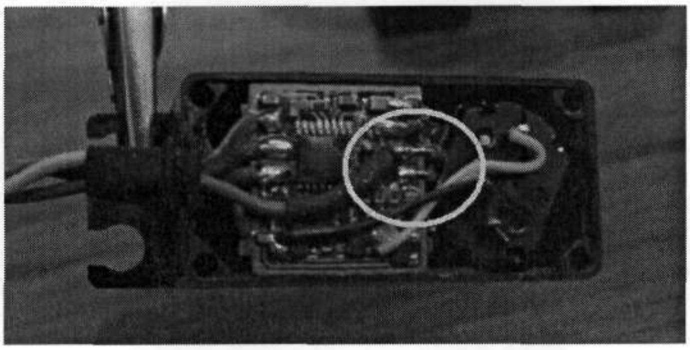


Figure 35 Wire tap into feedback potentiometer of servo motor

5.5 Laser Stimulant

Twenty-four 635 nm laser diodes were used, 12 for the vertical screen and 12 for the horizontal screen. The number of lasers used increases the sensor system resolution. See section 3.2.3 for an explanation on the arrangement of these lasers. The vertical laser set is 40 mm below z_0 (end effector z coordinate reference). This was allowed so that it would not affect the legs during motion. The 40 mm vertical offset can be accounted for in software or physically with the mounting of the vertical screen.

The lasers cannot be used continuously as they overheat and this leads to destruction of the diode in minutes. The nature of this design however does not require a continuous stream of optical power. The lasers are switched off just after the parallel to serial converters are loaded with the sensor data, through the data transfer stage until the next data acquisition. The data processing takes roughly 80 ms and the detector needs 10 us to switch on completely [54]. Utilizing a design factor of 100 yields a laser on time of 1 ms (100×10 us). This implies that the laser is on 1.25% of the time, which is sufficient to prevent burnout.

5.6 Light Detectors

5.6.1 Available Light Sensing Options

Light sensing applications vary widely from specialized scientific instrumentation that must detect individual light "particles" (photons) to systems that control high speed welding and cutting lasers which produce kilowatts of optical power. There are sensors for almost any application imaginable: from a photomultiplier tube which gives a large voltage pulse for every photon it detects, to cooled thermopiles that absorb kilowatts of power providing a thermocouple voltage proportional to the optical power absorbed. Other detectors include photodiodes, phototransistors, photodarlingtons, photoresistors, integrated circuits, and various hybrids. Table C1 of Appendix C summarizes these characteristics. For any application the following needs must be considered: [55, 56, 57, 58]

- Light source spectral characteristics,
- Optical power,
- Mating electronics,
- Packaging constraints,
- Image size,
- Signal-to-noise ratio,
- Frequency bandwidth,
- Operating lifetime,
- Reliability,
- Operation and storage environment,
- Performance,
- Cost.

The following sections describe the most popular light sensing technologies.

a. Photomultiplier Tubes

Photomultiplier tubes are special vacuum tubes that have a light sensing surface (the photocathode) that absorbs incoming light photons and emits secondary electrons. These secondary electrons are accelerated and multiplied within the photomultiplier tube by dynode plates. Each time an electron strikes a dynode, it has gained enough momentum to create a larger number of secondary electrons. This multiplication process continues for each dynode within the tube. Tubes with ten to twelve dynodes can easily generate multiplications of more than a million, resulting in sufficient current to develop hundreds of milli-volts across an output 50 ohm load resistor for a single incident photon.

Photomultiplier tubes provide ultimate detection sensitivity. They can sense the smallest amount of optical energy there is, i.e. an individual photon. When cooled, they can be essentially noise free, with at most one false photon pulse in a one second time period. However, there are many disadvantages to this light sensor, it [55, 56, 59, 60]:

- Is mechanically fragile
- Requires an extremely stable high voltage power supply
- Is expensive
- Has a package that is limited in shape and size
- Is susceptible to external magnetic fields
- Has limited wavelength sensitivity

b. Photodiodes

All diodes and transistors are light-sensitive. Photodiodes and phototransistors are designed specifically to take advantage of this fact. Photodiodes are manufactured in essentially the same way as semiconductor diodes used in conventional electronic circuits. The primary differences are that photodiode ICs are larger and they are packaged to allow light onto the sensitive area of the diode.

Photodiodes offer many conveniences and advantages that make them very practical for a wide range of applications:

- Can measure pico to milli-watts of optical power,
- Have standard packages which can be tooled to fit the application exactly,
- Almost any photosensitive shape can be fabricated with costs starting at R24 000,
- Wide range of wavelength sensitivity, from 190 to more than 2000 nm,
- Small and light,
- Highly reproducible sensitivity,
- Cheap,
- Very large detectable surface areas can be fabricated,
- Fast response time (as fast as 10 picoseconds),
- Can be conditioned to resist noise.

Photodiodes are used in applications ranging from sensors for door openings, assembly line controls, load levellers in luxury cars, to personal blood sugar meters for diabetics, sun-tan exposure meters, smoke detectors and x-ray baggage inspection systems. [55, 59, 60]

c. Phototransistors and Photodarlingtons

The most common phototransistor is an NPN bipolar transistor with an exposed base region. In this case light striking the base replaces a voltage that would have been applied there. It therefore amplifies variations in the amount of light striking it.

They are often more convenient than photodiodes because they have built in gain (amplification). Photodarlingtons have two stages of gain, with net gains that can be greater than 100,000. Phototransistors/photodarlingtons can therefore be coupled with a load resistor to accommodate TTL level voltages for a wide range of light levels. They have become popular due to their ease of use, low cost, TTL compatible signal levels, and suitability in applications that have nano-watts of available optical power. These devices however, do have some drawbacks when compared to photodiodes. The frequency bandwidth and linearity are relatively limited and spectral response is restricted to between 350 and 1100 nm. In addition, there are very large variations in sensitivity between individual devices and only a few standard package options. [55, 59, 60, 61]

d. Photoconductive Sensors

A photoconductive sensor is a thick film semiconductor material whose electrical resistance decreases with increasing incident light. These rugged assemblies can withstand hundreds of volts and are typically smaller than a 6 mm diameter.

Photoconductive sensors based on cadmium sulphide (CdS) have sensitivity curves that closely match the sensitivity of the human eye. They are useful in applications involving human light perception such as headlight dimmers and intensity adjustments on information displays. These sensors can be designed for measuring microwatts to milli-watts of optical power and are inexpensive at high volume. These characteristics make CdS photoconductors the sensor of choice in applications such as street light control and in the toy industry where economy is a major consideration.

Photoconductors made from materials other than CdS such as lead telluride and mercury cadmium telluride are also available. These materials have spectral sensitivities that cover the range that photodiodes cannot, i.e. from 2-15 μm. This longer wavelength sensitivity is very important for infrared imaging cameras and for long wave instrumentation such as is used to monitor carbon dioxide laser emission and atmospheric physics. These sensors tend to be more expensive than both silicon photodiodes and CdS photoconductors. [55, 56, 57, 58]

e. Integrated Circuits

Incorporating additional electronics directly onto a semiconductor sensor chip makes it possible to add additional functions to the sensor. An "optical IC" is an integrated circuit comprising photodiode and electronic-signal-processing-circuits. Additional functions such as current to voltage conversion and reference level sensing (e.g. a Schmitt trigger) can be incorporated. Other optical ICs can provide signals highly immune to noise, such as a current-to-frequency conversion.

The principal advantages of an optical IC are ease of use, small size and immunity to electronic noise when compared to a photodiode with separate electronics. These devices are much more expensive and offer a very limited active light sensing area. Custom tooling for specific applications is also expensive. [55, 56, 57, 58]

f. Hybrids

The electronic functions of an optical IC can also be provided by a hybrid circuit that has unpackaged IC components (die) attached to a substrate that also contains a photodiode. This type of sensor combines the ease of use and immunity to electrical noise of an optical IC with increased design flexibility and lower tooling costs. In addition, the sensitivity can easily be increased with a larger photodiode active area without the added cost of a separate detector.

The primary disadvantages of a hybrid sensor are its cost and reliability. Cost can be several times higher than the electronic assembly option discussed below and reliability testing is difficult to quantize, so either limited reliability screening is implemented, or the piece cost becomes high. [55, 56, 57, 58]

g. Sensor Electronic Assemblies

Combining any of the sensors listed above with printed circuit based electronic signal processing creates sensor assemblies or "black boxes". The user defines specifications for light input and the desired output response, the vendor then builds and tests the systems to ensure that the specifications are met. An assembly can also include optical components such as lenses and special wavelength filters. The user bolts the assembly into place and connects it to the high-level electronics; there are no concerns about mismatch between the purchased sensor and front-end amplifiers or diagnostic electronics. The system is relatively immune to noise and is highly reliable due to the mature manufacturing technologies used.

Sensor electronic assemblies are easy to implement. Experienced vendors can often deliver better reliability and lower cost products compared to in-house manufacturing. The main disadvantage is less flexibility in making changes dynamically, but this is not an issue for a responsive vendor or mature designs. [55]

h. Other Sensors

There are many other types of sensors. These include avalanche photodiodes, bolo-meters, self-scanned arrays an photon drag detectors. A sensor vendor can provide information about these devices and can discuss the physics and advantages of each detector technology.

i. Selecting a Sensor

Reviewing a few design aspects provides sufficient information for making an optimal choice of detector for a given application.

Wavelength – An effective choice for detector can be made based on the range of wavelengths of interest. These can be seen in table C1 of Appendix C. For example detecting wavelengths below 1100 nm, photoconductive cells or a silicon-based detector would be appropriate. At wavelengths above 1100 nm, the costs and technology options are not straightforward, and a detector vendor consultant should provide the most effective guidance.

Optical Power – A detector must be capable of providing an output at the given optical power. Applications detecting wavelengths with at least microwatts of optical power in the visible spectrum, in which the sensor is simply required to detect if light is present, can use one of the least expensive and most rugged detectors available, the photoconductive cell.

Silicon phototransistors and photodarlingtons should be considered for applications that are required to detect nano-watts of optical power within a 5 mm diameter spot at wavelengths between 350 and 1100 nm.

Performance – For UV to near IR wavelengths, photodiodes offer the best overall performance. They are only slightly more expensive than phototransistors, but their spectral range is broader and they have lower noise, more uniform sensitivity and reproducibility, a larger dynamic range, better linearity and more packaging options. Also, photodiodes can routinely detect pico-watts of optical power. If phototransistors or photoconductive cells are not appropriate for an application, more often than not a photodiode will afford the best alternative.

[55, 56, 57, 58]

At least 90% of detector applications should be satisfied by using phototransistors, photodarlingtons, photodiodes or photoconductive cells. When light levels are extremely low, ambient electronic noise levels high, or there are limited space requirements, alternatives such as optical ICs, hybrids or photomultipliers should be investigated.

j. Sensor Screen Detectors

The OP521 surface mount phototransistor was chosen as the detector. It is sensitive to light wavelengths in the range 400 – 1100 nm, with the best spectral response at 900 nm [54]. There are 512 sensors spread over two detector screen PCBs (256 sensors each) covering 225 cm^2 each. There is a 10 mm resolution (spacing) between sensors on both the vertical columns and horizontal rows. The detector screen provides coordinates for check points in space. These are used to correct position errors in the robots workspace. Errors are no longer accumulated from one extremity to the next but are limited to the resolution of the combined laser and sensor arrangement.

5.7 Buffers/Amplifiers

The need for buffers or amplifiers for sensor signal conditioning depends on the strength of the incident sensor stimulant. As lasers were used, a large percentage of the output optical power falls on the detector screen sensors, due to the coherent nature of laser light. Furthermore the flat and tiny 1206 surface mount package of the OP521 phototransistor allows more incident light to fall on it than previous versions of the detector screens, made by the author, using phototransistors (LPT3133) in LED type packages. The 1206 package diagonal length is smaller than the beam diameter of the laser. Hence there was no need for signal conditioning via buffers or amplifiers.

5.8 Serialization

The outputs from each sensor are fed to the parallel inputs of a parallel to serial data converter, the 74LS166 [62]. This was to serialize the data for transfer to a PC. There are 4 control lines (CLEAR, SHIFT/LOAD, CLOCK and CLOCK INHIBIT) and 10 data lines (1 serial input, 8 parallel inputs and 1 serial output) per data converter. There are 32 parallel-to-serial-data-converters per detector assembly. The serial output line from each 74LS166 is fed into the serial input line of the following 74LS166. The resulting configuration provides one data output line for each detector assembly. The controller searches through the 64 bytes of data for bits representing stimulated sensors. It then transfers the number of each sensor.

5.9 Power Supply

An ATX computer power supply was used to power all the electronics and servo motors. It had to be modified, however, to work independently from a PC. The steps for converting these power supplies can be viewed on many electronic hobbyist websites.

"How to convert a Computer ATX Power Supply to a Lab Power Supply" - http://www.wikihow.com/Convert-a-Computer-ATX-Power-Supply-to-a-Lab-Power-Supply

5.10 Schematics and PCBs

Two detector screen sensor electronic assemblies were made, one for each of the vertical and horizontal planes. Each of these electronic assemblies consists of a sensor screen and a data/control-signal routing board.

The sensor electronic assemblies were made to be modular, so that a screen of any practical size could be built. This is the single biggest advantage of this system. Each sensor module plugs into each data/control-signal module via rail headers. The modules then plug into each other (via wired link or a separate board with rail headers) and are arranged to form one large detector screen. The modules were made such that the ICs and phototransistor components were on either side of the modules outer layers after the boards were put together, so that components could be changed easily if they malfunctioned. Each sensor module consists of 16 OP521 phototransistors. Each data/control-signal module consists of two 74LS166 ICs. See Figures 36 – 39.

Each sensor assembly screen consists of 256 OP521 phototransistors and of 32 74LS166 parallel to serial data converters. There are 16 sensor modules per sensor screen assembly. These 2 sensor electronic assemblies (1 vertical and 1 horizontal) then connect to a main embedded controller. This is shown in Figure 40.

The main components of the controller board were the microcontroller, MAX232 level shifter, BC547 transistors used to power the lasers and the TLC548CP analogue to digital converters. See Figures 41-43.

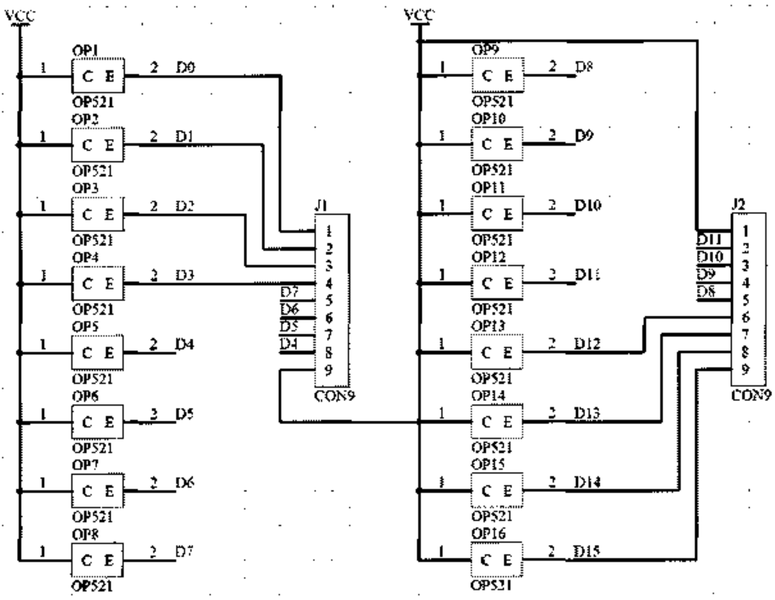


Figure 36 Schematic of sensor screen module

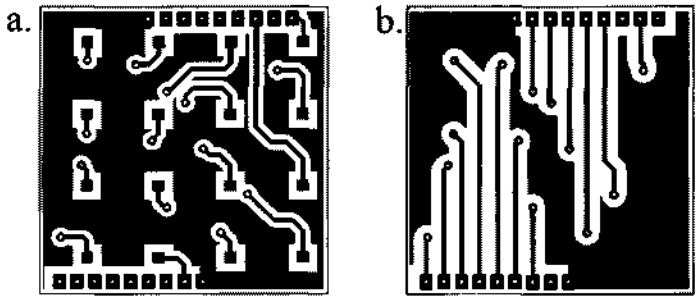


Figure 37 PCB of sensor screen module

(Not to scale)

a. Top layer holding components
b. Bottom layer

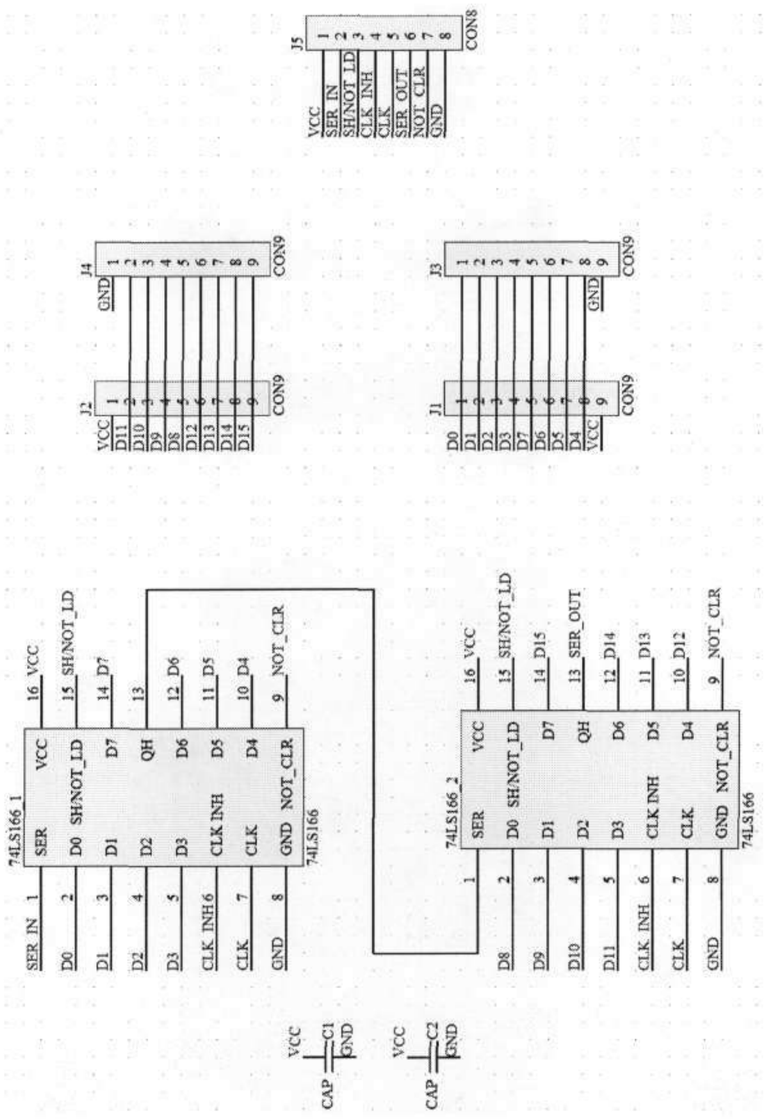


Figure 38 **Schematic of data/control-signal module**

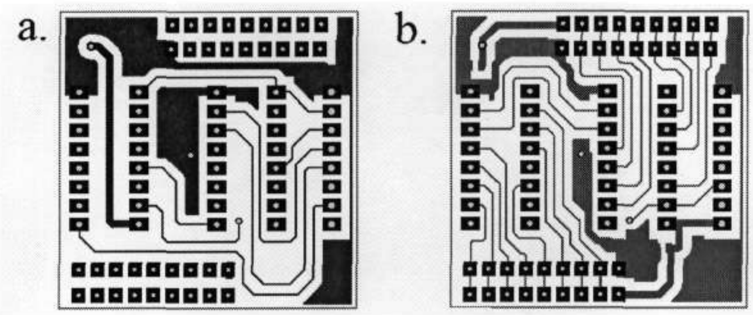


Figure 39 **PCB of data/control-signal module**

(Not to scale)

a. Top layer

b. Bottom layer which holds components

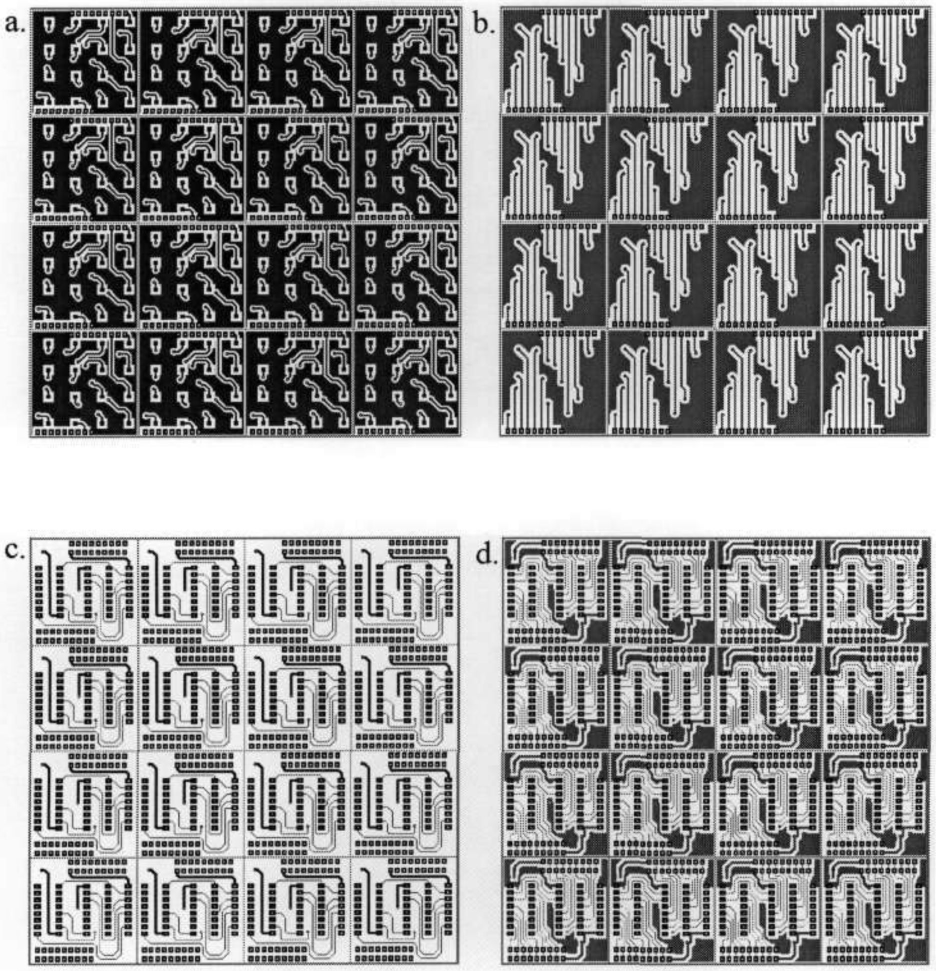


Figure 40 Sensor electronic assembly

a. Detector screen, top layer – holds components
b. Detector screen, bottom layer
c. Data/control-signal board, top layer
d. Data/control-signal board, bottom layer – holds ICs

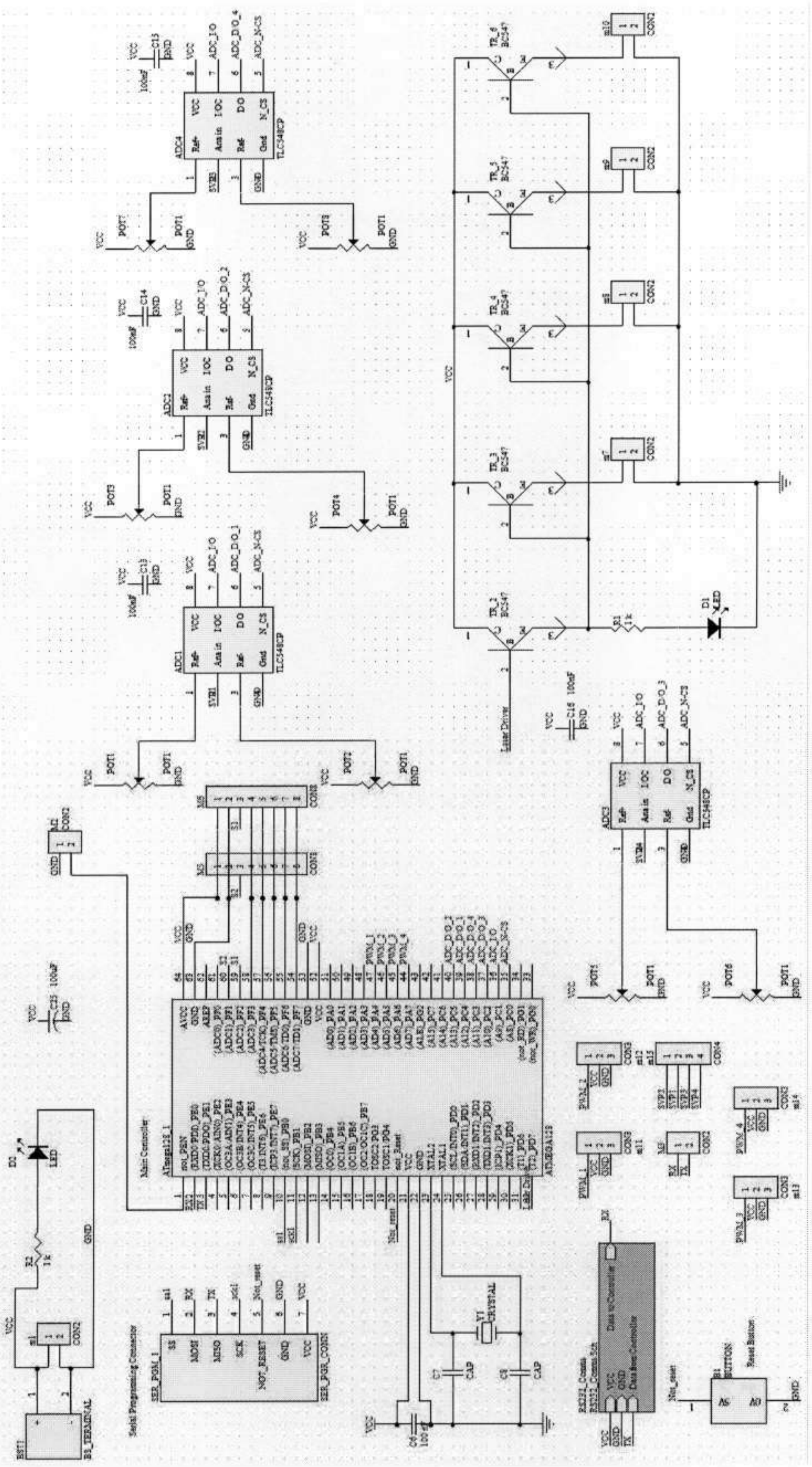


Figure 41 Schematic of embedded controller

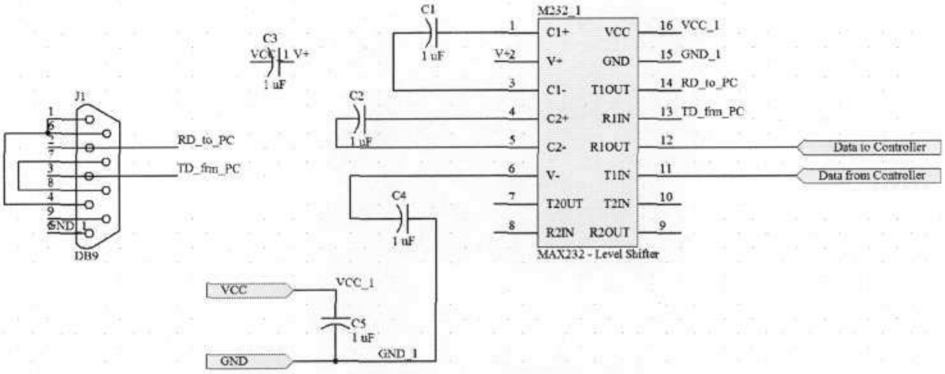


Figure 42 Schematic of RS232 level shifter

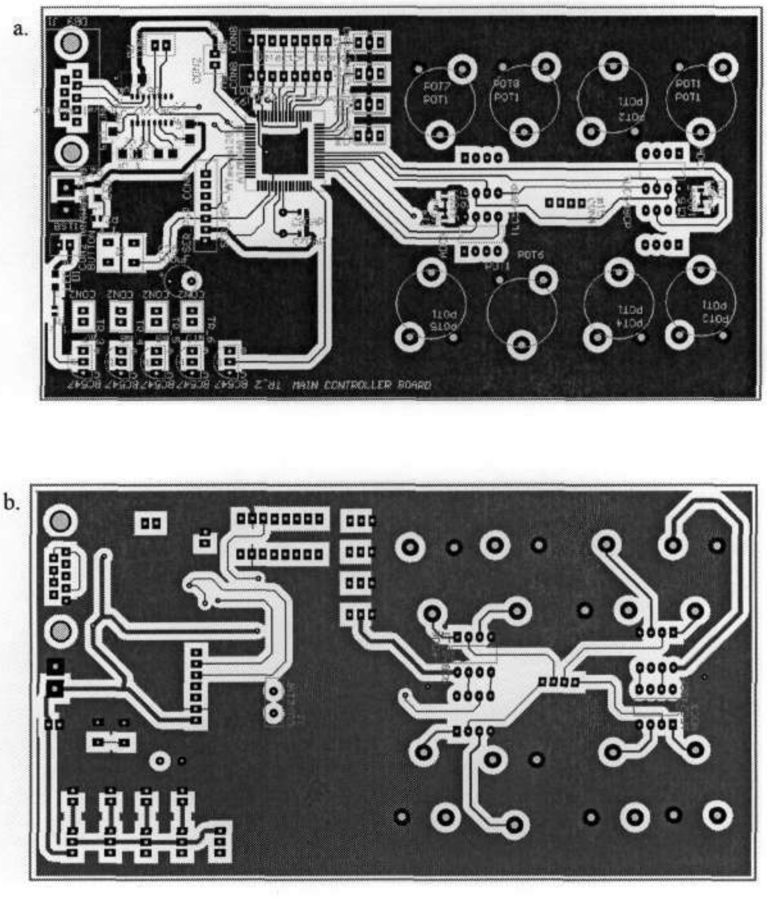


Figure 43 Embedded controller PCB

(Not to scale)

a. Top layer b. Bottom layer

5.11 Chapter Summary

The electronic design is presented. The components and their implementations in this design are described. It begins with a motivation for the choice of embedded microcontroller, the ATmega128. One processor is used in the design to acquire sensor data, process that data and for servo motor control. Communication with the PC control software occurs via the controller's USART and the PC's RS232 serial port. The Max232 facilitates voltage conversion. The servo motors used are then discussed as well as the modification to read the servo position, through 4 TLC548CP ADCs. A discussion of the detector screen components, i.e. the lasers and light sensors, then follows. In total 24 lasers were used, 12 per detector screen, and the OP521 phototransistor was used as the light sensor. There are 256 phototransistors per screen. They connect to 32 74LS166 parallel to serial data converters. The output serial line from each converter feeds into the serial input line of the following converter, and this allows 2 data lines to carry all the data to the microcontroller. The design schematics and PCB diagrams are then illustrated.

6 Control Design

The control system design consists of 2 levels of control. At the macro stage, it consists of tracking and control of the coordinates in 3D space in software. This is done by solving the inverse kinematics for each position of the robots trajectory and obtaining angular values for rotation of the upper legs. The trajectory of the robot is the path in 3D space which the end effector reference coordinates must follow. This macro stage has 2 sources of position feedback i.e. from the motor encoder relating angular position of the leg, and the direct end effector sensor.

The micro stage consists of an embedded system controlling the position of each leg and the manner in which it drives the legs to the desired angular position at maximal speed with a control algorithm that inherently takes into account the system dynamics. There is only one source of feedback which is the motor position potentiometer.

6.1 Macro Stage Control

The macro level control system problem for this PKM can be stated: from the current position, which can be measured accurately via the combined sensor readings of the direct end effector sensor system and the motor position potentiometers, follow a trajectory to the point of interest in the workspace of the manipulator. The block diagram for this is shown in Figure 44.

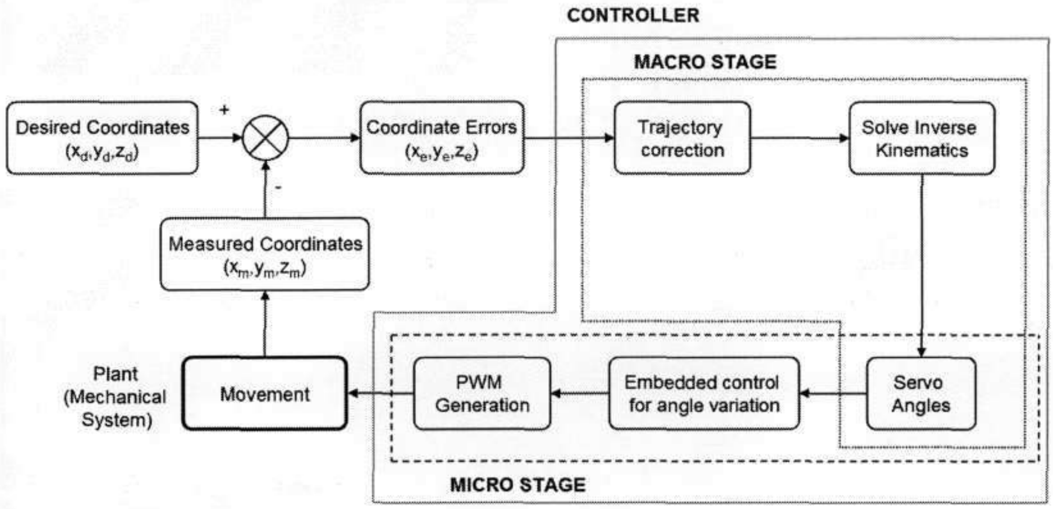


Figure 44 Block diagram for PC software controller

6.2 Micro Stage Control

6.2.1 Characterising the system

The first step in designing the control system for the parallel robot scale model was to obtain a transfer function for the compound leg. Each leg (comprising servo motor, upper leg, lower leg and part of end effector) is treated as a plant. Inherently stored in this model are the dynamics of the system.

Method

To obtain a transfer function in the S-Domain, first apply a known input to the system ($x(t)$) and measure the output. Then find a best fit curve to the output data obtained ($y(t)$), that is find a mathematical function for the output (if not exact then best approximate) in the time domain. Next take the Laplace transform of the output ($Y(s)$) and divide it by the Laplace transform of the input ($X(s)$). To derive the transfer function of the "plant" in the time domain ($g(t)$), take the inverse Laplace transform of$G(s)$. This method provides a means of plant estimation for linear time invariant (LTI) systems. To determine if a system is LTI apply 2 inputs and measure their outputs. Then apply an input to the system which is the sum of those 2 previous inputs, provided that this value is within the operating range of the system, and measure the output. If this output is the sum of the previous 2 outputs then the system is LTI. This is the principle of superposition, and it applies to linear systems. [63]

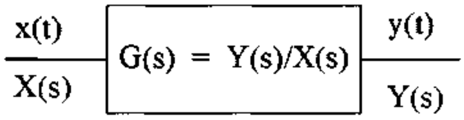


Figure 45 Typical plant model

Usually $x(t)$ is a simple standard function whose Laplace transform is known. The most common types are an impulse, step or ramp function. The corresponding output is called the impulse, step or ramp response. Of the three functions named the easiest to generate is the step input, and it was used to obtain the step response. It is generated by changing the input from some initial value to some final value. The function generated is not ideal as there will be some delay involved in reaching the final value, however for an acceptable approximation these non ideal characteristics do not have to be modelled if the system time constants (generally reactive time for mechanical components or chemical processes) is far larger. A number of different sized steps were applied to the system, and their outputs measured. The test to determine if the system was linear was carried out, and it was found that the system satisfied the condition for linearity. This can be seen readily if one considers the steady state (it was confirmed for the transient state) of the system, where each angle has a linear mapping to a particular voltage. [63]

The mathematical model of the system is obtained at the maximum speed at which it can operate. The embedded software controlling the mechanical system at this point changes the input to the servos immediately when it receives control signals from the host PC control software. That is, no digital controller is implemented to condition the "plant" input to account for system dynamics. For the system in question, the input is a reference angle, which is changed from an initial angle to a final angle. These angles were chosen so that the effect of one upper leg (or motor) on another was minimal and moving vertically along the z-axis ensures this. Furthermore pure vertical movements mean that the legs operate uniformly, that is the angular change is the same for each leg. The step change in input therefore moves the reference point on the end effector from $(0,0,z_a)$ to $(0,0,z_b)$.

Positional feedback is measured via the servo potentiometer. The Cleverscope CS328 PC oscilloscope was used to acquire the step response. This is shown in Figure 46. The noise seen on the output is caused by an oscillation of the potentiometer wiper on its windings. The modelling is accomplished by averaging the maximum and minimum values of the errors around time t_i, for all i in the data set.

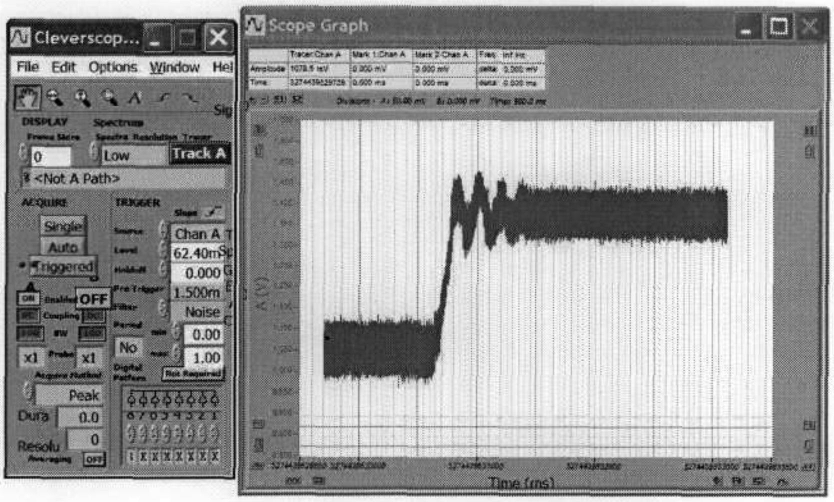


Figure 46 Step response via display on Cleverscope Software

The data was then exported as text, and imported to MATLAB, where it was normalized, in this context meaning that the output was shifted and scaled to start at 0 (with the time of the step change in input occurring at 0 s) and end at 1. These operations on the data did not affect the design of the control system. A control system designed for normalized data will work with the actual system. This was confirmed in section 8.2.5 Figure 70 f.

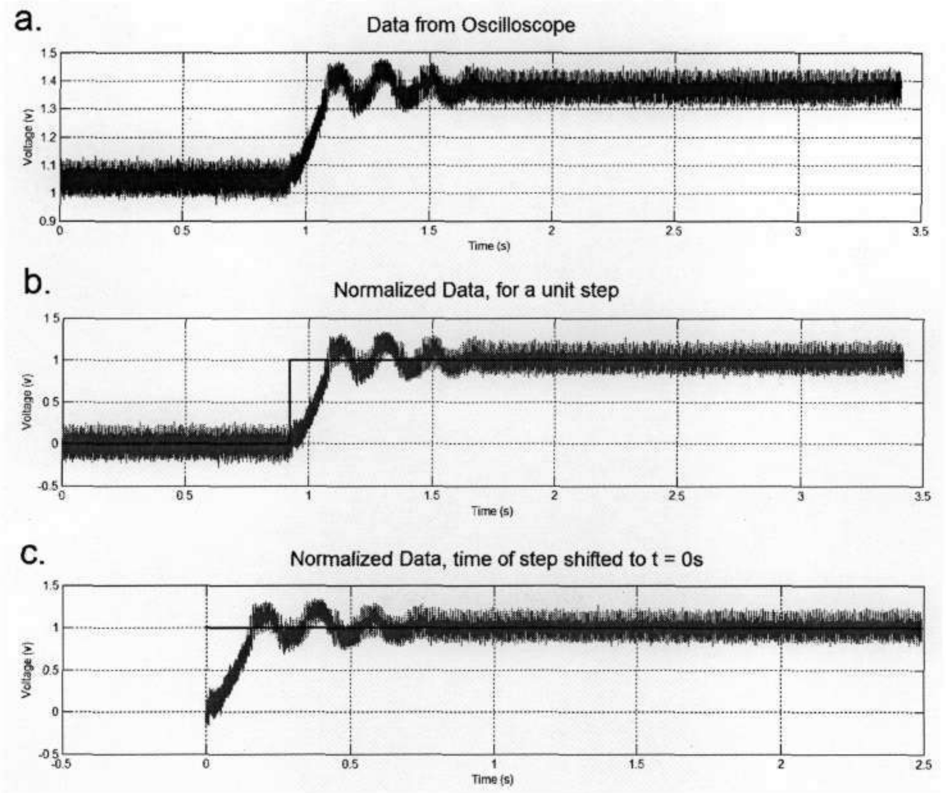


Figure 47 Display of data in MATLAB

a. Data from oscilloscope

b. Normalized Data

c. Normalized and step time shifted to t = 0s.

Once this was done an approximate mathematical function was fitted to the data (as it was not possible to find an exact function to fit the data). The output data resembles the response of a second order system. The time function of a general second order step response is given by: $y(t) = 1 - e^{-at}.\cos(\omega t)$. Suitable values for a and ω had to be found.

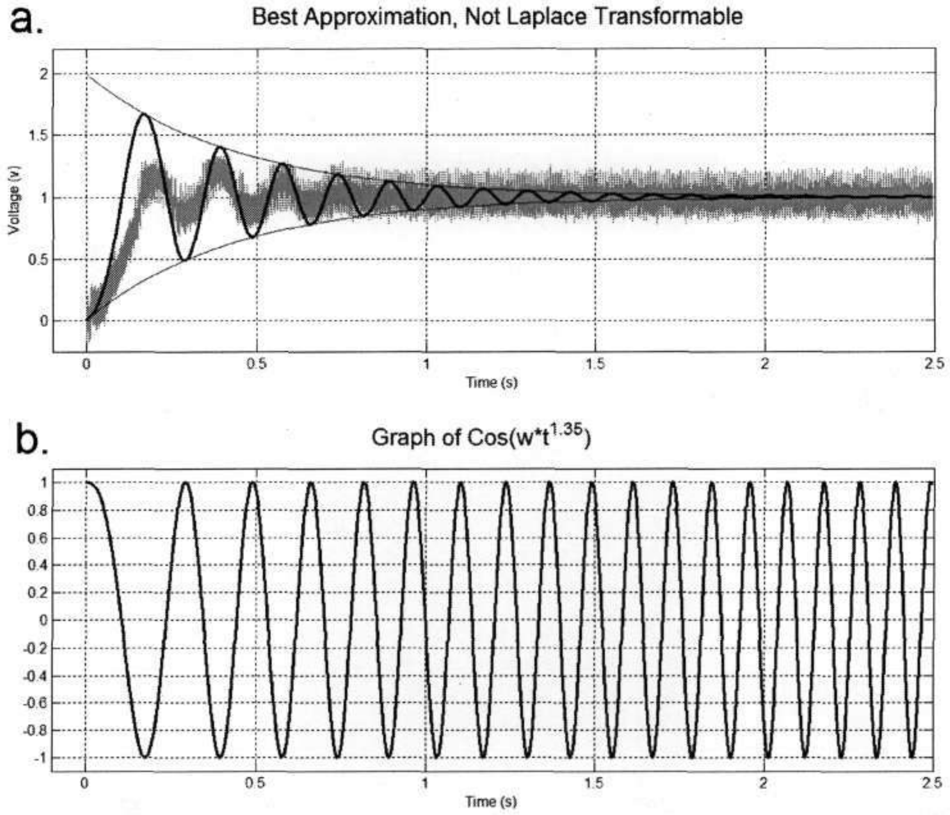


Figure 48 Superposition of the best single function approximation to the data

a. Best approximate single mathematical function to the data, not Laplace transformable

b. Graph of $\cos(\omega.t^{1.35})$

The best fit curve for the data was given by the function:

$$y(t) = 1 - e^{-at}.\cos\left(\omega t^{b}\right) = 1 - e^{-2.3t}.\cos(33t^{1.35}) \qquad \ldots \qquad (6.1)$$

This function however is not Laplace transformable. A graphical illustration is shown in Figure 48 a.

The best fit Laplace transformable function was found to be:

$$y(t) = 1 - e^{-at}.\cos\left(\omega t\right) = 1 - e^{-2.3t}.\cos(27t) \qquad \ldots \qquad (6.2)$$

This is shown in Figure 49 a. The 3rd oscillation of the approximation was made to fit the 3rd oscillation of the data curve, yielding values of 2.3 and 27 for a and ω respectively. The first 2 peaks and troughs do not coincide with those of the data curve. They occur ahead of time and have higher amplitude displacements. This means that the modelled system given by the approximation is more reactive. A control system designed for this slightly more reactive system would work with the real world system as this can be likened to designing for a worst case situation. A comparison of the best fit approximation and the acceptable Laplace transformable approximation is shown in Figure 49 b.

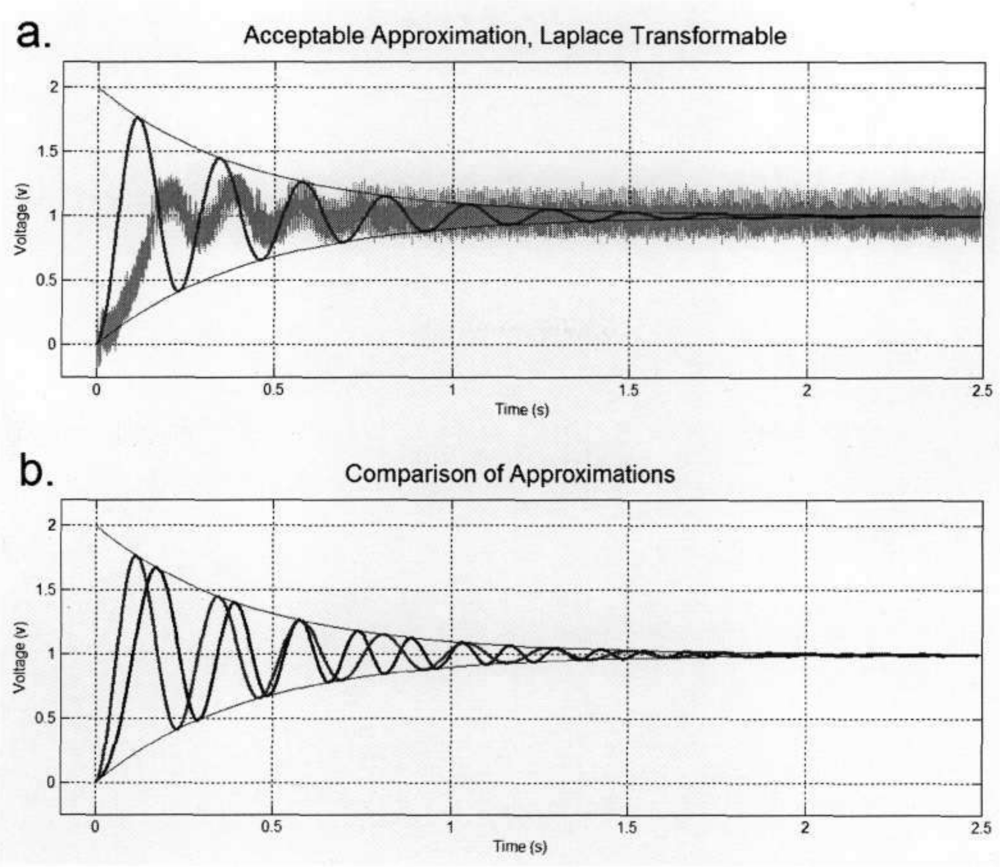


Figure 49 Plant approximations

a. Acceptable Laplace transformable approximation

b. Comparison of best approximation and the Laplace transformable approximation

Plant Transfer Function

Take the Laplace transform of the output equation $y(t)$, equation 6.2. [63]

$$\begin{aligned}
Y(S) &= L\left[1-e^{-at}.\cos(\omega t)\right] &&= \frac{1}{s}-\frac{s+a}{s^2+2as+\left(a^2+\omega^2\right)} \\
&= \frac{s^2+2as+\left(a^2+\omega^2\right)-s(s+a)}{s^2+2as+\left(a^2+\omega^2\right)}.\frac{1}{s} &&= \frac{s^2+2as+\left(a^2+\omega^2\right)-s^2-as}{s^2+2as+\left(a^2+\omega^2\right)}.\frac{1}{s} \\
&= \frac{as+\left(a^2+\omega^2\right)}{s^2+2as+\left(a^2+\omega^2\right)}.\frac{1}{s} &&= \frac{2.3s+\left(2.3^2+27^2\right)}{s^2+2\times 2.3s+\left(2.3^2+27^2\right)}.\frac{1}{s} \\
&= \frac{2.3s+734.29}{s^2+4.6s+734.29}.\frac{1}{s} && \ldots \quad (6.3)
\end{aligned}$$

The input equation, the step function $1(t)$ has the Laplace transform $\frac{1}{s}$, or $X(s) = \frac{1}{s}$. Hence the transfer function $G(s)$ is given by:

$$\begin{aligned}
G(s) &= \frac{Y(s)}{X(s)} = \frac{\frac{2.3s+734.29}{s^2+4.6s+734.29}.\frac{1}{s}}{\frac{1}{s}} = \frac{2.3s+734.29}{s^2+4.6s+734.29} \\
&= \frac{as+a^2+\omega^2}{s^2+2as+a^2+\omega^2} \qquad a=2.3 \;\&\; \omega=27 \qquad \ldots \qquad (6.4)
\end{aligned}$$

6.2.2 Control System in S-Domain

For positional control in a robotic system, the second order response with overshoot and oscillations is not acceptable as this means that the end effector overshoots its intended position and has damped oscillations until it settles to that position. A first order response is required, and a controller $GC(s)$ is used to reshape the output of $G(s)$. A typical feedback control system is described by Figure 50.

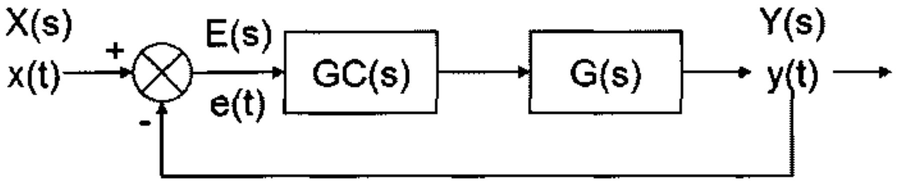


Figure 50 **A typical feedback control system with controller $GC(s)$ and no sensor conditioning**

The transfer function of this system is given by:

$$GFB(s) = \frac{Y(s)}{X(s)} = \frac{GC(s).G(s)}{1+GC(s).G(s)} \qquad \ldots \qquad (6.5)$$

A time function for a first order step response is given by $1-e^{-bt}$. Taking the Laplace transform yields:

$$Y(s) = L\left[1-e^{-bt}\right] = \frac{1}{s}-\frac{1}{s+b} = \frac{b}{s+b}.\frac{1}{s} \qquad \ldots \qquad (6.6)$$

Dividing by the Laplace transform of the input step, gives $GFB(s)$:

$$GFB(s) = \frac{Y(s)}{X(s)} = \frac{\frac{b}{s+b}.\frac{1}{s}}{\frac{1}{s}} = \frac{b}{s+b} \qquad \ldots \qquad (6.7)$$

The parameter b is used to shape the response, the smaller the value of b the slower the response. To obtain $GC(s)$, equation 6.7 must be written in the form of equation 6.5.

$$GFB(s) = \frac{b}{s+b} = \frac{\frac{b}{s}}{1+\frac{b}{s}} \qquad \ldots \qquad (6.8)$$

This implies:

$$GC(s).G(s) = \frac{b}{s} = GC(s).\frac{2.3s+734.29}{s^2+4.6s+734.29}$$

$$\Leftrightarrow GC(s) = \frac{s^2+4.6s+734.29}{2.3s+734.29}.\frac{b}{s} \qquad \ldots \qquad (6.9)$$

Using the STEP function in MATLAB, a value for b was selected to gain an acceptable output response. Using the 95% criterion, b or t can be designed when given the other.

$$\begin{aligned} 1-e^{-bt} &= 0.95 \\ e^{-bt} &= 0.05 \\ -bt &= \ln(0.05) \\ \Rightarrow b &= \frac{\ln(0.05)}{-t} \qquad \ldots \qquad (6.10) \end{aligned}$$

For $b = 4$, it takes 0.752 s to reach 95% of its final value.

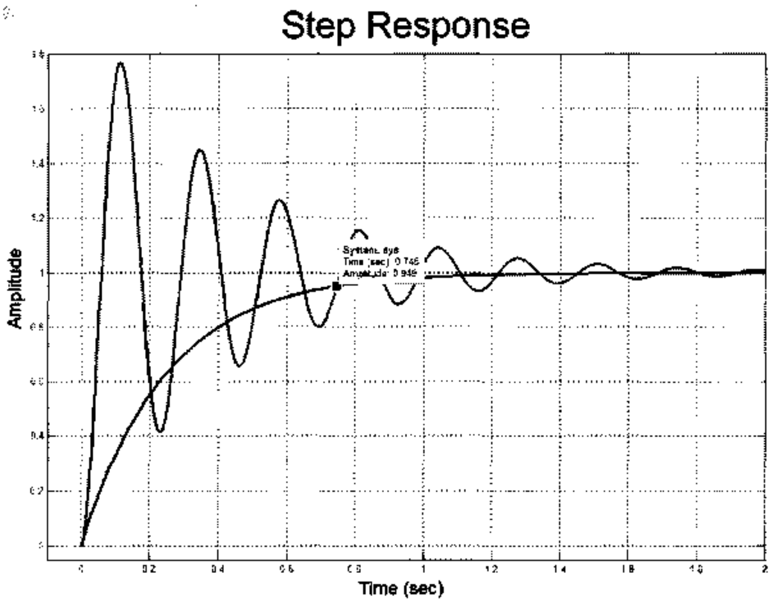


Figure 51 Comparison of modified and original step response of model

6.2.3 Control System in the Z-Domain (Discrete time)

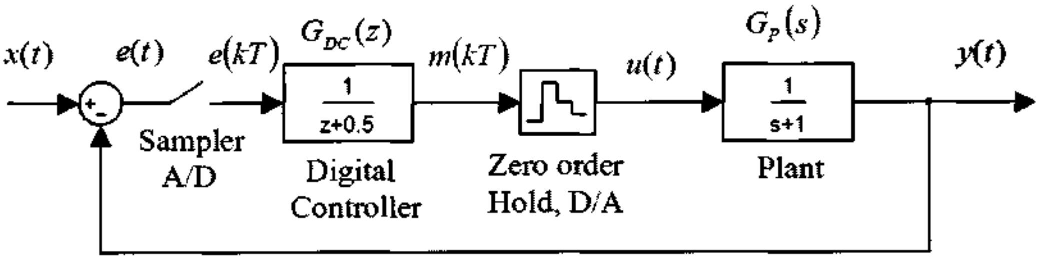


Figure 52 Block diagram of a typical discrete time feedback control system

The block diagram of Figure 52 represents a general form of a digital control system. The error signal is sampled and digitized before being passed to the digital controller. The digital controller modifies this error and feeds it to a hold circuit where the digital output, which is a quantized discrete time signal, is turned into a quantized continuous time signal. This control signal is then fed to the plant. [64, 65]

The PKM control system can be modelled similarly, even though the input signal is actually a digital signal and the output is digitized before the summing joint, shown in Figure 53.

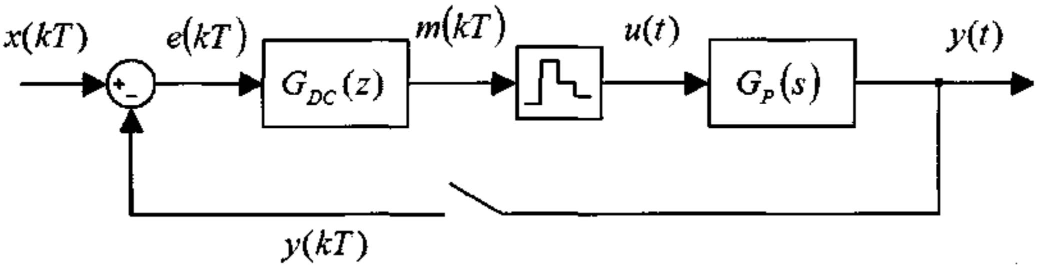


Figure 53 Block diagram of PKM servo motor controller

This block diagram may be modified as in Figure 54.

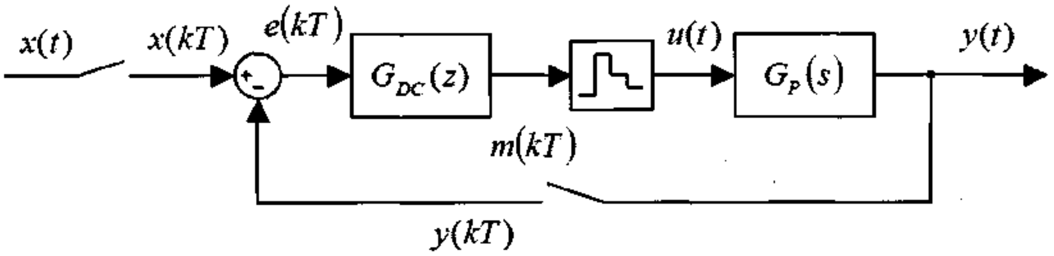


Figure 54 **Modified block diagram of PKM servo motor controller**

This then reduces to the block diagram of Figure 52.

The digital controller is not the Z transform of the controller $GC(s)$ in the S Domain. It has to be derived separately and this will be shown in stages.

To get the feedback transfer function divide the Z transform of the output function by the Z transform of the input function. The output function $y(kT)$ { $y(t)$ at discrete instants of time kT } is given by $1-e^{-bkT}$, the input function is the unit step $1(kT)$. [64, 65]

$$Z[y(kT)] = Z\left[1-e^{-bkT}\right] = \frac{\left(1-e^{-bT}\right)z^{-1}}{\left(1-z^{-1}\right)\left(1-e^{-bT}z^{-1}\right)} = Y(z)$$

$$Z[x(kT)] = Z[1(kT)] = \frac{1}{1-z^{-1}} = X(z)$$

$$\text{Therefore } GFB(z) = \frac{Y(z)}{X(z)} = \frac{\dfrac{\left(1-e^{-bT}\right)z^{-1}}{\left(1-z^{-1}\right)\left(1-e^{-bT}z^{-1}\right)}}{\dfrac{1}{1-z^{-1}}} = \frac{\left(1-e^{-bT}\right)z^{-1}}{1-e^{-bT}z^{-1}} \quad \ldots \quad (6.11)$$

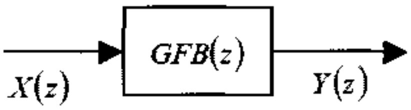


Figure 55 **Block diagram of feedback transfer function in the Z domain**

The feedback transfer function $GFB(z) = \dfrac{GC(z)G(z)}{1+GC(z)G(z)}$ ... (6.12), from the expansion of block $GFB(z)$ as shown in Figure 55.

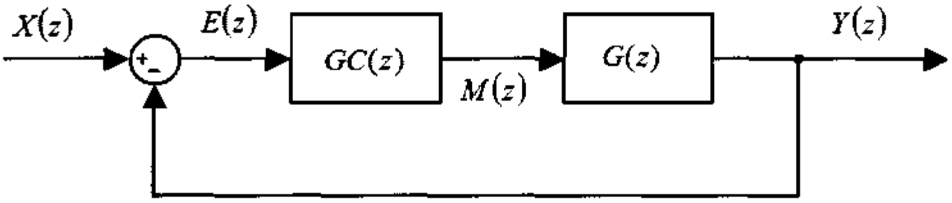


Figure 56 Block Diagram of digital feedback control system

Equation (6.11) must be written in the form of equation (6.12).

$$\frac{x}{1+x} = \frac{(1-e^{-bT})z^{-1}}{1-e^{-bT}z^{-1}}$$

$$\Rightarrow x(1-e^{-bT}z^{-1}) = (1+x)(1-e^{-bT})z^{-1}$$
$$\Rightarrow x - xe^{-bT}z^{-1} = z^{-1} - e^{-bT}z^{-1} + xz^{-1} - xe^{-bT}z^{-1}$$

$$\Rightarrow x - xz^{-1} = z^{-1} - e^{-bT}z^{-1}$$

$$\Rightarrow x(1-z^{-1}) = (1-e^{-bT})z^{-1}$$
$$\Rightarrow \quad x = GC(z)G(z) = \frac{(1-e^{-bT})z^{-1}}{1-z^{-1}} \quad ... \quad (6.13)$$

$G(z)$ Is the Z transfer function of the product of the zero order hold and the plant transfer function. The zero order hold transfer function is given by $\dfrac{1-e^{-Ts}}{s} = \dfrac{1-z^{-1}}{s}$.

Hence $G(z) = Z\left[\dfrac{1-z^{-1}}{s}.G(s)\right] = (1-z^{-1})Z\left[\dfrac{G(s)}{s}\right]$

To obtain the Z transform of $\frac{G(s)}{s}$, its inverse Laplace transform must first be found, that is a function in t. [65]

$$L^{-1}\left[\frac{G(s)}{s}\right] = L^{-1}\left[\frac{1}{s}.\frac{as+a^2+\omega^2}{s^2+2as+a^2+\omega^2}\right] \quad : \text{ (From equation } (6.4)\text{)}$$

$$L^{-1}\left[\frac{1}{s}.\frac{as+a^2+\omega^2}{s^2+2as+a^2+\omega^2}\right] = L^{-1}\left[\frac{1}{s} - \frac{(s+a)}{(s+a)^2+\omega^2}\right] : \text{ (Partial fraction expansion)}$$

$$= L^{-1}\left[\frac{1}{s}\right] - L^{-1}\left[\frac{s+a}{(s+a)^2+\omega^2}\right] = 1(t) - e^{-at}.\cos\omega t \qquad \ldots \qquad (6.14)$$

Therefore:

$$G(z) = Z\left[\frac{1-z^{-1}}{s}.G(s)\right] = (1-z^{-1}).Z\left[\frac{G(s)}{s}\right]$$
$$= (1-z^{-1}).Z[1(t) - e^{-at}.\cos\omega t] = (1-z^{-1}).Z[1(kT) - e^{-akT}.\cos\omega kT]$$

$$= (1-z^{-1}).[\ Z[1(kT)] - Z[e^{-akT}.\cos\omega kT]\ ]$$

$$= (1-z^{-1}).\left[\frac{1}{1-z^{-1}} - \frac{1-e^{-aT}.z^{-1}.\cos\omega T}{1-2e^{-aT}.z^{-1}.\cos\omega T + e^{-2aT}.z^{-2}}\right]$$

Make the following substitutions :

$$n_1 = e^{-aT}.\cos\omega T$$
$$m_1 = 2e^{-aT}.\cos\omega T$$
$$m_2 = e^{-2aT}$$

$$= \left(1-z^{-1}\right).\left[\frac{1}{1-z^{-1}} - \frac{1-e^{-aT}.z^{-1}.\cos\omega T}{1-2e^{-aT}.z^{-1}.\cos\omega T+e^{-2aT}.z^{-2}}\right]$$

$$= 1 - \frac{\left(1-z^{-1}\right)\left(1-n_1z^{-1}\right)}{1-m_1z^{-1}+m_2z^{-2}}$$

$$= \frac{1-m_1z^{-1}+m_2z^{-2}-\left[1-z^{-1}-n_1z^{-1}+n_1z^{-2}\right]}{1-m_1z^{-1}+m_2z^{-2}}$$

$$= \frac{\left(1+n_1-m_1\right)z^{-1}+\left(m_2-n_1\right)z^{-2}}{1-m_1z^{-1}+m_2z^{-2}} \qquad \ldots \qquad (6.15)$$

From equation (6.13) the controller $GC(z)$ is given by $\frac{\left(1-e^{-bT}\right)z^{-1}}{1-z^{-1}}.\frac{1}{G(z)}$. Hence:

$$GC(z) = \frac{\left(1-e^{-bT}\right)z^{-1}}{1-z^{-1}}.\frac{1}{G(z)} = \frac{\left(1-e^{-bT}\right)z^{-1}}{1-z^{-1}}.\frac{1-m_1z^{-1}+m_2z^{-2}}{\left(1+n_1-m_1\right)z^{-1}+\left(m_2-n_1\right)z^{-2}}$$

$$= \frac{\left(1-e^{-bT}\right)z^{-1}}{\left(1-z^{-1}\right)}.\frac{\left(1-m_1z^{-1}+m_2z^{-2}\right)}{z^{-1}\left(\left(1+n_1-m_1\right)+\left(m_2-n_1\right)z^{-1}\right)}$$

$$= \frac{\left(1-e^{-bT}\right)-m_1\left(1-e^{-bT}\right)z^{-1}+m_2\left(1-e^{-bT}\right)z^{-2}}{\left(1+n_1-m_1\right)\left(1-z^{-1}\right)+\left(m_2-n_1\right)z^{-1}\left(1-z^{-1}\right)}$$

$$= \frac{\left(1-e^{-bT}\right)-m_1\left(1-e^{-bT}\right)z^{-1}+m_2\left(1-e^{-bT}\right)z^{-2}}{\left(1+n_1-m_1\right)-\left(1+n_1-m_1\right)z^{-1}+\left(m_2-n_1\right)z^{-1}-\left(m_2-n_1\right)z^{-2}}$$

$$= \frac{\left(1-e^{-bT}\right)-m_1\left(1-e^{-bT}\right)z^{-1}+m_2\left(1-e^{-bT}\right)z^{-2}}{\left(1+n_1-m_1\right)-\left(1+2n_1-m_1-m_2\right)z^{-1}-\left(m_2-n_1\right)z^{-2}} \qquad \ldots \qquad (6.16)$$

Make the following substitutions:

$$\begin{aligned}
p_0 &= 1-e^{-bT} \\
p_1 &= m_1\left(1-e^{-bT}\right) && = 2e^{-aT}\left(1-e^{-bT}\right)\cos \omega T \\
p_2 &= m_2\left(1-e^{-bT}\right) && = e^{-2aT}\left(1-e^{-bT}\right)
\end{aligned}$$

$$\begin{aligned}
q_0 &= 1+n_1-m_1 && = 1+e^{-aT}.\cos \omega T - 2e^{-aT}.\cos \omega T && = 1-e^{-aT}.\cos \omega T \\
q_1 &= 1+2n_1-m_1-m_2 \\
&= 1+2e^{-aT}.\cos \omega T - 2e^{-aT}.\cos \omega T - e^{-2aT} && && = 1-e^{-2aT} \\
q_2 &= m_2-n_1 && = e^{-2aT}-e^{-aT}.\cos \omega T
\end{aligned}$$

$$GC(z) = \frac{p_0 - p_1 z^{-1} + p_2 z^{-2}}{q_0 - q_1 z^{-1} - q_2 z^{-2}} \qquad \ldots \qquad (6.17)$$

6.2.4 Controller Implementation

There are 2 approaches to the controller implementation: these are direct and standard programming. This controller was implemented using the standard programming approach, which uses the minimum number of delay elements, and in this case it is 2. [65]

$$GC(z) = \frac{p_0 - p_1 z^{-1} + p_2 z^{-2}}{q_0 - q_1 z^{-1} - q_2 z^{-2}}$$

$$= \frac{M(z)}{E(z)} = \frac{M(z)}{H(z)}.\frac{H(z)}{E(z)} = \frac{p_0 - p_1 z^{-1} + p_2 z^{-2}}{1}.\frac{1}{q_0 - q_1 z^{-1} - q_2 z^{-2}}$$

Where

$$\frac{M(z)}{H(z)} = \frac{p_0 - p_1 z^{-1} + p_2 z^{-2}}{1} \qquad \ldots \qquad (6.18)$$

And:

$$\frac{H(z)}{E(z)} = \frac{1}{q_0 - q_1 z^{-1} - q_2 z^{-2}} \qquad \ldots \qquad (6.19)$$

Equation (6.18) may be written as:

$$M(z) = p_0 H(z) - p_1 z^{-1} H(z) + p_2 z^{-2} H(z) \qquad \ldots \qquad (6.20)$$

Its block implementation is:

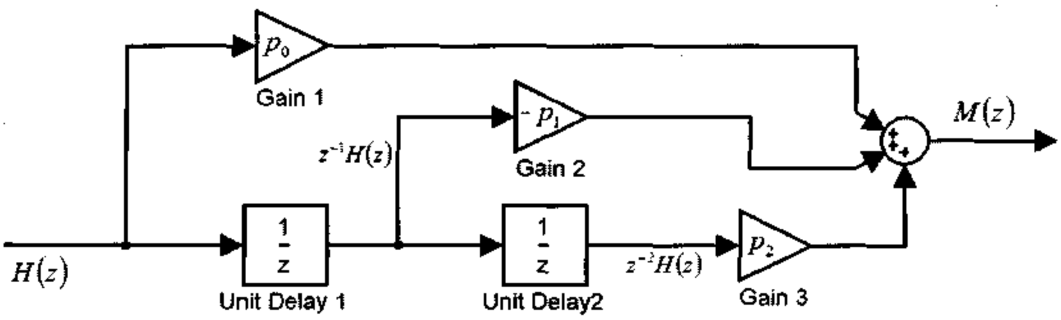


Figure 57 Block implementation of equation (6.20)

Equation (6.19) may be written as:

$$E(z) = q_0 H(z) - q_1 z^{-1} H(z) - q_2 z^{-2} H(z)$$

Rearranging to make $q_0 H(z)$ the subject of the formula:

$$q_0 H(z) = E(z) + q_1 z^{-1} H(z) + q_2 z^{-2} H(z) \qquad \ldots \qquad (6.21)$$

The block diagram of equation (6.21) is:

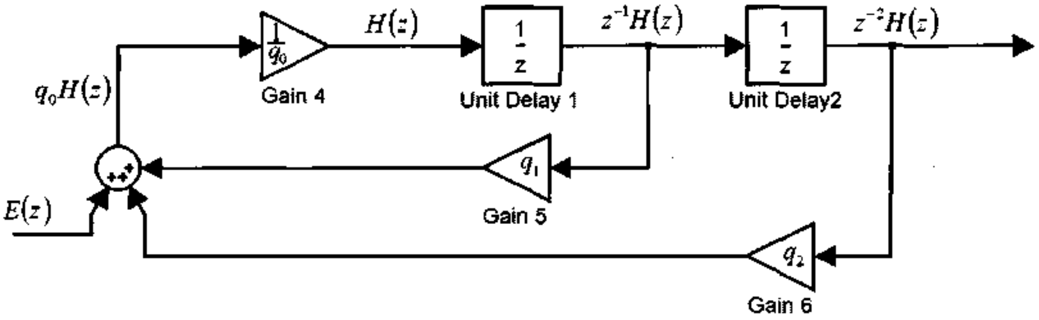


Figure 58 **Block diagram implementation of equation** (6.21)

Combining the block diagrams realizes the digital controller [65]:

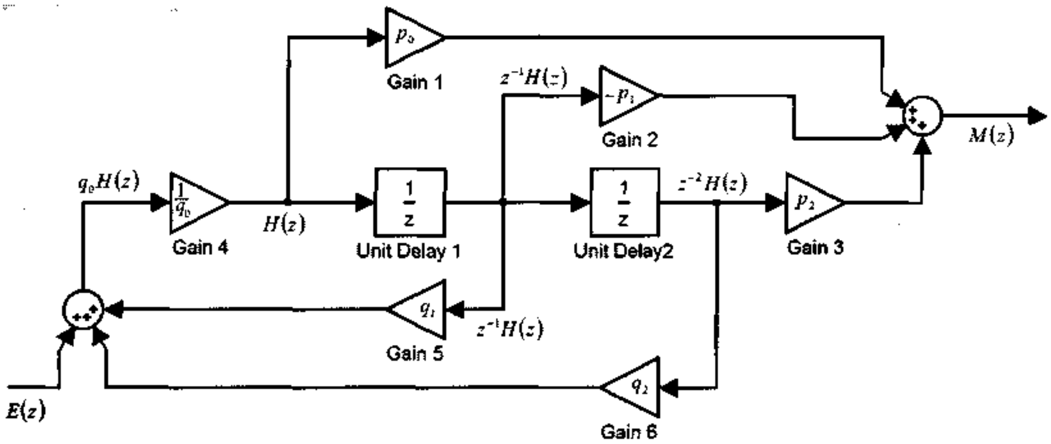


Figure 59 **Combined block diagram realizing discrete time controller**

6.3 Chapter Summary

The control system for the PKM was discussed and the method of obtaining the transfer function of the motor (with leg masses attached) was described. The method was based on standard linear control theory, and the system was proved to be linear. A mathematical function was fitted to the data. The Laplace transforms and Z transforms were found, and the control system was designed around these functions both in the S domain and the Z domain. This was done to evaluate the digital controller against a continuous time controller. Simulation results are shown in section 8.2.5.

7 Software

Various stages of this project required the use of different software packages. The 3 software packages used were MATLAB, Visual Basic (VB) and CODEVISION CAVR. MATLAB was used to simulate and resolve various unknown aspects of the mechanical design. Visual Basic was used to code the user interface. CAVR was used to program the embedded controllers.

7.1 MATLAB Software

7.1.1 Forward and Inverse Kinematics

M-files (MATLAB code) were written to solve both the forward and inverse kinematics. These 2 functions formed the basis for all the code that followed. These functions were a direct translation of the mathematical modelling used to solve the kinematics (sections 4.3.2 c, d and g). To test the functions and gauge whether they were coded correctly as well as verify the solution of the kinematics, a check was performed which was as follows. Given coordinates for the end effector, the inverse kinematics function was used to solve the angular values for each of the upper legs. These angular values were then input to the forward kinematics function, whose output is the end effector position. Both sets of coordinates were the same for all coordinates tested and it verified the correctness of both functions in terms of programming, mathematics, and the geometric model.

7.1.2 Workspace Envelope

The workspace envelope was visualised using the inverse kinematics as this made it easier to graph the data using commands in MATLAB. For each set of XY coordinates in a rough estimate of the workspace the minimum and maximum Z positions that could be reached were found. These were calculated by moving inwards from positions that could not be reached. The matrix data found were then combined and plotted using the SURF command, this was described in section 4.3.2 j.

7.1.3 Vibration

The effects of oscillatory vibration of the actuators needed to be determined. This oscillation is a sinusoidal function with magnitude α, frequency ω and phase shift θ, i.e. $\alpha\sin(\omega t+\theta)$. When given an end effector position, the inverse kinematics function was used to solve for the upper leg angles. The sinusoidal offset (with possibly different α, ω and θ) was then added to these angles. At time t_i the new angles are calculated (with offset added) and fed as input to the forward kinematics which calculates the position in space of the end effector. The spatial displacements of the end effector in X, Y and Z are then calculated when compared to the original position without vibration. These spatial offsets are then plotted to see the effect of the vibration. See section 8.2.2 for results.

7.1.4 Trajectory Simulation for Solid Edge (SE)

To create a graphical simulation in SE of the mechanical system moving on a trajectory, an M-file was written to solve the inverse kinematics for points along that trajectory. Each set of angular values found was time stamped and saved in a text file, 4 text files in all, one for each leg. These text files were then loaded into SE, which uses a spline interpolator to fit curves to these points in parts. A curve is drawn for every set of 4 consecutive points. Overlapping curves are averaged to get a continuous smooth curve. This curve represents time stamped positional information in step sizes that SE requires to create a smooth transition between graphic illustrations of mechanical configuration. See section 8.2.4 for the mechanical simulation.

The points along the designed trajectory had to have a small step size (spatial distance between points). If the step size was too large the end effector tends to bob from position *i* to position *i+1*. The reason for this lies in the structure of the machine. For a linear change in position the change in angular values of the upper legs varies in some way that the spline interpolation in SE cannot fit exactly.

7.2 Visual Basic (VB) Software

The user interface for the PKM consists of a few graphical controls and background functions. The graphical window controls allow manual direction of the end effector in its 3D (XYZ control) workspace and a visualisation of data received on its current position. As the intention is to only control the position of the end effector, only the inverse kinematics function in MATLAB was translated into VB code.

7.2.1 Graphical Control of End Effector

Two VB controls combined allow the user to control the 3 XYZ coordinates of the end effector. A picture box control with cross hair and bull's eye allow the user to control the XY coordinates. Left and right arrow keys allow the user to change the Z coordinate. Once the user left clicks on the bull's eye it attaches to the mouse pointer and a second left click releases the bull's eye. Once the bull's eye is attached to the mouse pointer a right click invokes the inverse kinematics function and solves the upper leg angle values for the current position selected in software. If these values are in the correct range it then converts these angles to a value between 0 and 255, and transfers the data to the embedded controller.

7.2.2 Graphical Display of Data

Each sensor in space is represented by a coloured circle in one of 2 picture boxes, indicating either vertical or horizontal screen data. The XYZ coordinates being controlled are also indicated on these controls via cross hairs. The horizontal data screen has XY coordinate information superimposed on it. The vertical data screen has the YZ coordinate information superimposed on it. The picture boxes were properly scaled to represent the data in its correct coordinates.

7.2.3 Transfer of Control Signals / Receiving Data

The control signals are start, stop and angular data for rotation. A value between 0-255 representing an angle is transferred via ASCII characters. For example the value 135 is sent as 49 (1), 51 (3), 53 (5) and 88 (X) which is a completion character.

The data received from the embedded controller indicates a reference number for a sensor that is stimulated and a reference for the plane of data, i.e. vertical or horizontal. The lasers on the end effector were arranged so that at most only one sensor is stimulated per screen. The reference number for each sensor on each screen may therefore be represented by a single byte of data (0-255) as there are 256 sensors per screen. When the data processing indicates that no sensor has been hit, a separate signal is sent (e.g. an ASCII character for a letter).

The MSCOMM control in VB handles the process of sending and receiving data through the RS232 serial communications port.

7.2.4 Calculation of Inverse Kinematics

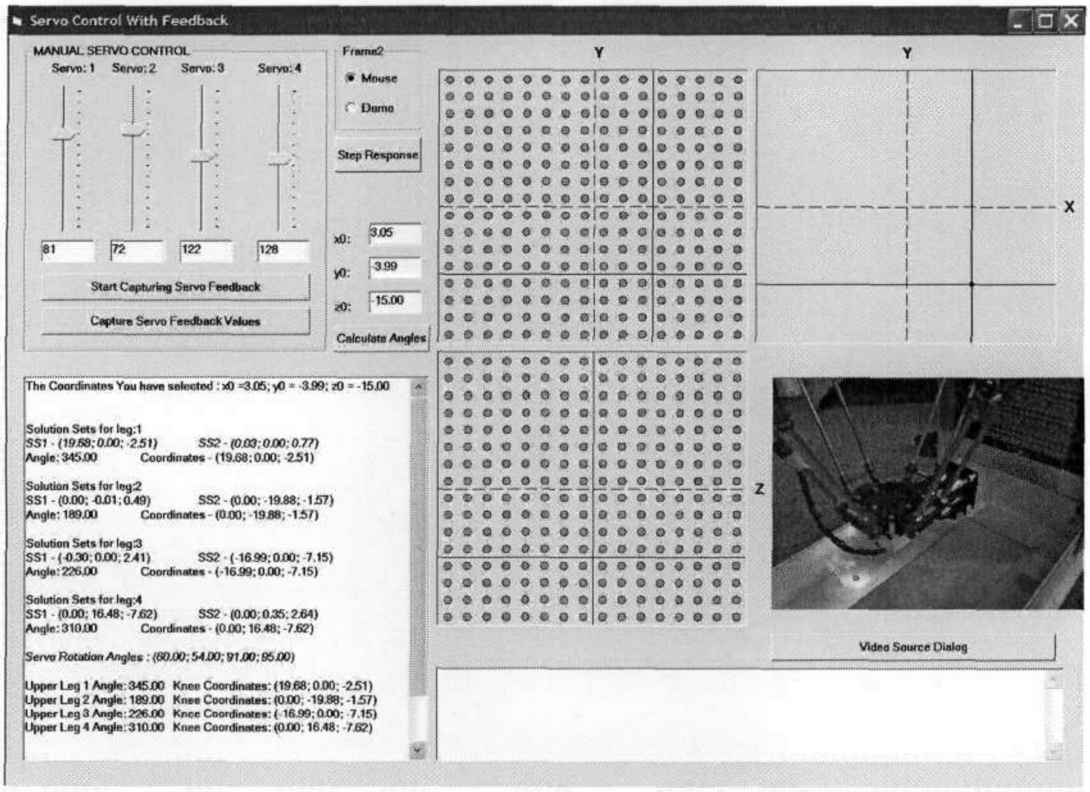
This function is a direct translation of that used in MATLAB to solve the inverse kinematics and make the angles available in VB.

7.2.5 Video Display

A video feed from a webcam was made available through a free open source software component created by E. J. Bantz for Visual Basic programmers. It accesses the driver API (application programming interface) and allows the software to display the video feed directly in a control window.

7.2.6 Software Calibration

This routine calibrates the coordinate system, that is, it aligns the software coordinate system with the real world system of the sensor screens. It is initiated with a key press, the F1 key which has a value of 112. This is explained further in section 8.1.2.



Servo: 1: 81
Servo: 2: 72
Servo: 3: 122
Servo: 4: 128

x0: 3.05
y0: -3.99
z0: -15.00

```
The Coordinates You have selected : x0 =3.05; y0 = -3.99; z0 = -15.00


Solution Sets for leg:1
SS1 - (19.68; 0.00; -2.51)          SS2 - (0.03; 0.00; 0.77)
Angle: 345.00          Coordinates - (19.68; 0.00; -2.51)

Solution Sets for leg:2
SS1 - (0.00; -0.01; 0.49)           SS2 - (0.00; -19.88; -1.57)
Angle: 189.00          Coordinates - (0.00; -19.88; -1.57)

Solution Sets for leg:3
SS1 - (-0.30; 0.00; 2.41)           SS2 - (-16.99; 0.00; -7.15)
Angle: 226.00          Coordinates - (-16.99; 0.00; -7.15)

Solution Sets for leg:4
SS1 - (0.00; 16.48; -7.62)          SS2 - (0.00; 0.35; 2.64)
Angle: 310.00          Coordinates - (0.00; 16.48; -7.62)

Servo Rotation Angles : (60.00; 54.00; 91.00; 95.00)

Upper Leg 1 Angle: 345.00  Knee Coordinates: (19.68; 0.00; -2.51)
Upper Leg 2 Angle: 189.00  Knee Coordinates: (0.00; -19.88; -1.57)
Upper Leg 3 Angle: 226.00  Knee Coordinates: (-16.99; 0.00; -7.15)
Upper Leg 4 Angle: 310.00  Knee Coordinates: (0.00; 16.48; -7.62)
```

Figure 60 Screenshot of GUI PC controller

Figure 60 illustrates the PC controller GUI. The text box at the bottom left hand corner displays the results of the inverse kinematics solution. The picture boxes in the middle with the blue dots illustrate the sensors. The one on top represents the horizontal screen, with X and Y axes. The one at the bottom represents the vertical screen, with Z and X axes. Superimposed in both representations of the screens are the position cross hairs. This indicates the alignment of the software coordinate system with real world positioning. The picture box on the top right corner indicates the XY coordinate mouse control. The dashed lines are the X and Y reference axes. The black solid lines with the blue BE represents the XY coordinates the user wishes to move to. When controlling the robot the BE is attached to the mouse pointer. The Z coordinate is controlled with the left and right arrow keys, which move it up and down respectively. Vertical lines on the vertical detector screen representation indicate constant Z values. The picture box at the bottom right displays video captured from a webcam.

7.3 CAVR Embedded Software

7.3.1 Command Interpretation / Data Reception

Only 2 commands are received, for starting and stopping mechanical control.

Data received is for controlling the angles of the servos. The angle for each servo is indicated by a character, followed by its digits in order of significance (i.e. hundreds, tens then units), and lastly by a completion character. The value is then built up in the embedded system software and used as the reference in the control algorithm.

7.3.2 Shift Register control

The 74LS166 function table in its datasheet illustrates how to control this IC and the procedure used is outlined. Each 74LS166 must be cleared with a low applied to the CLEAR pin. The CLOCK pin is enabled with a low to the CLOCK INHIBIT pin. To load parallel data into the shift register, the SHIFT/LOAD line is pulled low and the register clocked (low to high transition of pin CLOCK). The register is then set to shift data when the SHIFT/LOAD line is pulled high. The data is now shifted through the output QH, bit at a time, most significant bit (MSB) first with every clock signal. The register has to be clocked 8 times to read the 8 bits of each register representing 8 sensor current states. As there are 32 registers per detector screen, and as each serial output feeds into the serial input of the following register (the last serial output is fed to the microcontroller), the registers are clocked 256 times (32×8).

7.3.3 Data processing

The data is processed simultaneously when it is read from the parallel to serial converters. All the data from one detector screen is read on one line (as the output from each parallel to serial converter is fed to the serial input of the following register). When a bit is read its value is checked (is it 1 or 0), if it is 1 then that sensor, which is represented by the number of times the registers have been clocked at this point, is a sensor that has been stimulated. All the sensors are checked in this manner. All the reference values for those sensors with bit values of 1 are then transferred to the host PC control system. If no sensors are stimulated then the control system sends a character code indicating this.

7.3.4 Data Transfer

Data is transferred in the same way it is received i.e. via ASCII characters representing the decimal digits. The data transferred are the reference numbers for the sensors that have been stimulated. The angles measured by the ADC are only used by the control system. The control system makes these measured angles follow the reference input angles.

7.3.5 ADC control

The procedure for setting up and reading the ADC is as follows. All 4 ADCs are read simultaneously. Four temporary variables are used to store the data read from the ADCs, once read the values are stored in the reference variables.

The clocking pin (I/OC) is set low. $\overline{\text{CS}}$ (Read as not chip select) is initially high. This allows for the input voltage to be sampled continuously and digitized. This pin is now set low which stops the conversion process and allows the data to be read. The data is shifted out serially through the data output pin DO, from most significant bit to least significant bit. These bits are read into the temporary variables, with each clock of pin I/OC. The ADCs are clocked 8 times to read the 8 bits. Once this is done the values are stored to the output feedback values used in the control algorithm. $\overline{\text{CS}}$ is then set high to allow another conversion. The ADCs are read every 2 ms.

7.3.6 PWM Generation

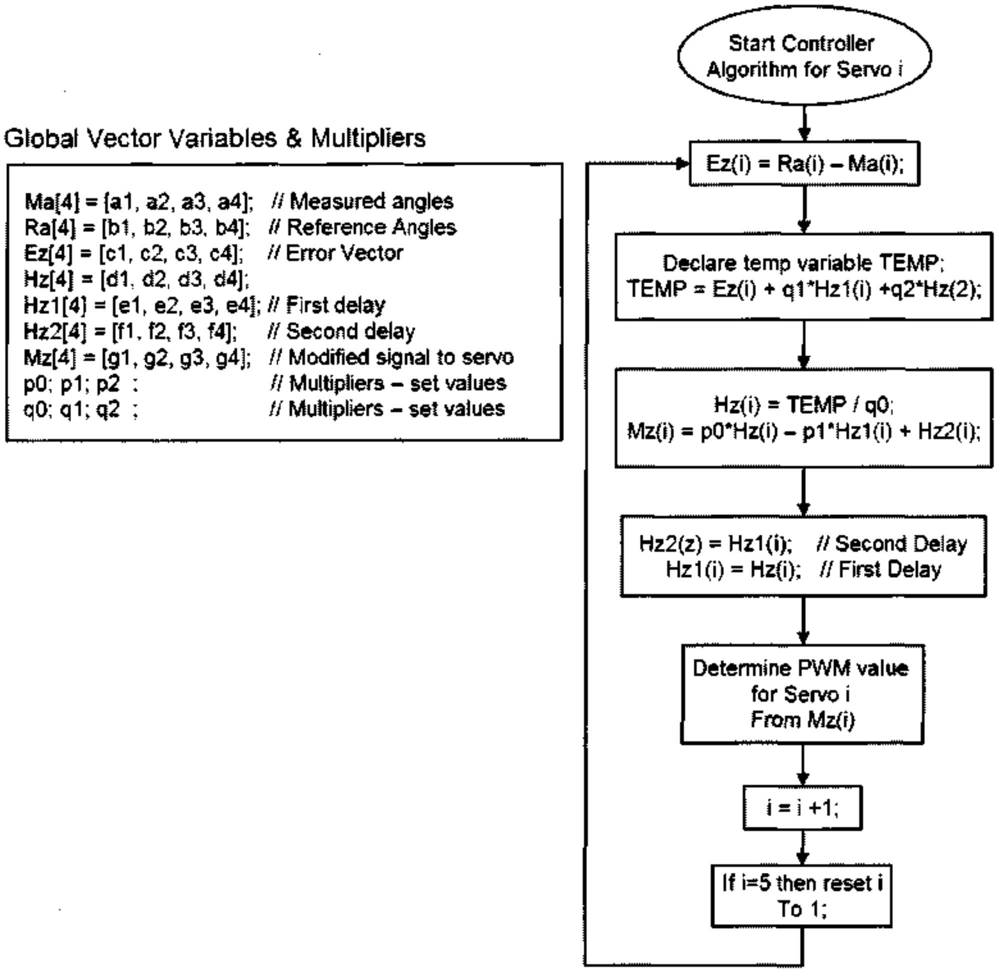
The timers were used without any pre-scaling, i.e. the frequency of the crystal resonator that clocks the microcontroller was not divided. This resonator has a 16 MHz frequency or a period of 0.0625 us (micro seconds). The 16 bit timer interrupt register increments every clock cycle or every 0.0625 us for a clock with no pre-scaling. Two 16 bit timers were used to generate the PWM signals.

There are 3 more 16 bit registers (compare interrupt registers A, B and C), the values of which are constantly compared to the 16 bit timer interrupt register. This functionality was built into the architecture of the microcontroller. There are 3 compare interrupts and when there is a match with the value stored in either of the compare interrupt registers with the value currently in the timer interrupt register, the corresponding compare interrupt is executed.

The compare interrupts A and B of each timer generates the PWM signals for the servos. Compare interrupt C is set at 2 ms, to reset the interrupt register. This 2 ms period represents the time in which corrections are made in the control algorithm. As the rotation angles were limited in value the full 2 ms period for a 180° rotation was not necessary.

7.3.7 Control Algorithm

The algorithm of the digital controller indicated by the digital control block diagram of Figure 59 is coded. It consists of 2 delays, and multiple additions and multiplications. A single function was written to perform the control algorithm and is shown in Figure 61.



Global Vector Variables & Multipliers

```
Ma[4] = {a1, a2, a3, a4};   // Measured angles
Ra[4] = [b1, b2, b3, b4];   // Reference Angles
Ez[4] = [c1, c2, c3, c4];   // Error Vector
Hz[4] = [d1, d2, d3, d4];
Hz1[4] = [e1, e2, e3, e4]; // First delay
Hz2[4] = [f1, f2, f3, f4];    // Second delay
Mz[4] = [g1, g2, g3, g4];   // Modified signal to servo
p0; p1; p2  ;                   // Multipliers – set values
q0; q1; q2  ;                   // Multipliers – set values
```

Start Controller
Algorithm for Servo i
Ez(i) = Ra(i) – Ma(i);
Declare temp variable TEMP;
TEMP = Ez(i) + q1*Hz1(i) +q2*Hz(2);
Hz(i) = TEMP / q0;
Mz(i) = p0*Hz(i) – p1*Hz1(i) + Hz2(i);
Hz2(z) = Hz1(i);   // Second Delay
Hz1(i) = Hz(i);   // First Delay
Determine PWM value
for Servo i
From Mz(i)
i = i +1;
If i=5 then reset i
To 1;

Figure 61 Function of embedded controller

7.4 Chapter Summary

This chapter describes how 3 software development languages were used in this Mechatronics project. VB was used to code and design the PC controller GUI. MATLAB was used to simulate and solve many of the mechanical issues related to the mechanical design, i.e. kinematics, vibration, SE Simulation and workspace. Once the IK solution was verified, it was then written in VB. CAVR, a C compiler for ATMEL microchips was used to code the program for the embedded controller, which included data acquisition, processing and control of motors.

8 Calibration, Simulation Results, Prototyping and Performance Tests

8.1 Calibration

The calibration of the system has 2 parts: the mechanical positioning calibration of the legs and the software coordinate calibration.

8.1.1 Mechanical Calibration

The mechanical calibration ensures that all legs rotate the same angular degrees for each reference angle input to the system. The range of motion was limited, however this does not infringe on the workspace specification mentioned in section 1.5.1.

The total angular range of motion was limited to 140°.The maximum actuation angle is set at 75° from the positive vertical in the clockwise direction (see Figure 23 for a depiction of the actuator reference for rotation). The minimum actuation angle is set at 140° from that maximum (clockwise) or 215° from the vertical, shown in Figures 62. The leg rotates from 215° to 75° counter clockwise.

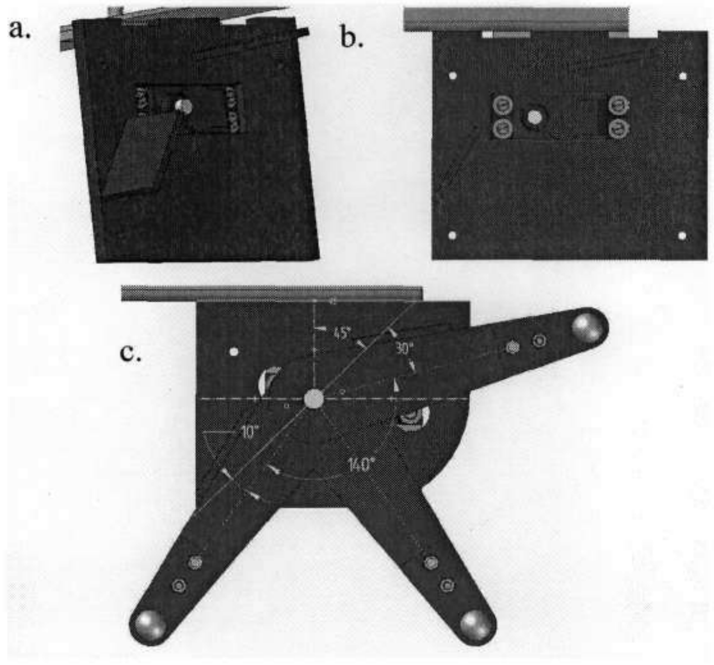


Figure 62 Upper leg indicating rotation limits

a. 3D View of upper leg alignment tool

b. Front view of upper leg alignment tool

c. Front view of alignment tool indicating rotation limits

At this minimum position, for each leg, the voltage read from the analogue feedback potentiometer was set as the negative reference to the corresponding servo ADC. Similarly the voltage at the maximum position was read for each servo and set as the positive reference to that servo's ADC. Each ADC converts a voltage representing an angle in the range of 0° to 140°, to an 8 bit digital value. The angular positioning resolution achieved was $\frac{140°}{255} \cong 0.55°$ (see section 8.2.1 for the effect of this quantisation). A mechanical alignment tool was made to ensure that the minimum and maximum angles were uniform for all legs. This tool was laser cut from 3 mm Perspex, and is shown in Figure 62 a-c.

Another important aspect of the mechanical calibration was that of aligning the detector screens. The mechanical designs in SE were accurate and the screens were aligned in this CAD package. The relative displacements from the edges of the mechanical frame were then measured and these measurements were used to position the real screens on the mechanical rig. In this manner each column of detectors on the horizontal screen was aligned with the column detectors on the vertical screen.

8.1.2 Software Calibration

The software calibration aligns the real world coordinate system with the software coordinate system. Errors may exist with the alignment of the screens and these have to be accounted for. These are relative errors that are fixed in value, and are corrected by either adding or subtracting the offset. To find the offset the control software moves the end effector to 4 points in space where it expects to be sensed by both the vertical and horizontal screen. For each set of coordinates (reference coordinates) if no sensor is found, the control software enters a horizontal spiralling routine to find a horizontal sensor. The y coordinate of the vertical detector screen should be the same if the detector screens are aligned properly. Once the horizontal sensor is found it moves the end effector up and down until it finds the vertical sensor for that coordinate set. If it does not find a vertical sensor, then the screens are not aligned properly and this must be corrected. If it finds the corresponding sensor (real world measured coordinates), the errors between the reference coordinates and the measured coordinates are then set as the error offsets in X, Y and Z. This procedure is followed for each of the 4 coordinate sets. The errors should read the same if not there is a problem with alignment once again.

This procedure aligns the reference system with the real physical system. It does not affect, restrict or reduce the motion of the end effector in anyway.

8.2 Simulation Results

Various aspects of the design were simulated in 2 software packages. The mechanical design was realized in SE, a CAD package. Once the design was complete it was used to simulate and provide a 3D visualization of the movement of the modelled PKM.

MATLAB was used to solve the kinematics which was the basis for simulating and determining other aspects of the design. SIMULINK, a simulation add-on to MATLAB, was used to design the digital controller and simulate the design in both the S and Z domains.

8.2.1 Forward and Inverse Kinematics Solver

A number of function m-files were written to solve the kinematics of the machine. The forward kinematics solutions were used to verify the inverse kinematics solutions and vice versa. When the inverse kinematics function is run in MATLAB the user is prompted for a set of end effector coordinates. The coordinates $(-7,6,-20)$ were input and the results follow:

```
MENU ...

1. Inverse Kinematics (with Forward Kinematics Check)
2. Forward Kinematics (with Inverse Kinematics Check)
3. Vibration Model  ... 1

Enter end effector coordinates ...
x Range: -7.5 to 7.5        y Range: -7.5 to 7.5        z Range: -22 to -12

x0 : -7
y0 : 6
z0 : -20

The coordinates you have selected ... [ -7.0000    6.0000    -20.0000]

Angle1 = 275.7000              [x1 y1 z1]  = [ 10.9932    0.0000    -9.9506]
Angle2 = 261.6058              [x2 y2 z2]  = [ 0.0000    -11.4598    -9.8929]
Angle3 = 208.7283              [x3 y3 z3]  = [-18.7691    0.0000    -4.8066]
```

```
Angle4 = 326.4105                [x4 y4 z4]  = [ 0.0000     18.3302    -5.5324]


Checking solution with Forward Kinematics...


[Angle1   Angle2   Angle3   Angle4]   =  [275.7000     261.6058     208.7283     326.4105]


Angle1 = 275.7000               [x1 y1 z1] = [10.9932     0.0000    -9.9506]

Angle2 = 261.6058               [x2 y2 z2] = [ 0.0000    -11.4598   -9.8929]

Angle3 = 208.7283               [x3 y3 z3] = [-18.7691    0.0000    -4.8066]

Angle4 = 326.4105               [x4 y4 z4] = [ 0.0000    18.3302    -5.5324]


Convert to Servo Rotation Angles....

[Angle1   Angle2   Angle3   Angle4]   =  [129.3000     126.6058     73.7283     78.5895]
```

The angular rotation values are with respect to the coordinate system. These values have to be converted to byte values that can be sent to the servo motors. Note that the servos have a 140° degree range of motion due to the mechanical calibration apparatus, mentioned in section 8.1.1. The effect of the quantisation of the ADC leads to uncertainty in the actual angular positions. The angular quantization steps are 0.55°, hence the maximum quantisation error is $\frac{0.55^\circ}{2} = 0.275^\circ$. To gauge the effect of this quantisation error, the angles obtained above were then quantised and input to the forward kinematics to see the relative changes in the end effector coordinates X, Y and Z. The angles were modified with the function $\left\lfloor \frac{Angle\_i}{0.275} \right\rfloor \times 0.275$. The floor function removes the remainder from division so $\left\lfloor \frac{Angle\_i}{0.275} \right\rfloor$ is an integer number. The servo rotation angles when modified are 129.2500, 126.5000, 73.7000 and 78.3750. Applying these angles to the forward kinematics yields:

```
MENU ...


1. Inverse Kinematics (with Forward Kinematics Check)

2. Forward Kinematics (with Inverse Kinematics Check)

3. Vibration Model  ... 2
```

```
Enter 4 actuation Angles ...          Servo Angles Range: 30 - 170 Degrees (140 Degree Range)

Servo Angle 1 : 129.2500
Servo Angle 2 : 126.5000
Servo Angle 3 : 73.7000
Servo Angle 4 : 78.3750

Convert to Coordinate System Angles....
[Angle1   Angle2   Angle3   Angle4]  =  [275.7500     261.5000     208.7000     326.6250]

[x0  y0  z0] = [ -6.9946     6.0139     -19.9864]
Angle1 = 275.7500                [x1 y1 z1] = [ 11.0019     0.0000     -9.9497]
Angle2 = 261.5000                [x2 y2 z2] = [  0.0000    -11.4781    -9.8902]
Angle3 = 208.7000                [x3 y3 z3] = [-18.7715     0.0000     -4.8022]
Angle4 = 326.6250                [x4 y4 z4] = [  0.0000    18.3509     -5.5012]

Checking solution with Inverse Kinematics...

Angle1 = 275.7500                [x1 y1 z1] = [ 11.0019     0.0000     -9.9497]
Angle2 = 261.6081                [x2 y2 z2] = [  0.0000    -11.4594    -9.8929]
Angle3 = 208.7000                [x3 y3 z3] = [-18.7715     0.0000     -4.8022]
Angle4 = 326.5304                [x4 y4 z4] = [  0.0000    18.3418     -5.5149]

[x0  y0  z0] = [ -6.9946     6.0139     -19.9864]
```

From this it can be seen that the errors in position are:

$$\Delta x = -6.9946 - (-7) = 0.0054\,\text{cm}$$

$$\Delta y = 6.0139 - 6 = 0.0139\,\text{cm}$$

$$\Delta z = -19.9864 - (-20) = 0.0136\,\text{cm}$$

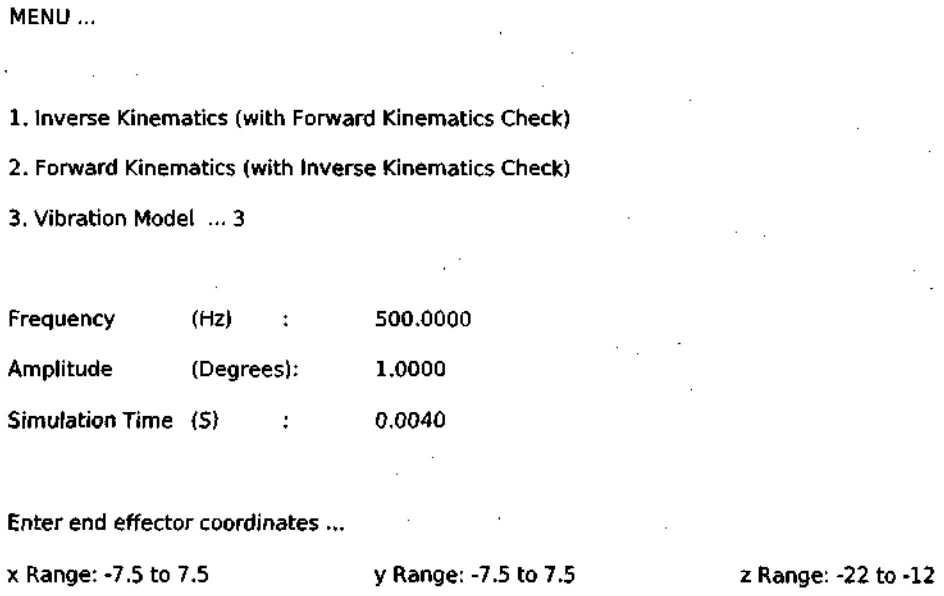
The uncertainty in position is small in relation to the span of motion. To determine the maximum effect of this error the end effector is moved vertically. The coordinates $(0,0,-20)$are investigated. The rotation angles come out as 90.9622, 90.9622, 90.9622 and

90.9622. Quantizing these angles yields values of 90.7500, 90.7500, 90.7500 and 90.7500, which are input to the forward kinematics and yield end effector coordinates of $(-0.0000, -0.0000, -19.9482)$. The X and Y errors are insignificant. The error in the Z coordinate is $\Delta z = -19.9482 - (-20) = 0.0518$ cm. This was repeated for various end effector coordinates and the results were similar. So from a mathematical standpoint using an 8 bit ADC over a range of 140° produces uncertainty errors that are far less that the specifications mentioned in section 1.5.1 for a perfect geometric model of the system.

The forward and inverse kinematics software solutions verify each other and proved that the mathematical solutions behind them are sound. This was crucial as these solutions lay the groundwork for other simulations that follow.

8.2.2 Vibration

The frame of the PKM significantly outweighs the moving parts of the machine. Any effect on the frame from the motors and during motion of the end effector cannot be detected visually. Vibration on the end effector results from oscillation of the upper arms of the machine. Another m-file was written to gauge the effect of a sinusoidal oscillation on each leg. This function was explained in section 7.1.3. The PWM signals are synchronized so the frequency of oscillation of each upper leg is the same and there is no phase shift. As each PWM signal is refreshed every 2 ms the frequency is 500 Hz. A vibration with a 1 degree amplitude is simulated and the results shown below. Figures 63 and 64 display the results.

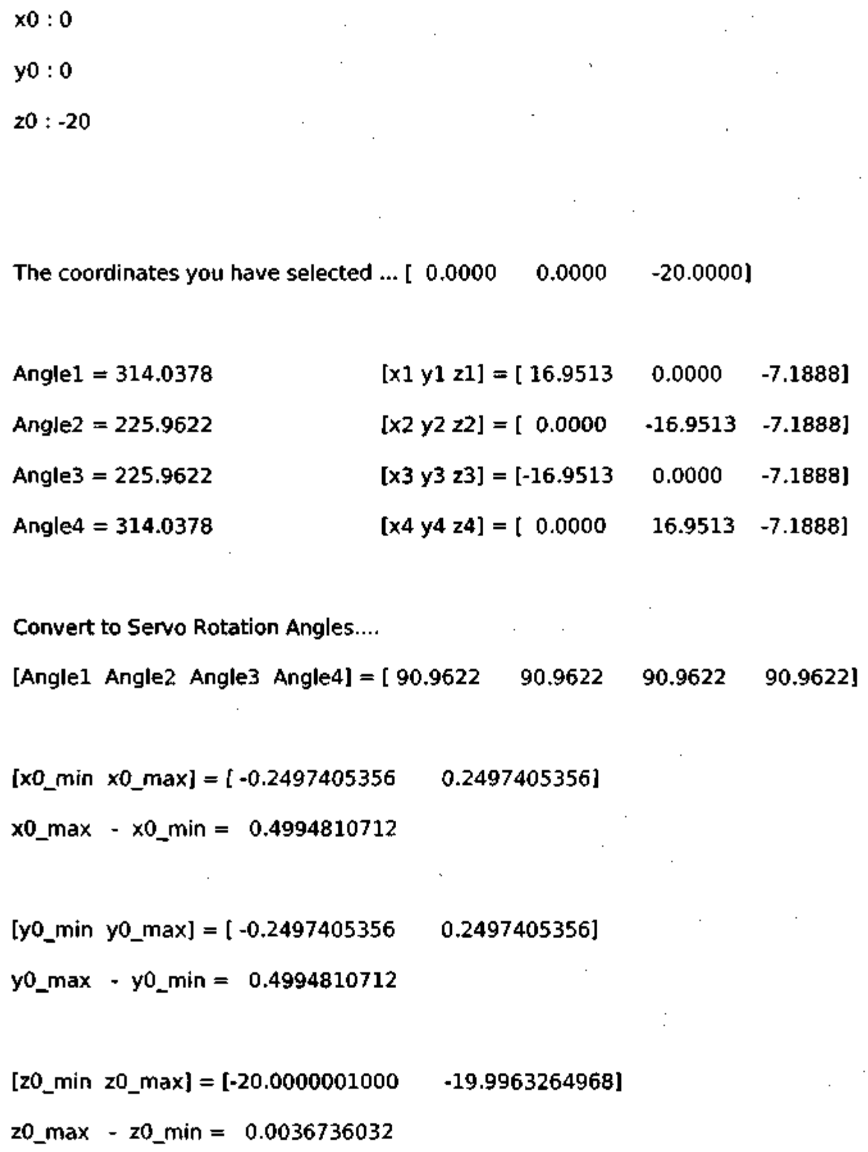


```
MENU ...

1. Inverse Kinematics (with Forward Kinematics Check)
2. Forward Kinematics (with Inverse Kinematics Check)
3. Vibration Model ... 3

Frequency        (Hz)       :      500.0000
Amplitude        (Degrees):        1.0000
Simulation Time  (S)        :      0.0040

Enter end effector coordinates ...
x Range: -7.5 to 7.5               y Range: -7.5 to 7.5               z Range: -22 to -12
```



```
x0 : 0

y0 : 0

z0 : -20


The coordinates you have selected ... [ 0.0000    0.0000    -20.0000]


Angle1 = 314.0378            [x1 y1 z1] = [ 16.9513    0.0000    -7.1888]

Angle2 = 225.9622            [x2 y2 z2] = [  0.0000  -16.9513   -7.1888]

Angle3 = 225.9622            [x3 y3 z3] = [-16.9513    0.0000    -7.1888]

Angle4 = 314.0378            [x4 y4 z4] = [  0.0000   16.9513   -7.1888]


Convert to Servo Rotation Angles....

[Angle1  Angle2  Angle3  Angle4] = [ 90.9622    90.9622    90.9622    90.9622]


[x0_min  x0_max] = [ -0.2497405356      0.2497405356]

x0_max  - x0_min =  0.4994810712


[y0_min  y0_max] = [ -0.2497405356      0.2497405356]

y0_max  - y0_min =  0.4994810712


[z0_min  z0_max] = [-20.0000001000     -19.9963264968]

z0_max  - z0_min =  0.0036736032
```

From the simulation it can be seen that the displacement between the minimum and maximum positions along the X and Y axes are about 5 mm. That displacement along the Z axis is negligible. Figure 64 indicates that the X_0 and Y_0 displacements are in phase which implies a diagonal movement with total displacement of 7.07 mm ($5\sqrt{2}$). Interestingly the Z displacement, although negligible in magnitude, has a frequency twice that of the oscillation at the legs. From this it can be seen that it is possible to reduce components of vibration error in PKMs, unlike serial machines which are always additive. This is done by change of phase in vibration from each leg or refreshing the PWM signal to each leg at different times.

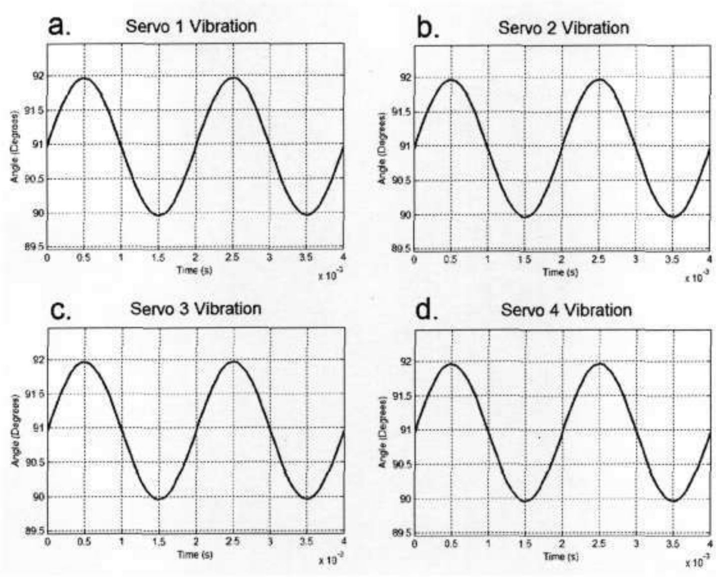


Figure 63 **PKM Legs angular variations at position** $(0, 0, -20)$, **with a 1 degree, 500 Hz vibration**

a.	Servo 1 vibration	b.	Servo 2 vibration
c.	Servo 3 vibration	d.	Servo 4 vibration

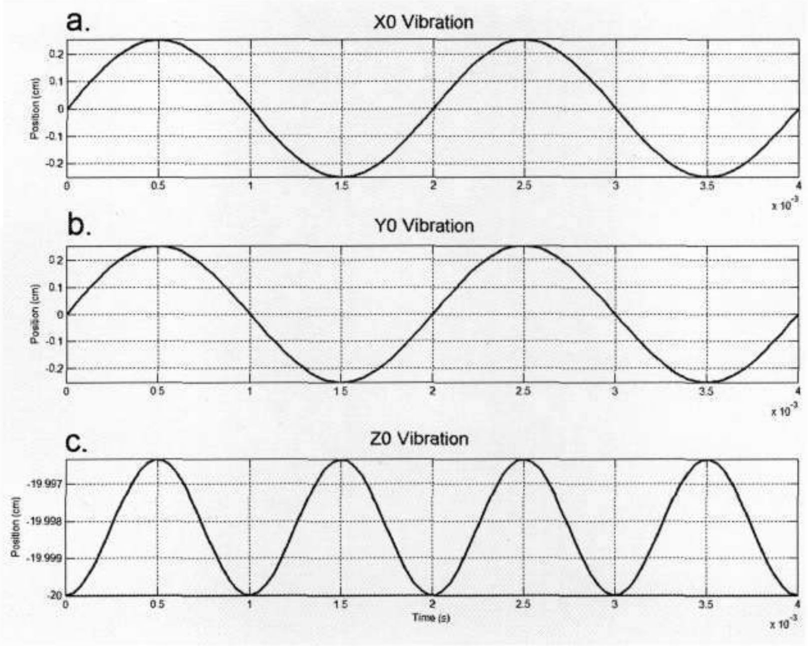


Figure 64 **End effector coordinate variation with upper leg oscillations of 1 degree and 500 Hz**

a.	x_0 Vibration	b.	y_0 Vibration
c.	z_0 Vibration		

8.2.3 Workspace envelope

The workspace envelope was calculated and its method was discussed in section 4.3.2 j. A graphical illustration is also displayed in said section.

8.2.4 Trajectory calculation for Solid Edge (SE)

In SE there is a motion simulator which can accept data for the positioning of mechanical elements to obtain a graphical mechanical simulation. The mechanical simulation was important to determine if the end effector would remain completely horizontal during all parts of its motion. An m-file was coded in MATLAB to obtain this positioning data for SE, i.e. time stamped angular values for the upper legs, which is discussed in section 7.1.4. When the positions of the upper legs are determined the kinematics solver in SE renders the other moving elements, which have dependencies on the upper legs, in their correct positions. This is due to structural relationships made during the assembly.

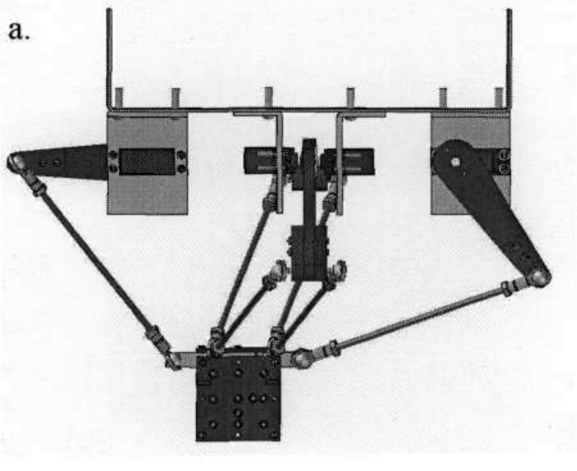
Figure 65 (a, b and c) shows that the end effector is completely horizontal at one position in the designed trajectory. The video simulation confirmed that the end effector did not twist or tilt on any of the X, Y or Z axes. This ensures that the end effector will always be parallel to the base frame and that the lasers will always be perpendicular to the sensor screens.

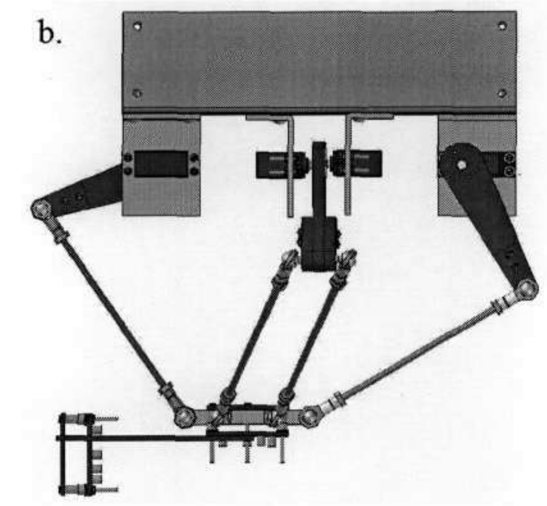
Figure 65 Mechanical simulation of PKM

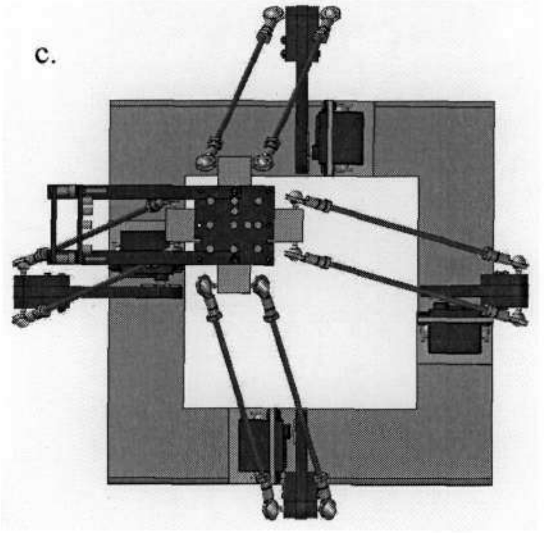
a. Front view

b. Side view

c. Bottom view







The XYZ coordinates for each point in the designed trajectory was plotted against time as well the solution to the inverse kinematics (upper leg angles) for the 23 second simulation, which are shown in Figures 66 and 67.

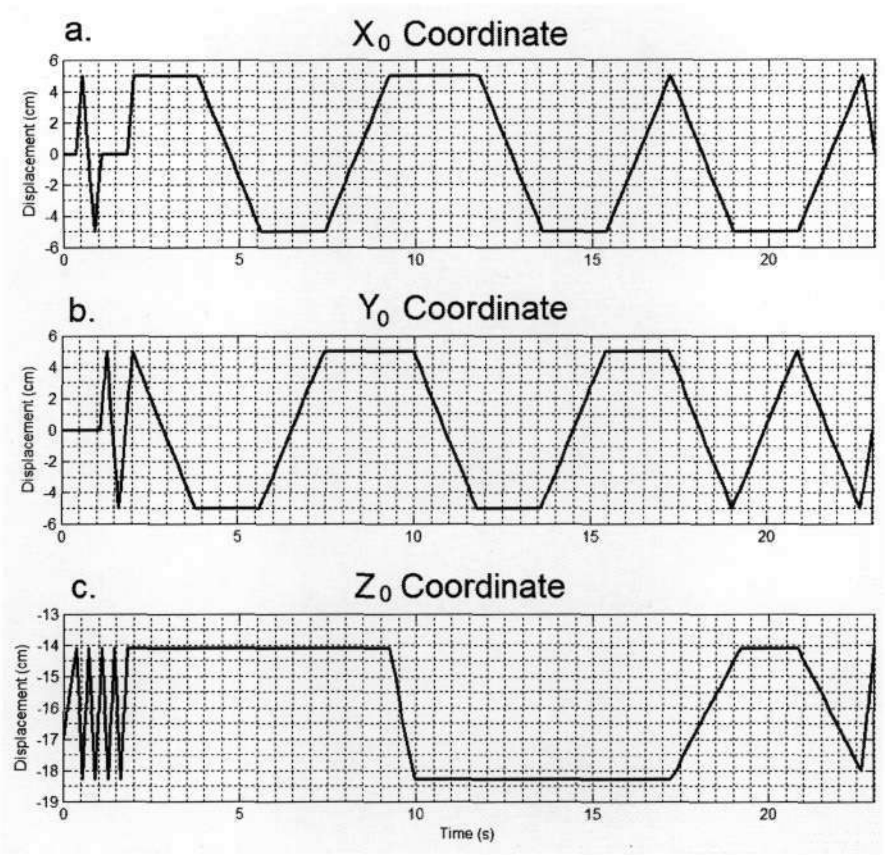


Figure 66 **Plot of the XYZ coordinates of the designed mechanical simulation trajectory**

a. x_0 Coordinate

b. y_0 Coordinate

c. z_0 Coordinate

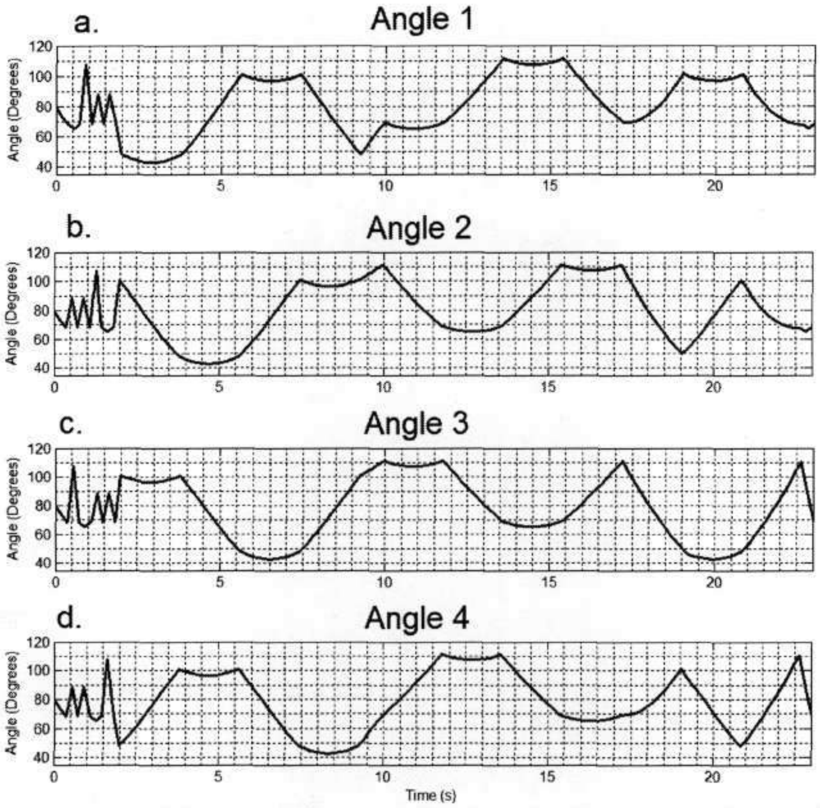


Figure 67 Upper leg rotation angles (0° - 180°) for the designed trajectory

a.	Angle 1	b.	Angle 2
c.	Angle 3	d.	Angle 4

8.2.5 Control System Simulation

The control system designed was intended to reduce the real world physical response of the motor leg combination of the PKM, which is approximately second order, to a first order system. This reason being to prevent any overshoot of the end effector the first time it reaches its intended position. The overshoot in position of the end effector can be seen at the leg with some oscillation of the upper leg. This can be measured at the feedback potentiometer of the servo motor (sections 5.4 and 6.2.1).

The controller was simulated in both the S (continuous) and Z (discrete time) domains and the block diagram is shown in Figure 68. The data is sent to the MATLAB workspace and plotted from there.

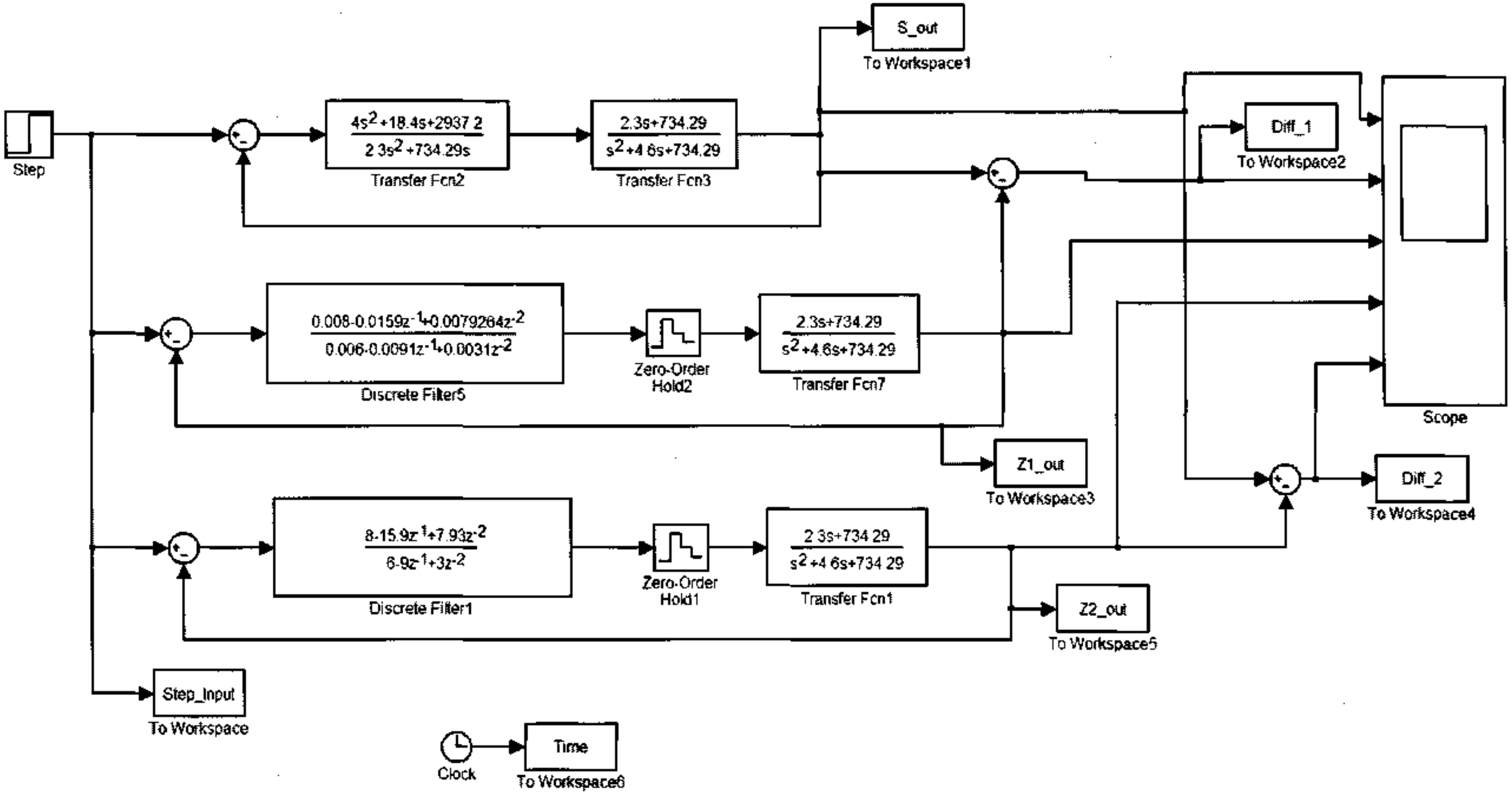
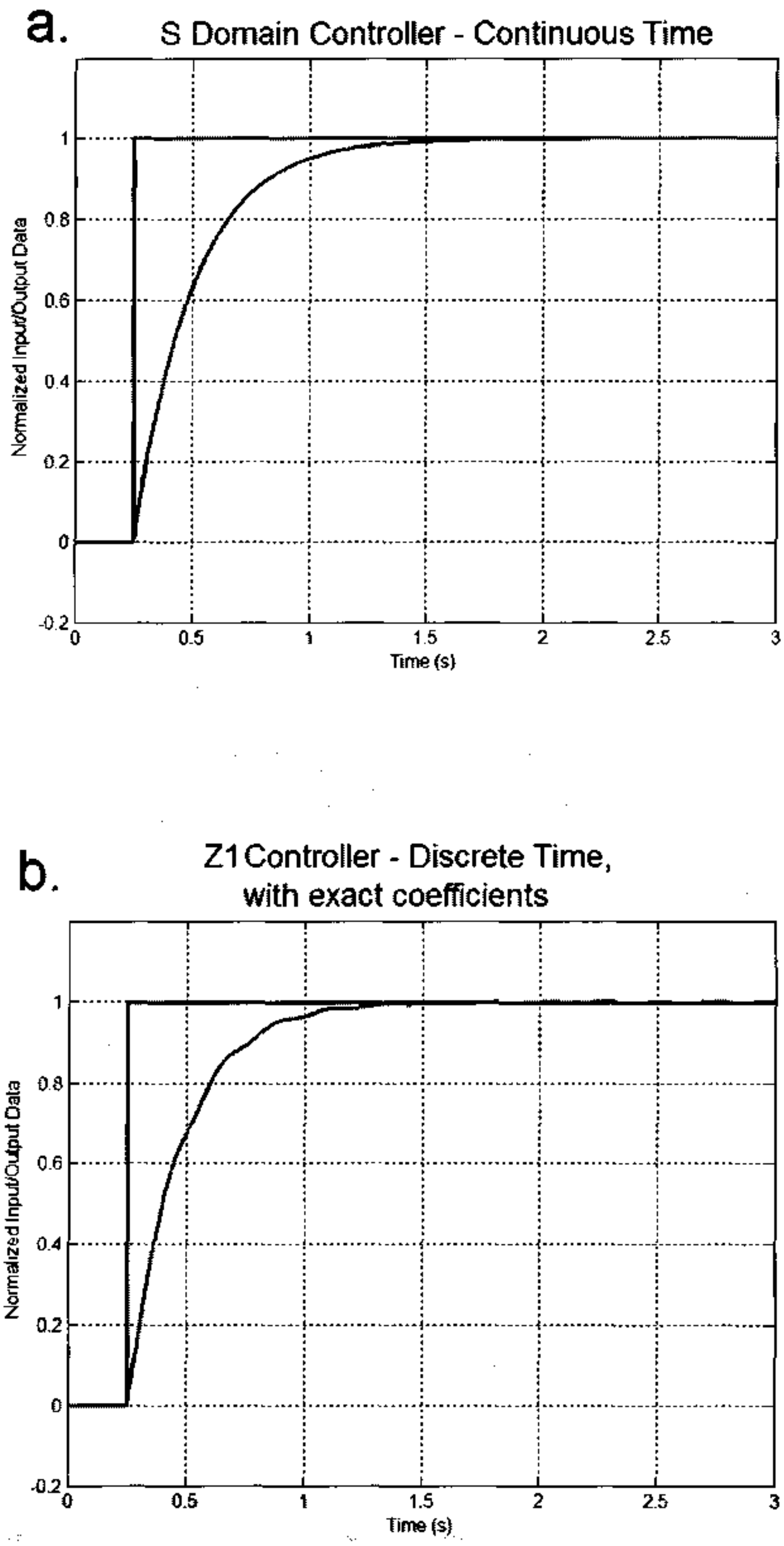


Figure 68 Block diagrams of controllers (1 continuous time and 2 discrete time)

The output of the continuous time (S domain) closed loop system is a first order response, this is seen in Figure 69 a. The output of the first closed loop discrete time system, i.e. with discrete time controller Z1, follows in Figure 69 b. Due to the discretizing nature of this controller it does not follow the S controller exactly and indentations can be seen on the output as it reaches steady state. The difference between the outputs of the systems with each controller at each instant is shown in Figure 69 c. The Z1 controller overcompensates; at each instant its value is slightly larger than that of the S controller. The coefficients of Z1 were multiplied by 1000 and some were rounded (to ease embedded calculation) to obtain controller Z2. One major problem with digital controllers is coefficient sensitivity. Changing the coefficients too much, to aid calculation, may make the controller unstable. The output of the closed loop system with controller Z2 illustrates this in Figure 69 d. The indentations are more distinct but the controller remains stable. The difference between the outputs of the systems with controllers Z2 and S is shown in Figure 69 e. Z2 is used in the embedded controller, even though it is not ideal so that the coefficients can be handled by the embedded system. The steady state time (time to reach steady state) is 1.25 s, and there is no severe oscillation that would be noticed. The digital controller was then simulated with the step conditions of the actual system, with initial value of 1.05 and final value of 1.375 at a step time of 4 s, as shown in Figure 46. The results are depicted in Figure 69 f.

Figure 69 Simulation results of controllers in SIMULINK

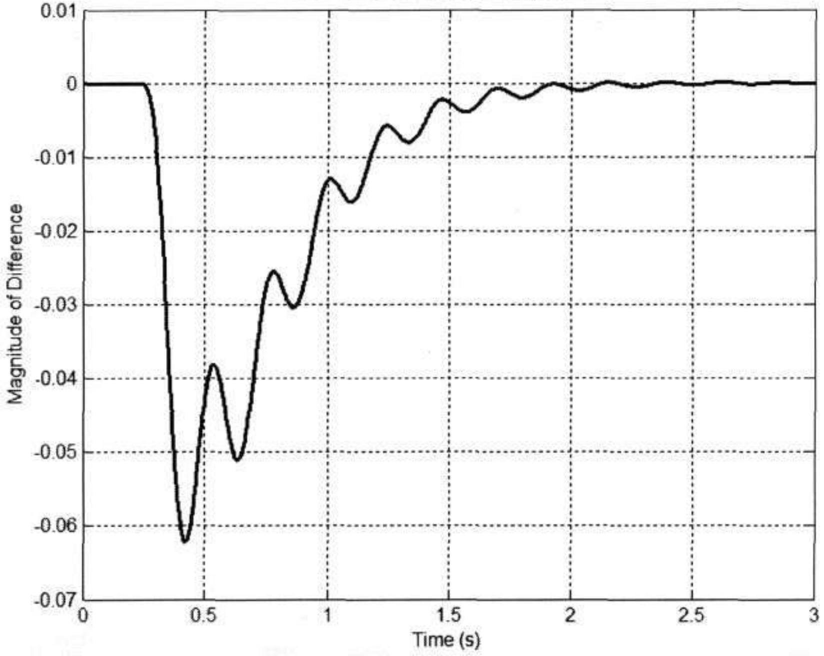
a. S Domain Controller – Continuous time
b. Z1 Controller – Discrete time, with exact coefficients
c. Difference between Z1 controller and S controller outputs
d. Z2 Controller – Discrete time, with rounded coefficients
e. Difference between Z2 controller and S controller outputs
f. Z2 Controller step response at conditions of measurement (section 6.2.1)



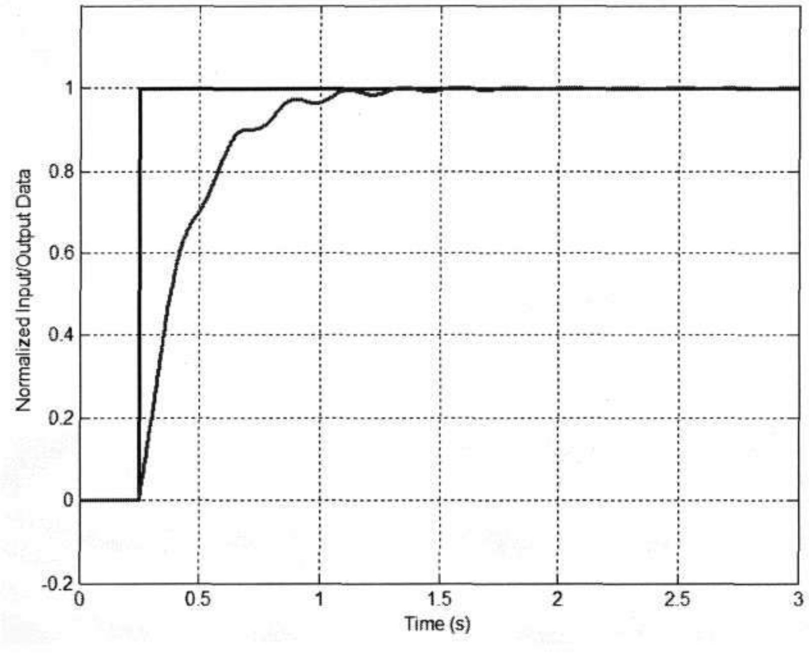
a. S Domain Controller - Continuous Time

b. Z1 Controller - Discrete Time, with exact coefficients

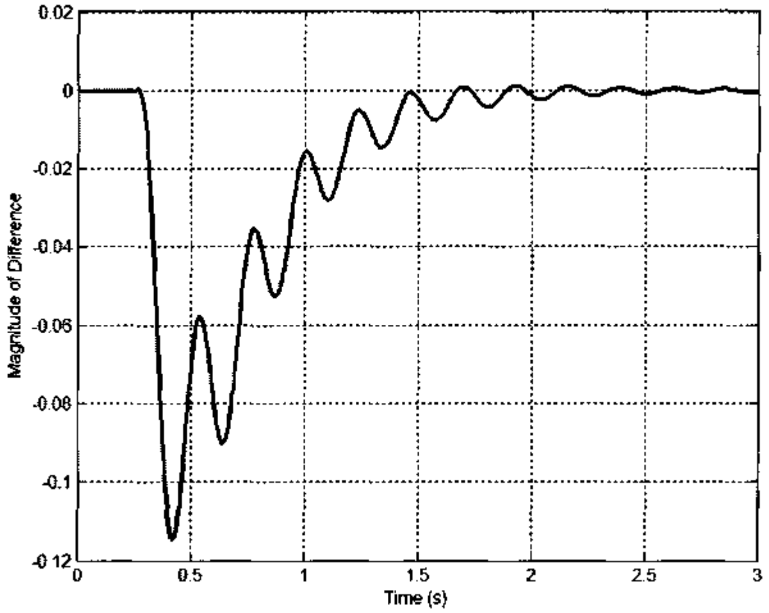
c. Difference between Z1-Controller and S-Controller outputs



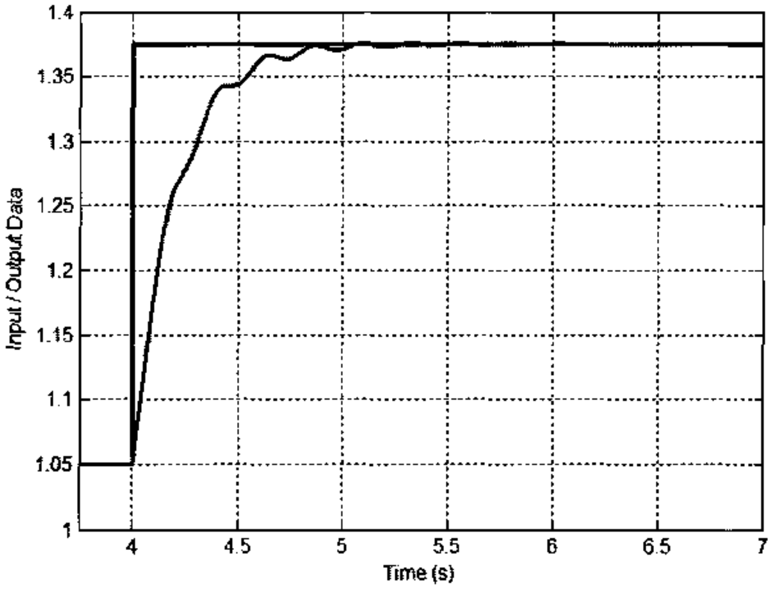
d. Z2 Controller - Discrete Time, with rounded coefficients



e. Difference between Z2-Controller (rounded coefficients) and S-Controller outputs



f. Z2 Domain Controller - At conditions of measurement (section 6.2.1)

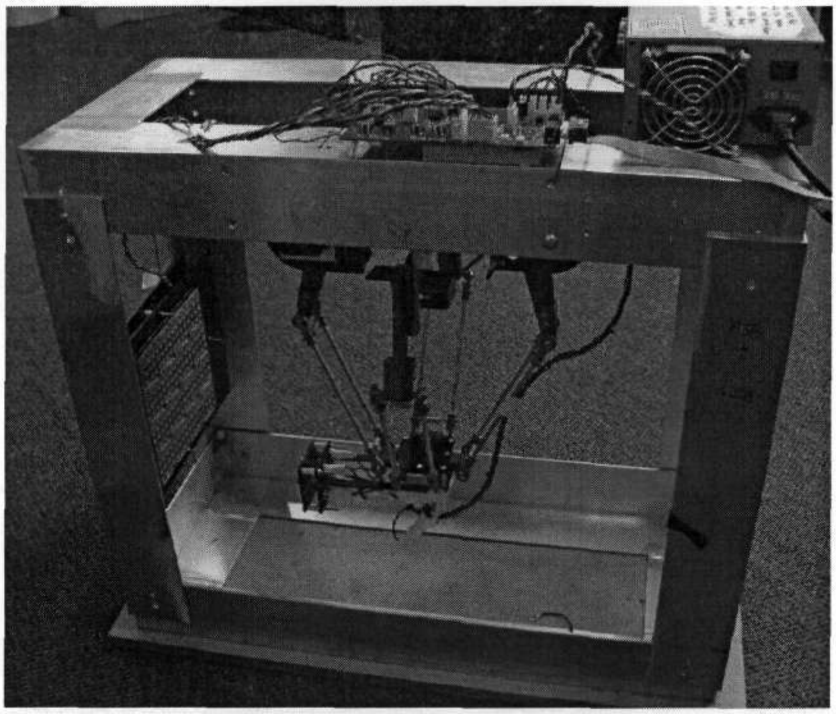


8.3 Prototype

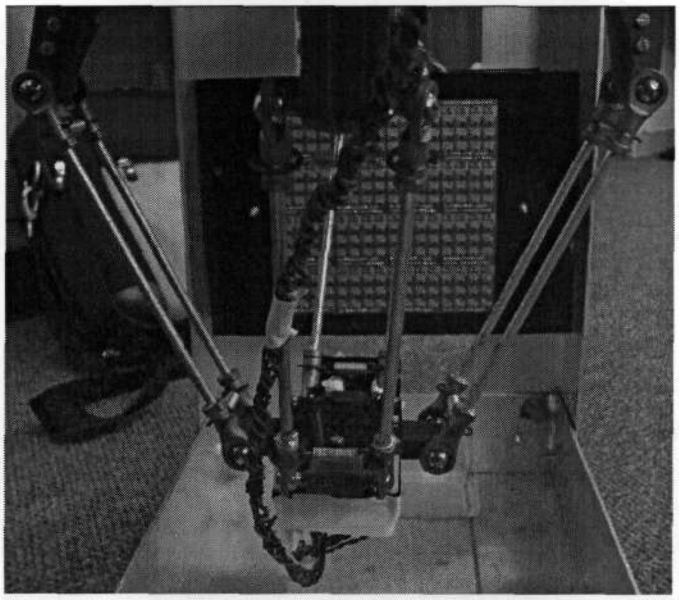
The mechanical structure was built without difficulty. A problem that was noticed was that the upper legs tend to loosen up at the servo after some time. The design could be improved with a bearing type mechanism providing reinforcement. When the PCBs were first designed the main controller board had some errors on it, these were corrected as shown in Figure 70 j, k with manual bypass wiring. The electronic designs in chapter 5 have these corrections taken into account. A few images of the actual PKM and electronic hardware are presented.

Figure 70 Illustrations of complete PKM and electronic hardware

a. Complete machine with controller and detector screens
b. Rear view of end effector and vertical detector electronic assembly
c. Horizontal detector electronic assembly and mechanical calibration tool
d. Top view showing controller and power supply
e. Controller board wired up to peripherals
f. Detector screen, front view
g. Detector screen, rear view
h. Signal routing board with components, front view
i. Signal routing board, rear view
j. Controller board with components, top view
k. Controller board, bottom view with corrections

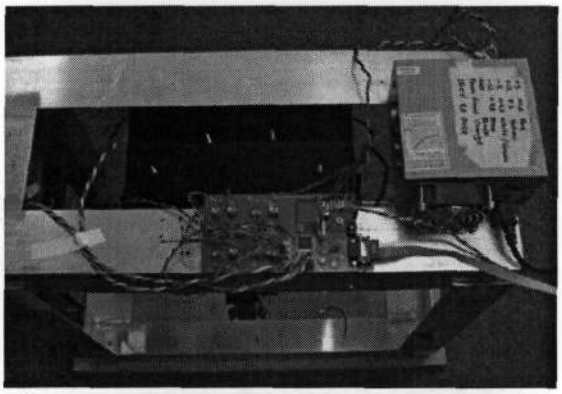


a.

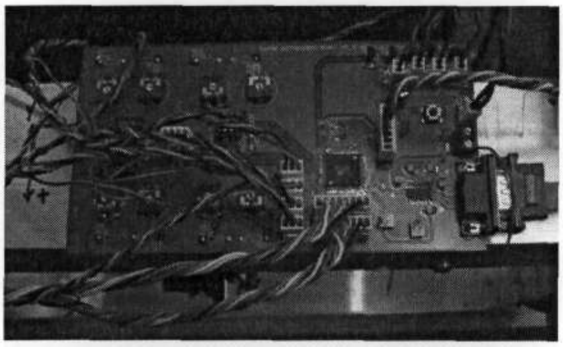


b.

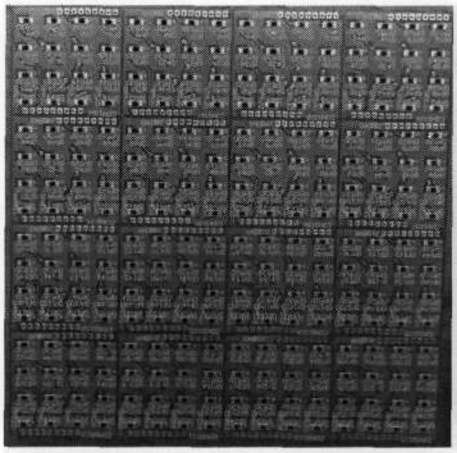
c.



d.

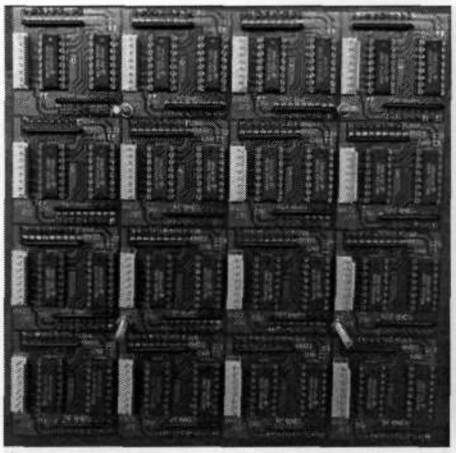


e.

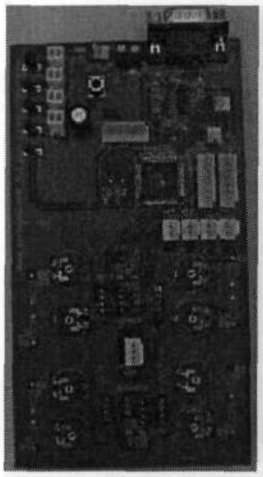


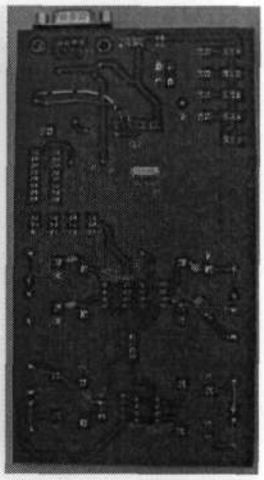
f.

g.


h.

i.


j.


k.

8.4 Performance Tests and Results

The performance tests carried out were to determine accuracy, precision, and repeatability of positioning as well as repeatability of sensor stimulation.

8.4.1 Step Response after Implementation of Controller

The step response of the system was tested after the controller was implemented to verify the controller design. The PC oscilloscope (Cleverscope) was used once again to obtain the data. The probe was attached to the shaft potentiometer on one of the servos, and the step was initialised as had been done to acquire the initial data.

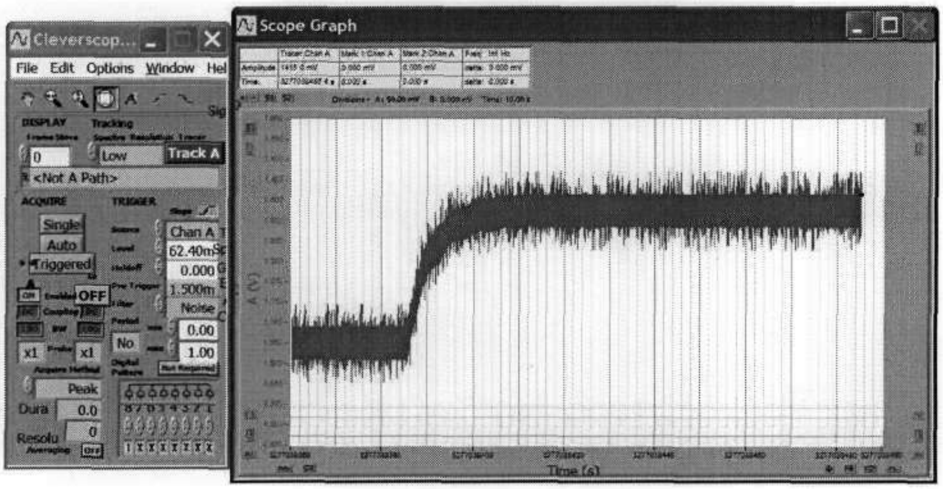


Figure 71 Step response after implementation of digital controller

Although the oscilloscope picks up noise on the probe, it is clear that the controller has reduced the response of the system to an approximate first order curve. The noise is due mainly to the potentiometer as its wiper moves across the windings.

8.4.2 Mechanical (Positioning)

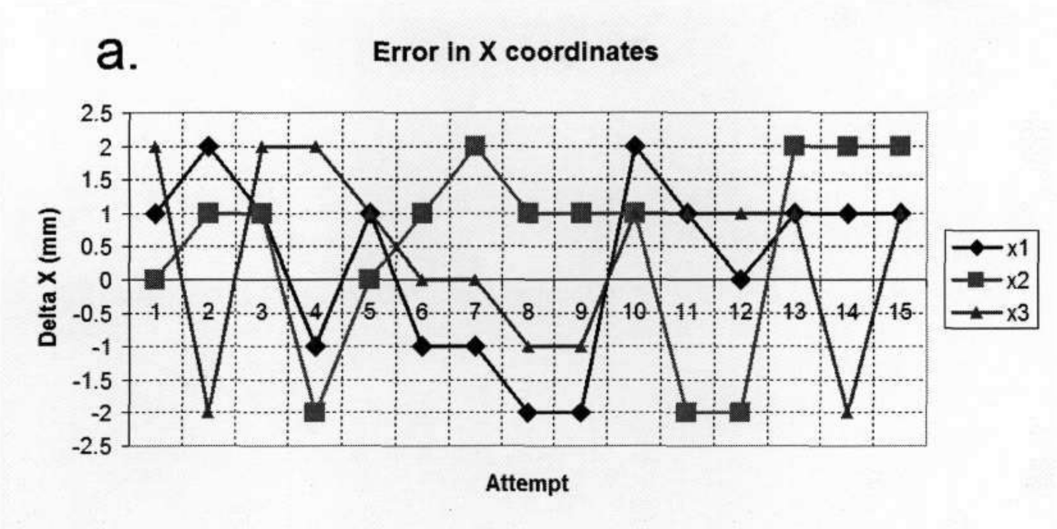
Accuracy is the degree of conformity of a measured or calculated quantity to its actual or true value. Accuracy is closely related to precision, the degree to which further measurements or calculations will show the same or similar results. Accuracy is defined as the maximum deviation from the theoretical, calculated or intended value, whereas precision is the maximum deviation from the mean value. The accuracy also gives an indication of repeatability which is the extent to which a similar result is attained. Due to the unavailability of a 3D metrology system, a method was devised to estimate the accuracy of the PKM. This method, although rudimentary, does provide some indication of the errors in positioning.

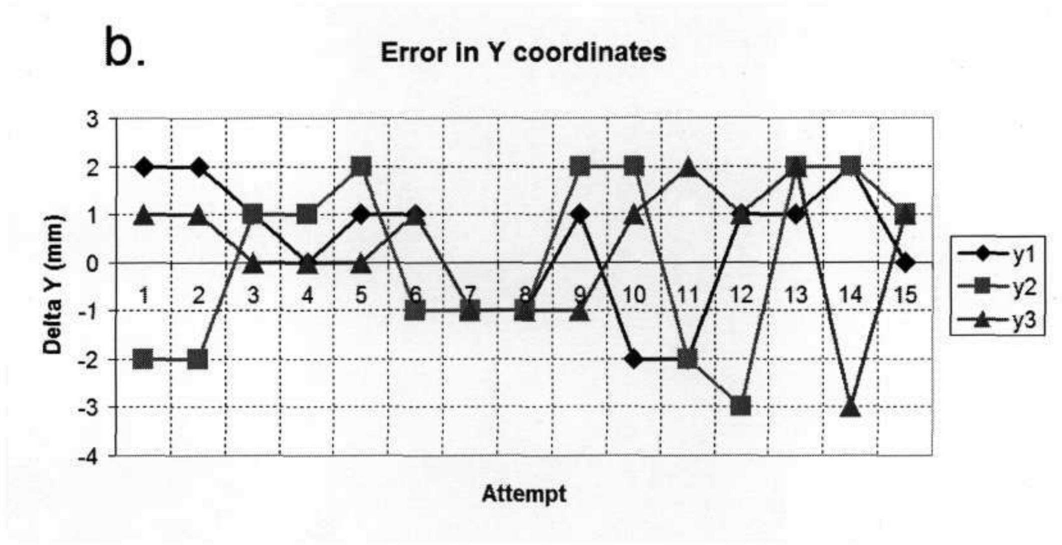
Method

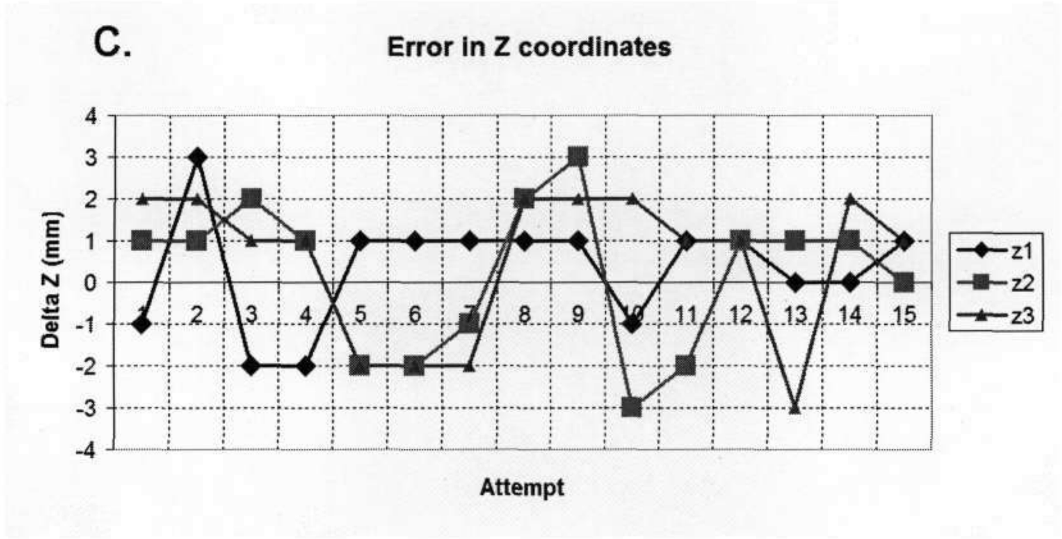
To get the accuracy or repeatability of the positioning systems, the electronic screens were removed and replaced with cardboard backings sporting a printout of the screens. These indicated exactly the positions of the sensors and facilitated tracking of the laser from above as well as the marking of positions and measuring of distances. The position offset was taken to be the distance from the centre of the laser point to the centre of the OP521 footprint. 15 Readings were taken for each of 3 coordinate sets (different X, Y and Z coordinates). Taking more measurements wouldn't offer much value based on the accuracy of the method. The measurements are made to the nearest mm. The coordinate sets chosen were $(0, 0, -20)$; $(-7, -7, -18)$ and $(5, -5, -16)$. The tables of results are located in Appendix D. The results are shown graphically in Figure 72 (a, b and c).

Figure 72 Positional accuracy results

a. Error in X Coordinates
b. Error in Y Coordinates
c. Error in Z Coordinates







The axial accuracy of the positioning is 2 mm, 3 mm and 3 mm for the X, Y and Z axes. This is the maximum measured deviation along the axis. The spatial accuracy is about 4 mm, this is the magnitude of the vector from the intended position to the actual position (occurs at data set 14, for coordinate set 3). The axial mean of the absolute values of the errors are 1.3 mm, 1.3 mm and 1.5 mm (X, Y and Z respectively). The spatial mean is therefore 2.4 mm. From the definition the precision (beginning of this section) is 1.6 mm (4 – 2.4). These values were obtained from mathematical functions. As the smallest measurement discernable has mm resolution these values are applied to a ceiling function yielding 2mm for each of the axial mean errors, 3 mm for the spatial mean and 2 mm for the precision.

The major factors involved in the positioning inaccuracy are due to hysteresis as well as backlash in the servo gearing system, and the ball socket joints.

Hysteresis is the difference in reading or positioning when the physical quantity being investigated is approached from different directions. It is due to mechanical friction, elastic deformation and thermal effects. It is a property of systems (usually physical systems) that do not instantly follow the forces applied to them, but react slowly, or do not return completely to their original state.

Backlash is the play or loose motion in an instrument due to the clearance existing between mechanically contacting parts. In gearing systems, it is the clearance between two gears, the amount by which the width of a tooth space exceeds the thickness of the engaging tooth on the pitch circles. It also occurs in lead screws, and is the amount of free movement between a screw and nut. Backlash cannot be eliminated completely as it is required to allow for lubrication, manufacturing errors, deflection under load and differential expansion between the gears and the housing.

Other errors due to inaccurate machining of parts and their placement also factor into the problem, but they are not nearly as significant as those from hysteresis and backlash.

Most manufacturers of geared motors do provide some indication of these errors in their datasheets. The hysteresis and backlash due to the ball and socket joints which were actually modified from ball in socket bearings are not quantifiable. In general this problem exists for all delta type mechanisms that use ball in socket joints. The problem can be alleviated with the use of compliant joints which was mentioned in section 2.2. Compliant joints are difficult to manufacture and are expensive. It was not possible to make them on this scale.

8.4.3 Electronic (Sensing) Repeatability

The repeatability measurement for the detector screen was done within the positioning capabilities of the rig. The PKM was commanded to sensor coordinate positions at Z displacements of 14 cm, 17 cm and 20 cm, and 25 attempts were made for each of 16 sensors. The repeatability for each distance is taken as the average of all the readings in the group measurement, and expressed as a percentage of the 25 attempts. They are 99.25%, 98.75% and 98% respectively. The repeatability decreases mainly due to the fact that the further away the laser, the more difficult it is to stabilize and hold a position. A stiffer machine with vibration damping should have no such problem.

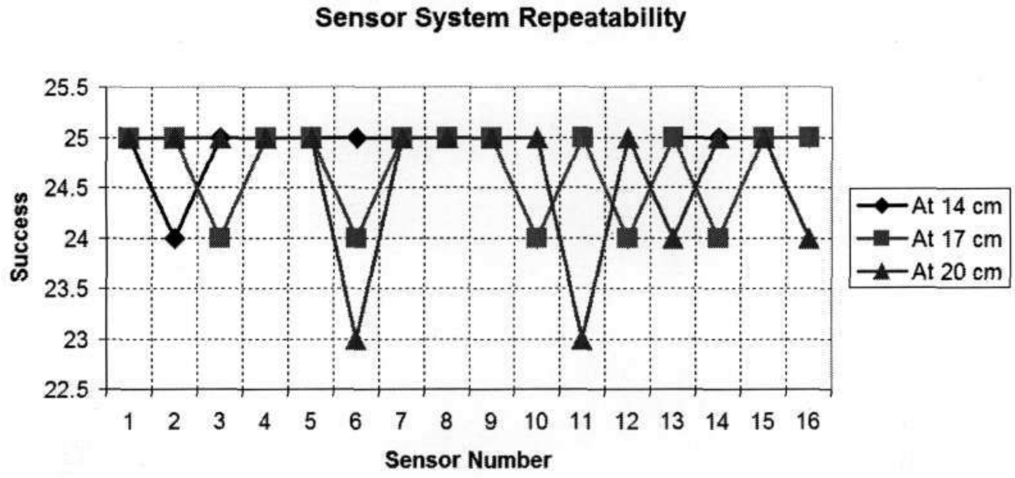


Figure 73 Electronic sensor system repeatability

8.5 Chapter Summary

This chapter starts with a description of the procedure and tools used to calibrate the PKM in both software and hardware. This ensures that the system is aligned with the coordinate system in the control software. It then presents all the simulation data. The forward and inverse kinematics functions were tested against and validated each other. The workspace was visualised and some vibration effects from the motors on the end effector illustrated. The trajectory designed was simulated in MATLAB and the data imported to SE. SE verified that the end effector would remain perfectly horizontal throughout its trajectory (which consisted of all motions the machine would have to make). A video was then created. The controller was designed and tested in SIMULINK. The output of the S controller is first order as this was designed. The Z controller's implementation for both exact and rounded coefficients were tested against the S controller. Both discrete time controllers produced less than perfect responses but were acceptable for this scaled model. Pictures of the real machine were then shown including all PCBs. The controller with rounded coefficients was implemented and the step response of the system measured. It was approximately first order and acceptable. The positioning accuracy of the mechanical system was then tested with a rudimentary method due to the lack of a 3D metrology system. This revealed a 3 mm axial accuracy and a 4 mm spatial accuracy. The repeatability of the sensor system was then tested and the results were within specification.

9 Conclusion

The objectives of this project were met:

- Various types of parallel kinematics machines were researched in particular the Delta type (Flex-Picker) PKM.
- A scaled and modified version of the Delta robot was designed, simulated and built.
- Various sensor systems were researched. A sensor system was then designed and implemented. It was developed in 2D and extrapolated to 3D. The system was then calibrated and tested.
- A control system was designed which interprets data from the sensor system, and a motor controller was designed. Algorithms were developed to control movement, and acquire data from sensors.
- A performance analysis was conducted for both the mechanical and the sensor systems.

The main purpose of this project was to design a parallel robot structure possessing an integrated end effector sensor system for use in the agricultural, or in general food processing, industry. For this application the resolution on position and sensing should be in the millimetre range. Robustness and reliability or repeatability of measurements and positioning are crucial.

A study was undertaken to decide on the PKM to model, modify and simulate. The investigation revealed that one of the most popular PKMs used in industry was the Flex-Picker by ABB automation. It was based on the design by Dr. Raymond Clavel. There was however no detailed, easily understandable literature on aspects of the kinematics of the machine. A major portion of this thesis is therefore dedicated to finding simplistic closed form solutions to both the forward and inverse kinematics of the machine that could be implemented on stand alone processors.

The mechanical structure of the machine designed was based on a Flex-Picker robot. It was scaled and modified to incorporate an additional arm. The purpose of that arm was to increase payload carrying ability, machine stiffness and aid the robot in exiting singular positions caused by the other three. A design specification was given in section 1.5.1. The mechanical design of this parallel robot was split into the two processes of structural design and dimensional synthesis. This robot was symmetric and its topology classified as 3-DOF 4 RSS (see section 4.1.1. a Machine Topology). Spherical joints were used for the "knee" and "ankle" joints as these give it more freedom than the universal joints used in the classical Delta mechanism. The most difficult problem was obtaining the ball and cup for each joint. These were eventually made from ball in socket bearings. Removing the ball from the socket and having the socket maintain a cup shape without scratching or damaging the ball was difficult and required dexterous workmanship. This was seen to by the mechanical design workshop.

The link lengths and joint locations were chosen to more than adequately satisfy the workspace requirements of the specification. This was not obtained from an exact mathematical formula or software simulation, but from an understanding of the machine and some rough sketches. These parameters were however tested later in software to determine if they satisfied all the specifications.

Both the forward and inverse kinematics were solved using a combination of coordinate geometry and algebra. Non-geometrical model parameters (section 4.1.2) were not considered. There is great difficulty in solving the forward kinematics for any PKM and closed form solutions were created for the PKM designed. There are multiple solutions to both the forward and inverse kinematics and geometrical relationships were devised to select the correct solution set. These also provided a means to determine a singular configuration, i.e. when the arms are completely extended or folded. These singularities are then avoided through the method discussed in section 4.3.2 g and i.

A dynamics model was created to accomplish mass and force modelling. Since this scaled model does not accomplish any pick and place operations, a theoretical analysis of the dynamics is not necessary. The dynamics of the system is considered in the control system, where the transfer function of the leg is obtained from LTI theory of plant estimation. That transfer function inherently contains dynamic information of the system.

These kinematics solutions were then coded in MATLAB. The forward kinematics function was used to verify the inverse kinematics and vice versa. These functions were then used to create a vibration model and visualisation of the workspace boundary. The vibration model revealed an interesting fact: since the legs are synchronised, any vibration on the motors results in the end effector having an oscillatory diagonal motion. The Z displacements are negligible in magnitude but have a frequency twice that of the leg vibration.

The mechanical design was created in SE, which was also used to generate a graphical 3D simulation of the machine. This required 4 sets of time stamped data indicating angular positions of the upper legs. SE resolves the other components in their correct position for each configuration of the upper legs based on assembly relationships made during the design. A graphical simulation was crucial for one other reason, it had to be verified that the end effector would remain parallel to the base frame during its motion. This was an important requirement of the sensor system design. A trajectory was then designed and the kinematics solutions implemented to obtain the angular configurations at each point along this trajectory. The angles were time stamped and saved each to its own text file. These text files were then imported to SE for each of the 4 actuated axes. A 3D video animation of the simulation was created, which confirmed that the end effector would remain parallel to the base frame during its motion.

To design the sensor screen an investigation was made into numerous technologies that are available to locate objects in space. These were varied and used different media, transducers or processing techniques. The problems faced are with resolution, and that different types of stimulating media present difficulties with respect to their use. After consideration of all the available physical stimuli used with these technologies, it was decided that a laser light stimulant would be the most suitable for the application at hand. A surface mount phototransistor the OP521 was selected as the detector.

A sensor system was designed to reduce errors encountered in the so called 'open loop' control of this robotic mechanism, and as such the sensors are arranged in a planar grid with a resolution of 10 mm and by itself functions in 2D space. A 3D system was created by attaching two of these 2D sensor planes at right angles. Vertical and horizontal laser grids on the end effector improve the resolution by a factor of 4, providing a final resolution of 2.5 mm. Each laser grid has 12 lasers which are strategically positioned.

The detector screen's nature was inherently modular thus allowing it to integrate with existing techniques for motion control without difficulty. The screens also have an advantage over existing systems in their modularity. Detector screens of any practical size may be built, without loss in resolution. The only issue may be wiring of the modules but this could be overcome with additional PCB routing connectors. This preservation of resolution and sensor error does not apply to many existing systems when they are scaled up. The laser and its detectors also make the system robust and immune to environmental errors.

The electronic designs, schematics and PCBs, were created in PROTEL 99SE. The only issue was of space, i.e. fitting all the components on the PCBs and soldering on both sides of the board. During testing the system functioned as expected. It was also noticed at this point that end effector vibration would be an issue, however for this machine there was none. For this structure vibration effects may be averaged by changing the phase of vibration for each leg.

In order to design a control system, a "plant" model had to be obtained for each leg. This model inherently contains dynamic information. A number of measurements were made, with different step inputs, to confirm that the system was LTI before proceeding. Standard techniques were used to obtain this model.

- Apply a step input
- Measure the output
- Fit an approximate mathematical function to the data
- Take the Laplace transform of the output function and divide it by the Laplace transform of the input function

The output data was measured with a PC oscilloscope with a tap directly into the servo's potentiometer. The data was then exported to MATLAB where a Laplace transformable function was generated to represent the plant. Once the plant function was obtained, controllers were designed both in the S domain and the Z domain. Simulations were carried out and the controllers were compared to each other. The digital controller was acceptable in its performance and was then coded in a 'C' based program.

The software was composed of the embedded system microcontroller code and the PC user interface. The embedded system has to receive and interpret commands from the PC, acquire data from sensors, process that data and indicate if there are stimulated sensors, and lastly generate PWM signals for servo rotation. A single microprocessor, the ATMEL AVR ATmega128, was used to accomplish this, with a clock frequency of 16 MHz. Custom data transfer routines were written to increase the speed of code execution and decrease the SRAM memory used in the microcontroller. These routines also facilitate easier processing. The PC control software had to transmit commands, receive and display data, and perform high level control.

Performance tests were carried out to determine accuracy, precision and repeatability of positioning as well as repeatability of sensor stimulation. A method was devised to carry out these tests which offered a good indication of the positioning capability of the Flex-Picker as well as the reliability of the detector screen. These were within the specifications mentioned.

In conclusion the project was completed as:

- A modified Flex-Picker was researched, designed, simulated and assembled;
- A sensor system was researched, designed and tested;
- Its control system was designed and implemented;
- Design specifications were met.

There were 1 journal article and 5 conference papers were written and accepted for publication at various stages of project development.

References

[1] "Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering", by W. Bolton; third edition, 2003. Pearson Education Limited, England.

[2] "Mechatronics: Principles and Applications", by G. C. Onwubolu; first edition, 2005. Elsevier Butterworth-Heinemann, England.

[3] "The True Origins of Parallel Robots" By Ilian Bonev, an article for Parallemic, January 2003. http://www.parallemic.org/

[4] "Robot Analysis: The mechanics of Serial and Parallel Manipulators", by L. W. Tsai; first edition, 1999. John Wiley & Sons, New York.

[5] "Parallel Robots", by J. P. Merlet; first edition, 2000. Kluwer Academic Publishers, Amsterdam.

[6] "Parallel Kinematics Robots" by Bennett Brumson, an article from http://www.roboticsonline.com/

[7] IRB 340 – Flex-Picker Datasheet from www.abb.com/robots

[8] "Conception D'un Robot Parallele Rapide A 4 Degres De Liberte" by Reymond Clavel, Ingenieur mecanicien diplome, Departement De Microtechnique, Ecole Polytechnique Federale De Lausanne (EPFL), Ph.D Thesis, Lausanne, 1991.

[9] "Modelling And Model Based Performance Prediction For Parallel Kinematic Manipulators" by Jan-Gunnar Persson, Kjell Andersson; Engineering Design, Department of Machine Design; KTH – Royal Institute of Technology, Stockholm, Sweden. Presented at Mechatronics Meeting, Gothenburg, August 2003.

[10] "Mechatronic design of a parallel robot for high-speed, impedance-controlled manipulation" by L. E. Bruzzone, R. M. Molfino, M. Zoppi; Proceedings of the 11th Mediterranean Conference on Control and Automation, Rhodes, Greece, June 2003. http://www.dimec.unige.it/PMAR/

[11] "Design of the 'Granit' Parallel Kinematic Manipulator" by Alessandro Tasora[1], Paolo Righettini[2], Steven Chatterton[2]. 1 – Università degli Studi di Parma, Dipartimento di Ingegneria Industriale, Parma, Italy; 2 – Politecnico di Milano, Italy. Proceedings of RAAD'05 – 14th International Workshop on Robotics, Bucharest, May 2005.

[12] "A parallel robot for the Strain Imager (SALSA)" by S. Rowe (ILL), Millennium Programme and Technical Developments. http://www.ill.fr/AR-02/site/areport/fset_96.htm

[13] "Design of Compliant Parallel Kinematics Machines" by Yong-Mo Moon, Prof. Sridhar Kota, Mechanical Engineering, University of Michigan. Proceedings of DETC'2002 – Biannual Mechanisms and Robotics Conference, DETC'2002/MECH-34204, Montreal, Canada, September-October 2002.

[14] "Kinematics Design of In-Parallel Robots," by G. Yang, I. M. Chen, W. K. Lim and S. H. Yeo, School of Mechanical and Production Engineering, Nanyang

Technological University, Singapore. Journal of Autonomous Robots, Vol. 10, No. 1, p83-89, 2001.

[15] "Multi-Criteria Optimal Design of Parallel Manipulators Based on Interval Analysis" by F. Hao, J.P. Merlet; INRIA Sophia-Antipolis, France, 6 July 2004. Journal of Mechanism and Machine Theory, Vol. 40, No. 2, p157-171, February 2005.

[16] "Two Novel Parallel Mechanisms with Less than Six DOFs and the Applications" by Xin-Jun Liu[1], Jongwon Kim[1], Jinsong Wang[2]; 1 – Robust Design Engineering Lab, Seoul National University, Seoul, Republic of Korea; 2 – Manufacturing Engineering Institute, Tsinghua University, Beijing, China. Proceedings of the workshop on Fundamental Issues and Future Research Directions for Parallel Mechanisms and Manipulators, Vol. 1, No. 1, p172-177, Quebec, Canada, October 2002.

[17] "The GPS Manual: Principles & Applications", by S. Dye and F. Baylin; first edition 1997. Baylin-Gale Productions.

[18] "Understanding the GPS: An Introduction to the Global Positioning System", by G. T. French; first edition, 1997.

[19] "Location Sensing Technologies and Applications", by George Roussos. School of Computer Science and Information Systems, Birkbeck College, University of London, November 2002.

[20] "High Precision GPS Guidance of Mobile Robots" by R. Willgoss, V. Rosenfeld and J. Billingsley. ACRA – Australasian Conference on Robotics and Automation, August 2003.

[21] "Indoor GPS Technology" by Frank van Diggelen and Charles Abraham, Global Locate, Inc. Presented at CTIA Wireless-Agenda, Dallas, USA, May 2001.

[22] "Indoor Positioning Systems in Healthcare, a Basic Overview of Technologies" by M. Dempsey; Radianse Inc. CIMIT – Innovative Technology for Medicine, June 2003.

[23] "In-building location using Bluetooth" by Miguel Rodriguez, Juan P. Pece, Carlos J. Escudero. Departamento de Electronica e Sistemas, Universidade da Coruna, Spain. INWAN – International Workshop on Wireless Ad-Hoc Networks, London, UK, May 2005.

[24] "Indoor and Outdoor Positioning in Mobile Environments – A Review and some Investigations on Wlan-Positioning" by R. Bill, C. Cap, M. Kofahl and T. Mundt. University Rostock, Germany. International Workshop on Ubiquitous Geographical Information Sciences, Vol. 10, No. 2, p91 -98, Gavle, Sweden, June 2004.

[25] "Location Awareness in Ad Hoc Wireless Mobile Networks", by Y. Tseng, S. Wu, W. Liao and C. Chao. IEEE Computer, Vol. 34, No. 6, p.46-52, June 2001.

[26] "Location Sensing Techniques" by Jeffrey Hightower and Gaetano Borriello, Computer Science and Engineering, University of Washington. Technical report, UNCSE 01-07-01, July 2001. http://seattle.intel-research.net

[27] "On the Potential Use of Mobile Positioning Technologies in Indoor Environments", by A. Pateli, G. M. Giaglis, K. Fouskas, P. Kourouthanassis and A. Tsamakos;

Proceedings of 15th Bled Electronic Commerce Conference -e-Reality: Constructing the e-Economy; Bled, Slovenia, April 2002.

[28] "A Taxonomy of Indoor and Outdoor Positioning Techniques for Mobile Location Services", by G.M. Giaglis, V. Zeimpekis, and G. Lekakos. ACM SIGECOM Exchanges, Vol. 3, No. 4, p19-27, August 2003.

[29] "Displacement Measuring Interferometers Provide Precise Metrology", by D. Musinski. Laser Focus World, Vol. 39, No. 12, December 2003.

[30] "A new sensor for the micro-metre-level measurement of three-dimensional, dynamic contours," by Tony Schmitz and John Ziegert; Machine Tool Research Center, Department of Mechanical Engineering, University of Florida, Gainesville, USA. Measurement Science and Technology, IOP – Institute of Physics – Electronic Journals, Vol. 10, p51-62, Feb. 1999. http://www.IOP.org/EJ

[31] "Active Vision/Display Sensors for Precision Positioning" by J. Ziegert. University of Florida, USA. A project proposal to the NSF – National Science Foundation, September 2005.

[32] "Image Classifiers for Scene Analysis", by B. L. Saux and G. Amato. Computational Imaging and Vision, Vol. 32, p39-44, March 2006.

[33] "Visual Schemas in Object Recognition and Scene Analysis", by R. Miikkulainen and W. K. Leow. The handbook of brain theory and neural networks, p1029-1031, July 1998.

[34] "Simultaneous Map Building and Localization for an Autonomous Mobile Robot", by J. J. Leonard and H. F. Durrant-Whyte. Proceedings of the IEEE International Workshop on Intelligent Robots and Systems, p1442-1447, Osaka, Japan, November 1991.

[35] "Mobile Robot Localisation and Mapping in Extensive Outdoor Environments" by I. Bailey. PhD thesis, ACFR, University of Sydney, Australia, August 2002.

[36] "Design and Accuracy Evaluation of High-Speed and High Precision Parallel Mechanisms" by Yoshihiko Koseki[1], Tatsuo Arai[2], Kouichi Sugimoto[3], Toshiyuki Takatuji[4], Mitsuo Goto[4]; 1 – Mechanical Engineering Laboratory, AIST; Namiki1-2, Tsukuba, Ibaraki, Japan; 2 – Graduate School of Engineering Science, Osaka University, Japan; 3 – Hitachi ltd., Japan; 4 – National Research Laboratory of Metrology, AIST, Japan. Proceedings of ICRA'98, Vol. 3, No. 1, p1340-1345, Leuven, Belgium, May 1998.

[37] "Optimal Design of Robots" by J. P. Merlet, INRIA Sophia Antipolis, France. Proceeding of Robotics: Science and Systems, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 2005. http://www.roboticsproceedings.org/index.html

[38] "Manipulability and Static Stability of Parallel Manipulators" by A. Muller, Institute of Mechatronics, Chemnitz University of Technology, Germany. Journal of Multi-Body System Dynamics, Vol. 19, No. 1, p1-23, February 2003.

[39] "Kinematic Design of a Six-DOF Parallel-Kinematics Machine With Decoupled-Motion Architecture," by Guilin Yang, I-Ming Chen, Weihai Chen, and Wei Lin, IEEE Transactions on Robotics and Automation, Vol. 20, No. 5, p876-884, 2004.

[40] "Kinematic and Dynamic Properties of Parallel Manipulators," by A. Muller and P. Maiber; Institute of Mechatronics, Chemnitz University of Technology, Germany. Journal of Multi-Body System Dynamics, Vol. 5, No. 3, p223-249, April 2001.

[41] "On the Kinematic Analysis of Robotic Mechanisms," by James Nielsen and Bernard Roth; Design Division, Department of Mechanical Engineering, Stanford University, USA. International Journal of Robotics Research, Vol. 18, No. 12, p1147-1160.

[42] "Parallel robots" by Herman Bruyninckx, August 2005. The Robotics WeBook, the online textbook project. http://www.roble.info/robotics/parallel

[43] "Position Analysis of a Class of Translational Parallel Mechanisms" by M. Zoppi, L. E. Bruzzone and R.M. Molfino; PMAR Robot Design Research Group - DIMEC - University of Genova, Genova, Italy. International Journal of Robotics and Automation, Vol. 19, No. 3, 2004.

[44] "Geometric Analysis Of Parallel Mechanisms," by Ilian Alexandrov Bonev, D´epartement de g´enie m´ecanique, Facult´E Des Sciences Et De G´Enie, Universit´E Laval, Qu´ebec, November 2002.

[45] "Descriptive Geometric Kinematic Analysis of Clavel's 'Delta' Robot," by P. J. Zsombor-Murray, Department of Mechanical Engineering, Centre for Intelligent Machines, McGill University, Canada, April 2004.

[46] "A New Approach to the Design of a DELTA Robot with a Desired Workspace," by [1 & 2] Xin-Jun Liu, [1] Jinsong Wang, [2] Kun-Ku Oh and [2] Jongwon Kim; [1] Manufacturing Engineering Institute, Department of Precision Instruments and Mechanology, Tsinghua University, Beijing, China; [2] Robust Design Engineering Lab, School of Mechanical and Aerospace Engineering, Seoul National University, Korea. Journal of Intelligent and Robotic Systems, Vol. 39, No. 2, p209-225, February 2004.

[47] "Singularities of Robot Manipulators," by Peter Donelan; School of Mathematics, Statistics and Computer Science; Victoria University of Wellington, New Zealand. Proceedings of "5 Weeks in Singularities," Luming, 2005.

[48] "Singularities of Parallel Manipulators: A Geometric Treatment," by Guanfeng Liu, Yunjiang Lou, and Zexiang Li. IEEE Transactions of Robotics and Automation, Vol. 19, No. 4, p579-594, 2003.

[49] ATmega128 Microcontroller Datasheet, from www.atmel.com.

[50] "+5V-Powered, Multi-Channel RS-232, Drivers/Receivers" by MAXIM. www.maxim-ic.com/packages.

[51] "The Robot Builder's Bonanza", by G. McComb; second edition 2001. McGraw-Hill.

[52] "Build Your Own Humanoid Robots", by K. Williams; first edition 2004. Tab Books.

[53] TLC548CP – Analogue to digital converters, from Texas instruments. www.ti.com

[54] OP521 Phototransistor datasheet, from www.alldatasheet.com.

[55] "Choosing the Detector for your Unique Light Sensing Application," by Larry Godfrey. http://www.engr.udayton.edu/faculty/jloomis/ece445/topics/egginc/tp4.html.

[56] "Sensors and Transducers", by I. R. Sinclair; third edition 2001. Newnes.

[57] "Sensor Technology Handbook", by J. Wilson; first edition 2004. Newnes/Elsevier.

[58] "Intelligent Sensor Technology", by Y. Ohba and Y. Chba; first edition 1992. John Wiley & Sons.

[59] "Handbook of Through the Air Communications", by D. A. Johnson. Online book, http://www.imagineeringezine.com/ttaoc-pdf/OTTAC-Handbook.PDF

[60] "Design and Implementation of a Laser Level Detector" by G. Cook; School of Information Technology and Electrical Engineering, The University of Queensland, honours thesis, October 2002.

[61] "Light Sources and Detectors", by M. Jonasz; chapter 91 of "Handbook of Measuring System Design", editors: P. H. Sydenham and R. Thorn; 2007, John Wiley & Sons.

[62] 74LS166 Parallel to serial shift register datasheet, from www.alldatasheet.com.

[63] "Modern Control Engineering", by Katsuhiko Ogata; University of Minnesota; third edition, 1997. Prentice Hall, New Jersey.

[64] "Digital Control Systems: Theory, Hardware and Software" by Constantine H. Houpis and Gary B. Lamont; Air Force Institute of Technology, Write Patterson Air Force Base, Ohio; second edition, 1992. McGraw-Hill Inc.

[65] "Discrete-Time Control Systems", by Katsuhiko Ogata; University of Minnesota; second edition, 1995. Prentice Hall, New Jersey.

Appendices

Appendix A - Forward Kinematics for Legs 2, 3 and 4

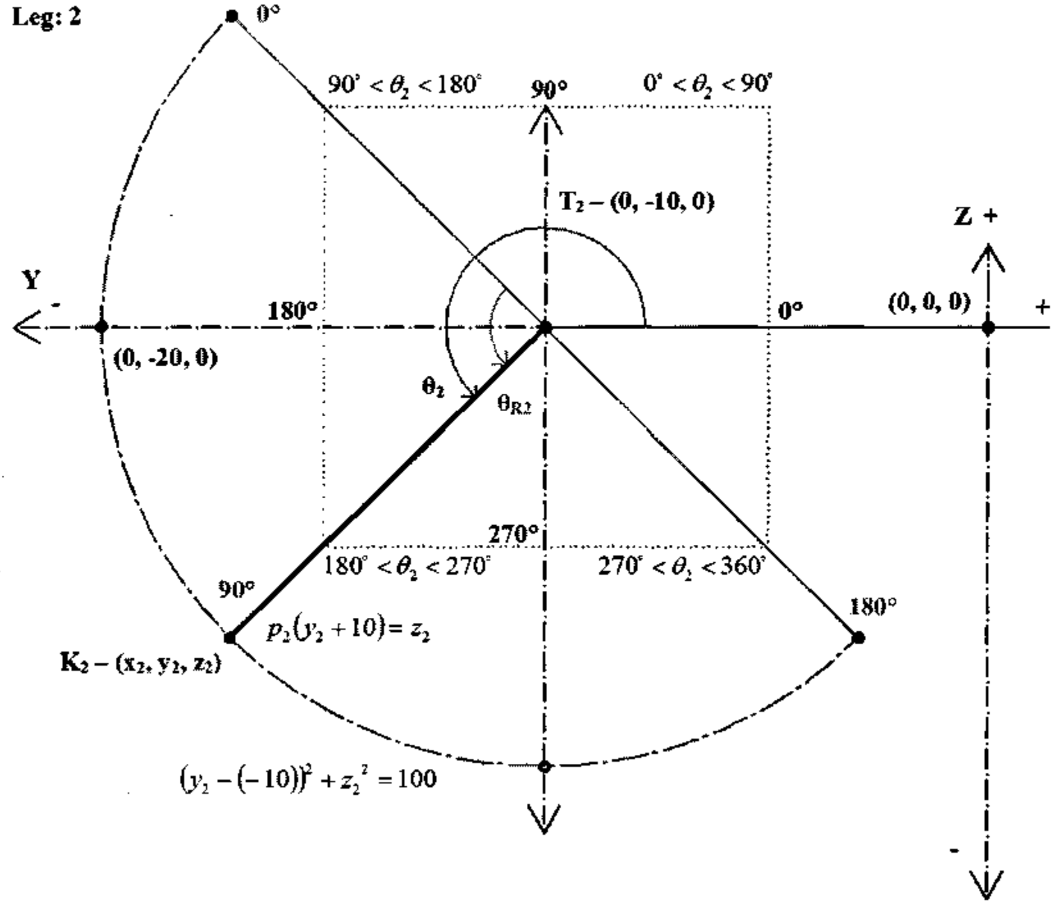


Figure 74 Illustration of leg 2 coordinate frame and angular conventions

Equation of straight line: $p_2(y_2 + 10) = z_2$ (when $y_2 = -10, z_2 = 0$. p_2 Is the gradient of the line and $p_2 = \tan\theta_2$.)

Equation of circle: $(y_2 - (-10))^2 + z_2^2 = 100$

$$= \quad y_2^2 + 20y_2 + 100 + z_2^2 = 100$$

$$\Rightarrow \quad y_2^2 + 20y_2 + z_2^2 = 0$$

$$\Rightarrow \quad y_2^2 + 20y_2 + (\tan\theta_2(y_2 + 10))^2 = 0$$

$$\Rightarrow \quad y_2^2 + 20y_2 + y_2^2\tan^2\theta_2 + 20y_2\tan^2\theta_2 + 100\tan^2\theta_2 = 0$$

$$\Rightarrow \quad y_2^{\,2}\left(1+\tan^2\theta_2\right)+20y_2\left(1+\tan^2\theta_2\right)+100\tan^2\theta_2 \quad = \quad 0$$

This is a quadratic in y_2, and using the binomial formula to resolve y_2 yields:

$$\Rightarrow \quad y_2 = \frac{-20(1+\tan^2\theta_2)\pm\sqrt{400\left(1+\tan^2\theta_2\right)-400\tan^2\theta_2\left(1+\tan^2\theta_2\right)}}{2\left(1+\tan^2\theta_2\right)}$$

$$= -10\pm\frac{10\sqrt{1+2\tan^2\theta_2+\tan^4\theta_2-\tan^2\theta_2-\tan^4\theta_2}}{1+\tan^2\theta_2}$$

$$= -10\pm\frac{10}{\sqrt{1+\tan^2\theta_2}}, \quad \forall \;\; \theta_2 \neq 90^\circ + k.180^\circ, \quad k \in \mathrm{N}_0$$

The transform used to obtain θ_2 from θ_{R2} is:

$$\theta_2 \; = \; 135^\circ + \theta_{R2} \quad 0^\circ \le \theta_{R2} \le 180^\circ$$

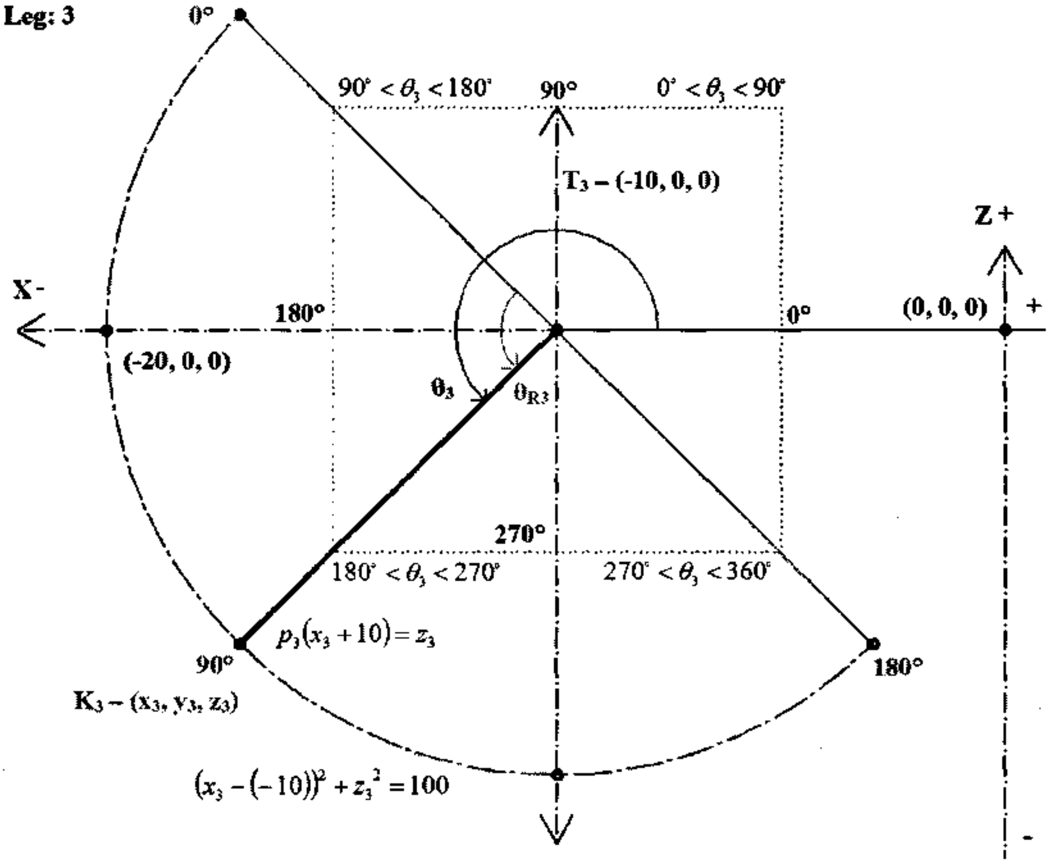

Leg: 3
0°
$90^\circ < \theta_3 < 180^\circ$
90°
$0^\circ < \theta_3 < 90^\circ$
T₃ – (-10, 0, 0)
Z+
X-
180°
0°
(0, 0, 0)
+
(-20, 0, 0)
θ₃
θR3
270°
$180^\circ < \theta_3 < 270^\circ$
$270^\circ < \theta_3 < 360^\circ$
$p_3(x_3+10)=z_3$
90°
K₃ – (x₃, y₃, z₃)
180°
$(x_3-(-10))^2+z_3^{\,2}=100$
-

Figure 75 Illustration of leg 3 coordinate frame and angular conventions

Equation of straight line: $p_3(x_3+10)=z_3$ (when $x_3=-10, z_3=0$. p_3 Is the gradient of the line and $p_3=\tan\theta_3$.)

Equation of circle: $(x_3-(-10))^2+z_3^{\,2}=100$

$$= \quad x_3^{\,2}+20x_3+100+z_3^{\,2}=100$$

$$\Rightarrow \quad x_3^{\,2}+20x_3+z_3^{\,2}=0$$

$$\Rightarrow \quad x_3^{\,2}+20x_3+(\tan\theta_3(x_3+10))^2 \;=\; 0$$

$$\Rightarrow \quad x_3^{\,2}+20x_3+x_3^{\,2}\tan^2\theta_3+20x_3\tan^2\theta_3+100\tan^2\theta_3 \;=\; 0$$

$$\Rightarrow \quad x_3^{\,2}\left(1+\tan^2\theta_3\right)+20x_3\left(1+\tan^2\theta_3\right)+100\tan^2\theta_3 \;=\; 0$$

This is a quadratic in x_3, and using the binomial formula to resolve x_3 yields:

$$\Rightarrow \quad x_3 = \frac{-20(1+\tan^2\theta_3)\pm\sqrt{400\left(1+\tan^2\theta_3\right)-400\tan^2\theta_3\left(1+\tan^2\theta_3\right)}}{2\left(1+\tan^2\theta_3\right)}$$

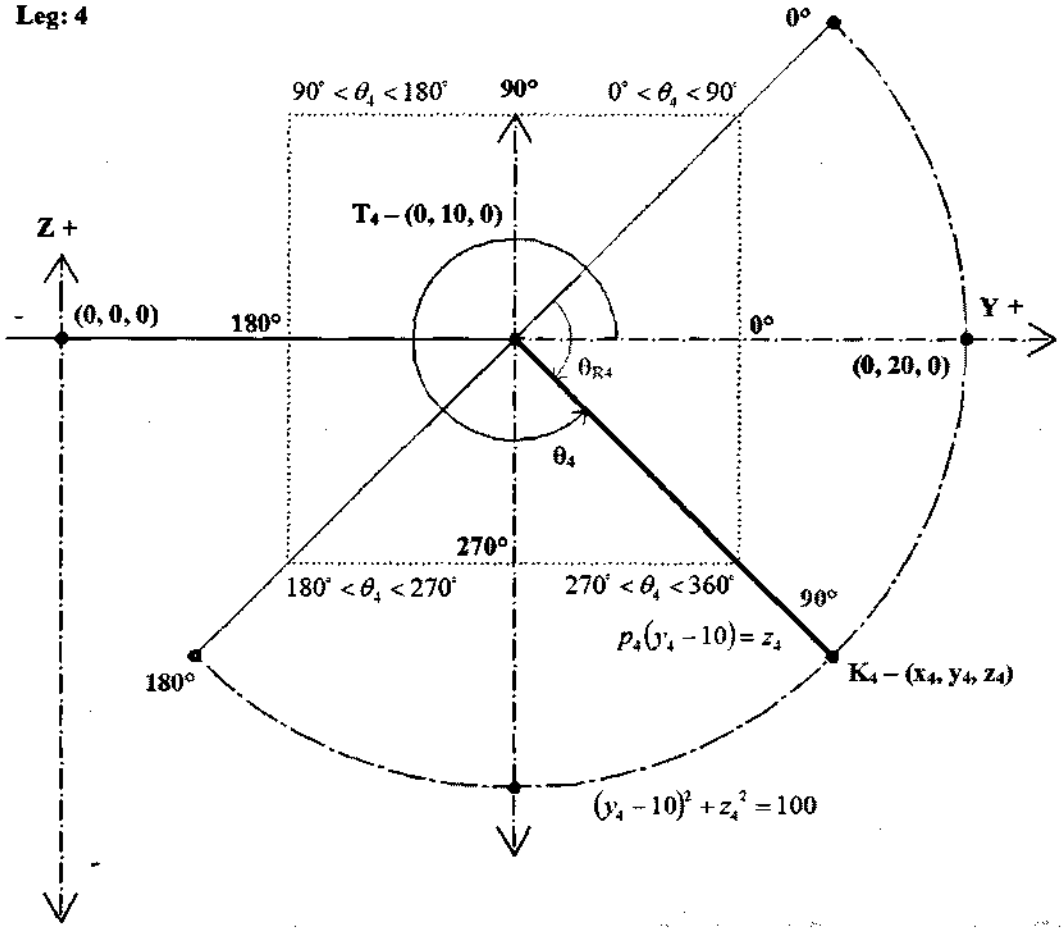
$$= -10\pm\frac{10\sqrt{1+2\tan^2\theta_3+\tan^4\theta_3-\tan^2\theta_3-\tan^4\theta_3}}{1+\tan^2\theta_3}$$

$$= -10\pm\frac{10}{\sqrt{1+\tan^2\theta_3}}, \quad \forall \;\; \theta_3\neq 90^\circ+k.180^\circ, \quad k\in \mathrm{N}_0$$

The transform used to obtain θ_3 from θ_{R3} is:

$$\theta_3 \;=\; 135^\circ+\theta_{R3} \quad 0^\circ\leq\theta_{R3}\leq 180^\circ$$

Leg: 4



0°
$90^\circ < \theta_4 < 180^\circ$
90°
$0^\circ < \theta_4 < 90^\circ$
T₄ – (0, 10, 0)
Z +
-
(0, 0, 0)
180°
0°
Y +
θ_{R4}
(0, 20, 0)
θ_4
270°
$180^\circ < \theta_4 < 270^\circ$
$270^\circ < \theta_4 < 360^\circ$
90°
$p_4(y_4 - 10) = z_4$
180°
K₄ – (x₄, y₄, z₄)
$(y_4 - 10)^2 + z_4^2 = 100$

Figure 76 Illustration of leg 4 coordinate frame and angular conventions

Equation of straight line: $p_4(y_4 - 10) = z_4$ (when $y_4 = 10, z_4 = 0$. p_4 Is the gradient of the line and $p_4 = \tan\theta_4$.)

Equation of circle: $(y_4 - 10)^2 + z_4^2 = 100$

$$= \quad y_4^2 - 20y_4 + 100 + z_4^2 = 100$$

$$\Rightarrow \quad y_4^2 - 20y_4 + z_4^2 = 0$$

$$\Rightarrow \quad y_4^2 - 20y_4 + (\tan\theta_4(y_4 - 10))^2 = 0$$

$$\Rightarrow \quad y_4^2 - 20y_4 + y_4^2\tan^2\theta_4 - 20y_4\tan^2\theta_4 + 100\tan^2\theta_4 = 0$$

$$\Rightarrow \quad y_4^2(1 + \tan^2\theta_4) - 20y_4(1 + \tan^2\theta_4) + 100\tan^2\theta_4 = 0$$

This is a quadratic in y_4, and using the binomial formula to resolve y_4 yields:

$$\Rightarrow \qquad y_4 = \frac{20(1+\tan^2\theta_4) \pm \sqrt{400(1+\tan^2\theta_4) - 400\tan^2\theta_4(1+\tan^2\theta_4)}}{2(1+\tan^2\theta_4)}$$

$$= 10 \pm \frac{10\sqrt{1+2\tan^2\theta_4+\tan^4\theta_4-\tan^2\theta_4-\tan^4\theta_4}}{1+\tan^2\theta_4}$$

$$= 10 \pm \frac{10}{\sqrt{1+\tan^2\theta_4}}, \quad \forall \ \theta_4 \neq 90^\circ + k.180^\circ, \quad k \in \mathrm{N}_0$$

The transform used to obtain θ_4 from θ_{R4} is:

$$\theta_4 = (405^\circ - \theta_{R4})\operatorname{mod}360; \ \ 0^\circ \le \theta_{R4} \le 180^\circ, \ \ 225^\circ \le \theta_4 \le 360^\circ \ \cup \ 0^\circ \le \theta_4 \le 45^\circ$$

Appendix B - Solving the Inverse Kinematics for Legs 2, 3 and 4

Leg 2:

From the circle equation:

$$(y_2+m)^2+(z_2)^2 = R_1^2 \quad \text{where } m=10 \text{ and } R_1=10.$$

$$\Rightarrow \quad (y_2+10)^2+(z_2)^2 = 10^2$$

$$= \quad y_2^{\ 2}+20y_2+100+z_2^{\ 2} = 100$$

$$\Rightarrow \quad y_2^{\ 2}+20y_2+z_2^{\ 2}=0$$

$$\Rightarrow \quad z_2^{\ 2}=-20y_2-y_2^{\ 2} \qquad .....(B1)$$

This places a restriction on y_2 as $z_2^{\ 2}$ is always non-negative, so $-20y_2-y_2^{\ 2}\geq 0$, which implies that $-20 \leq y_2 \leq 0$.

Now from the sphere equation:

$$(x_0)^2+(y_2-(y_0-n))^2+(z_2-z_0)^2 = R_0^{\ 2}$$

$$= \quad (x_0)^2+(y_2-y_0+n)^2+(z_2-z_0)^2 = R_0^{\ 2} \ \textit{where} \ n \ = \ 4.45 \ \textit{and} \ R_0=17.9.$$

$$\Rightarrow \quad (x_0)^2+(y_2-y_0+4.45)^2+(z_2-z_0)^2 = 17.9^2 = 320.41$$

$$= \quad x_0^{\ 2}+y_2^{\ 2}+y_0^{\ 2}+19.803-2y_2y_0+8.9y_2-8.9y_0+z_2^{\ 2}-2z_2z_0+z_0^{\ 2} \ = \ 320.41 \ ... \ (B2)$$

Substituting z_2 from equation (B1) into (B2) above yields:

$$\Rightarrow \quad x_0^{\ 2}+y_2^{\ 2}+y_0^{\ 2}-2y_2y_0+8.9y_2-8.9y_0+\left(-20y_2-y_2^{\ 2}\right)-2z_2z_0+z_0^{\ 2}$$

$$= \quad 320.41 \ -19.803 \ = \ 300.607$$

$$= \quad y_2(-11.1-2y_0)-2z_2z_0+\left(x_0^{\ 2}+y_0^{\ 2}-8.9y_0+z_0^{\ 2}\right)=300.607$$

Now rearrange and make z_2 the subject of the formula:

$$\Rightarrow \quad z_2=\frac{y_2(-11.1-2y_0)+\left(x_0^{\ 2}+y_0^{\ 2}-8.9y_0+z_0^{\ 2}-300.607\right)}{2z_0} \quad .... \quad (B3)$$

The unknowns here are z_2 and y_2 , the rest are known. Collecting terms and making the following substitution to ease readability results in:

$c_1 = \dfrac{-11.1 - 2y_0}{2z_0}$ and $c_2 = \dfrac{x_0^2 + y_0^2 - 8.9y_0 + z_0^2 - 300.607}{2z_0}$

$$\Rightarrow \quad z_2 = c_1 y_2 + c_2 \quad \ldots \quad (B4)$$

Squaring both sides of (B4):

$$\Rightarrow \quad z_2^2 = c_1^2 y_2^2 + 2c_1 c_2 y_2 + c_2^2$$

However from equation (B1) z_2 equals $-20y_2 - y_2^2$, so:

$$c_1^2 y_2^2 + 2c_1 c_2 y_2 + c_2^2 = -20y_2 - y_2^2$$

$$\Rightarrow \quad c_1^2 y_2^2 + y_2^2 + 2c_1 c_2 y_2 + 20y_2 + c_2^2 = (c_1^2 + 1)y_2^2 + (2c_1 c_2 + 20)y_2 + c_2^2 = 0$$

This is a quadratic in y_2 and using the binomial formula $\dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ yields both solutions i.e.:

$$\begin{aligned} y_2 &= \frac{-(2c_1c_2 + 20) \pm \sqrt{(2c_1c_2 + 20)^2 - 4(c_1^2 + 1)c_2^2}}{2(c_1^2 + 1)} \\ &= \frac{-2c_1c_2 - 20 \pm \sqrt{4c_1^2c_2^2 + 80c_1c_2 + 400 - 4c_1^2c_2^2 - 4c_2^2}}{2c_1^2 + 2} \end{aligned}$$

$$\begin{aligned} &= \frac{-2c_1c_2 - 20 \pm \sqrt{400 + 80c_1c_2 - 4c_2^2}}{2c_1^2 + 2} = \frac{-2c_1c_2 - 20 \pm 2\sqrt{100 + 20c_1c_2 - c_2^2}}{2c_1^2 + 2} \\ &= \frac{-c_1c_2 - 10 \pm \sqrt{100 + 20c_1c_2 - c_2^2}}{c_1^2 + 1} \end{aligned}$$

For real solutions to exist $100 + 20c_1c_2 - c_2^2 \geq 0$ must hold. Since y_2 is now known, having taken into account the restriction of equation (B1), z_2 can be found by taking the square root of both sides of said equation. There are 2 solutions for z_2 (a positive and a negative solution), the correct one must be selected to get the right angle for actuation. The wrong solution would mean that the leg is folded inwards instead of outwards, and would imply that it must have passed through a singularity condition (see section 4.3.2 h Singularities).

Leg 3:

From the circle equation:

$$(x_3+m)^2+(z_3)^2=R_1^2 \quad \text{where } m{=}10 \text{ and } R_1=10.$$

$$\Rightarrow \quad (x_3+10)^2+(z_3)^2 = 10^2$$

$$= \quad x_3^{\;2}+20x_3+100+z_3^{\;2} = 100$$

$$\Rightarrow \quad x_3^{\;2}+20x_3+z_3^{\;2}=0$$

$$\Rightarrow \quad z_3^{\;2}=-20x_3-x_3^{\;2} \qquad .....(B5)$$

This places a restriction on x_3 as $z_3^{\;2}$ is always non-negative, so $-20x_3-x_3^{\;2}\geq 0$, which implies that $-20 \leq x_3 \leq 0$.

Now from the sphere equation:

$$\Rightarrow \quad (x_3-(x_0-n))^2+(y_0)^2+(z_3-z_0)^2=R_0^{\;2}$$

$$= \quad (x_3-x_0+n)^2+(y_0)^2+(z_3-z_0)^2=R_0^{\;2} \text{ where } n \;=\; 4.45 \text{ and } R_0{=}17.9.$$

$$\Rightarrow \quad (x_3-x_0+4.45)^2+(y_0)^2+(z_3-z_0)^2=17.9^2=320.41$$

$$= \quad x_3^{\;2}+x_0^{\;2}+19.803-2x_3x_0+8.9x_3-8.9x_0+y_0^{\;2}+z_3^{\;2}-2z_3z_0+z_0^{\;2} \;=\; 320.41 \;...\; (B6)$$

Substituting z_3 from equation (B5) into (B6) above yields:

$$\Rightarrow \quad x_3^{\;2}+x_0^{\;2}-2x_3x_0+8.9x_3-8.9x_0+y_0^{\;2}+\left(-20x_3-x_3^{\;2}\right)-2z_3z_0+z_0^{\;2}$$

$$= \quad 320.41 \text{ -}19.803 \;=\; 300.607$$

$$= \quad x_3(-11.1-2x_0)-2z_3z_0+\left(x_0^{\;2}-8.9x_0+y_0^{\;2}+z_0^{\;2}\right)=300.607$$

Now rearrange and make z_3 the subject of the formula:

$$\Rightarrow \quad z_3=\frac{x_3(-11.1-2x_0)+\left(x_0^{\;2}-8.9x_0+y_0^{\;2}+z_0^{\;2}-300.607\right)}{2z_0} \quad .... \quad (B7)$$

The unknowns here are z_3 and x_3, the rest are known. Collecting terms and making the following substitution to ease readability results in:

$$c_1 = \frac{-11.1-2x_0}{2z_0} \quad and \; c_2 = \frac{x_0^2 - 8.9x_0 + y_0^2 + z_0^2 - 300.607}{2z_0}$$

$$\Rightarrow \qquad z_3 = c_1x_3 + c_2 \quad ... \qquad (B8)$$

Squaring both sides of (B8):

$$\Rightarrow \qquad z_3^2 = c_1^2x_3^2 + 2c_1c_2x_3 + c_2^2$$

However from equation (B5) z_3 equals -$20x_3 - x_3^2$, so:

$$c_1^2x_3^2 + 2c_1c_2x_3 + c_2^2 = -20x_3 - x_3^2$$

$$\Rightarrow \qquad c_1^2x_3^2 + x_3^2 + 2c_1c_2x_3 + 20x_3 + c_2^2 = \left(c_1^2+1\right)x_3^2 + \left(2c_1c_2 + 20\right)x_3 + c_2^2 = 0$$

This is a quadratic in x_3 and using the binomial formula $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ yields both solutions i.e.:

$$\begin{aligned} x_3 &= \frac{-\left(2c_1c_2+20\right) \pm \sqrt{\left(2c_1c_2+20\right)^2 - 4\left(c_1^2+1\right)c_2^2}}{2\left(c_1^2+1\right)} \\ &= \frac{-2c_1c_2 - 20 \pm \sqrt{4c_1^2c_2^2 + 80c_1c_2 + 400 - 4c_1^2c_2^2 - 4c_2^2}}{2c_1^2+2} \end{aligned}$$

$$\begin{aligned} &= \frac{-2c_1c_2 - 20 \pm \sqrt{400 + 80c_1c_2 - 4c_2^2}}{2c_1^2+2} = \frac{-2c_1c_2 - 20 \pm 2\sqrt{100 + 20c_1c_2 - c_2^2}}{2c_1^2+2} \\ &= \frac{-c_1c_2 - 10 \pm \sqrt{100 + 20c_1c_2 - c_2^2}}{c_1^2+1} \end{aligned}$$

For real solutions to exist $100 + 20c_1c_2 - c_2^2 \geq 0$ must hold. Since x_3 is now known, having taken into account the restriction of equation (B5), z_3 can be found by taking the square root of both sides of said equation. There are 2 solutions for z_3 (a positive and a negative solution), the correct one must be selected to get the right angle for actuation. The wrong solution would mean that the leg is folded inwards instead of outwards, and would imply that it must have passed through a singularity condition (see section 4.3.2 h Singularities).

Leg 4:

From the circle equation:

$$(y_4 - m)^2 + (z_4)^2 = R_1^{\,2} \quad \text{where } m=10 \text{ and } R_1 = 10.$$

$$\Rightarrow \quad (y_4 - 10)^2 + (z_4)^2 = 10^2$$

$$= \quad y_4^{\,2} - 20y_4 + 100 + z_4^{\,2} = 100$$

$$\Rightarrow \quad y_4^{\,2} - 20y_4 + z_4^{\,2} = 0$$

$$\Rightarrow \quad z_4^{\,2} = 20y_4 - y_4^{\,2} \qquad .....(B9)$$

This places a restriction on y_4 as $z_4^{\,2}$ is always non-negative, so $20y_4 - y_4^{\,2} \geq 0$, which implies that $0 \leq y_4 \leq 20$.

Now from the sphere equation:

$$(x_0)^2 + (y_4 - (y_0 + n))^2 + (z_4 - z_0)^2 = R_0^{\,2}$$

$$= \quad (x_0)^2 + (y_4 - y_0 - n)^2 + (z_4 - z_0)^2 = R_0^{\,2} \text{ where } n = 4.45 \text{ and } R_0 = 17.9.$$

$$\Rightarrow \quad (x_0)^2 + (y_4 - y_0 - 4.45)^2 + (z_4 - z_0)^2 = 17.9^2 = 320.41$$

$$= \quad x_0^{\,2} + y_4^{\,2} + y_0^{\,2} + 19.803 - 2y_4y_0 - 8.9y_4 + 8.9y_0 + z_4^{\,2} - 2z_4z_0 + z_0^{\,2} = 320.41 \ ... \ (B10)$$

Substituting z_4 from equation (B9) into (B10) above yields:

$$\Rightarrow \quad x_0^{\,2} + y_4^{\,2} + y_0^{\,2} - 2y_4y_0 - 8.9y_4 + 8.9y_0 + \left(20y_4 - y_4^{\,2}\right) - 2z_4z_0 + z_0^{\,2}$$

$$= \quad 320.41 - 19.803 = 300.607$$

$$= \quad y_4(11.1 - 2y_0) - 2z_4z_0 + \left(x_0^{\,2} + y_0^{\,2} + 8.9y_0 + z_0^{\,2}\right) = 300.607$$

Now rearrange and make z_4 the subject of the formula :

$$\Rightarrow \quad z_4 = \frac{y_4(11.1 - 2y_0) + \left(x_0^{\,2} + y_0^{\,2} + 8.9y_0 + z_0^{\,2} - 300.607\right)}{2z_0} \quad .... \quad (B11)$$

The unknowns here are z_4 and y_4, the rest are known. Collecting terms and making the following substitution to ease readability results in:

$$c_1 = \frac{11.1 - 2y_0}{2z_0} \text{ and } c_2 = \frac{x_0^{\,2} + y_0^{\,2} + 8.9y_0 + z_0^{\,2} - 300.607}{2z_0}$$

$$\Rightarrow \quad z_4 = c_1y_4 + c_2 \ ... \quad (B12)$$

Squaring both sides of (B12):

$$\Rightarrow \quad z_4^2 = c_1^2 y_4^2 + 2c_1c_2y_4 + c_2^2$$

However from equation (B9) z_4 equals $20y_4 - y_4^2$, so :

$$c_1^2 y_4^2 + 2c_1c_2y_4 + c_2^2 = 20y_4 - y_4^2$$

$$\Rightarrow \quad c_1^2 y_4^2 + y_4^2 + 2c_1c_2y_4 - 20y_4 + c_2^2 = (c_1^2+1)y_4^2 + (2c_1c_2 - 20)y_4 + c_2^2 = 0$$

This is a quadratic in y_4 and using the binomial formula $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ yields both solutions i.e.:

$$y_4 = \frac{-(2c_1c_2-20) \pm \sqrt{(2c_1c_2-20)^2 - 4(c_1^2+1)c_2^2}}{2(c_1^2+1)}$$
$$= \frac{-2c_1c_2 + 20 \pm \sqrt{4c_1^2c_2^2 - 80c_1c_2 + 400 - 4c_1^2c_2^2 - 4c_2^2}}{2c_1^2+2}$$

$$= \frac{-2c_1c_2 + 20 \pm \sqrt{400 - 80c_1c_2 - 4c_2^2}}{2c_1^2+2} = \frac{-2c_1c_2 + 20 \pm 2\sqrt{100 - 20c_1c_2 - c_2^2}}{2c_1^2+2}$$
$$= \frac{-c_1c_2 + 10 \pm \sqrt{100 - 20c_1c_2 - c_2^2}}{c_1^2+1}$$

For real solutions to exist $100 - 20c_1c_2 - c_2^2 \geq 0$ must hold. Since y_4 is now known, having taken into account the restriction of equation (B9), z_4 can be found by taking the square root of both sides of said equation. There are 2 solutions for z_4 (a positive and a negative solution), the correct one must be selected to get the right angle for actuation. The wrong solution would mean that the leg is folded inwards instead of outwards, and would imply that it must have passed through a singularity condition (see section 4.3.2 h Singularities).

Appendix C - Light Sensor Characteristics [55]

Table C1 - Comparison of Light Sensor Characteristics

Electrical Characteristics	Photo-multiplier Tubes	Photo-diodes	Photo-transistors	CdS Photocells	Other Photo-conductors	Integrated Circuits	Hybrids	Sensor Electronic Assembly
Available Wavelengths (µm)	0.2-0.9	0.2-2.0	0.4-1.1	0.4-0.7	2-15	0.2-1.1	0.2-15.0	0.2-15.0
Performance-to-cost ratio	Fair	Good	Excellent	Excellent	Fair	Fair	Fair	Good
Sensitivity	Excellent	Very Good	Very Good	Very Good	Very Good	Very Good	Very Good	Very Good
Linearity	Good	Excellent	Good	Good	Good	Good	Good	Good
Ambient Noise Performance	Fair	Very Good	Very Good	Very Good	Very Good	Excellent	Excellent	Excellent
Dynamic Range	Very Good	Excellent	Very Good	Good	Good	Very Good	Very Good	Very Good
Stability	Very Good	Very Good	Good	Poor	Fair	Very Good	Very Good	Very Good
Reproducibility	Fair	Excellent	Fair	Poor	Fair	Very Good	Very Good	Very Good
Cost	High	Low	Very Low	Very Low	High	Medium	High	Medium
Ruggedness	Poor	Excellent	Excellent	Excellent	Good	Excellent	Very Good	Excellent
Physical Size	Large	Small	Small	Small	Small	Small	Medium	Medium
Ease of Customization	Poor	Easy	Fair	Fair	Poor	Poor	Poor	Fair
Cost of Customization	Very High	Low	Medium	Low	High	Very High	High	Medium
Lead time for Customization (weeks)	40	12	14	12	20	40	30	16

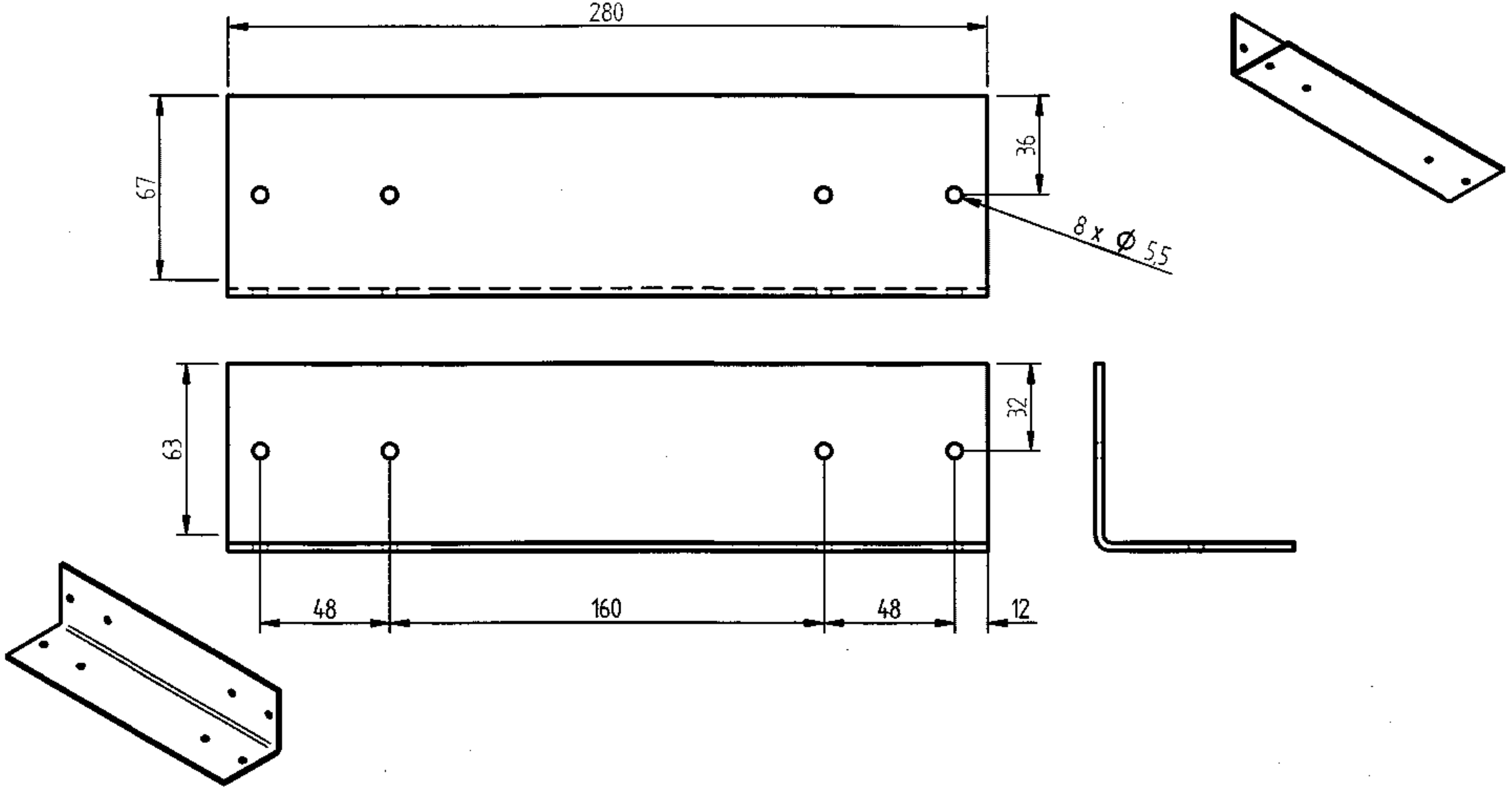
Appendix D - Measurements

D1. PKM positioning repeatability / accuracy

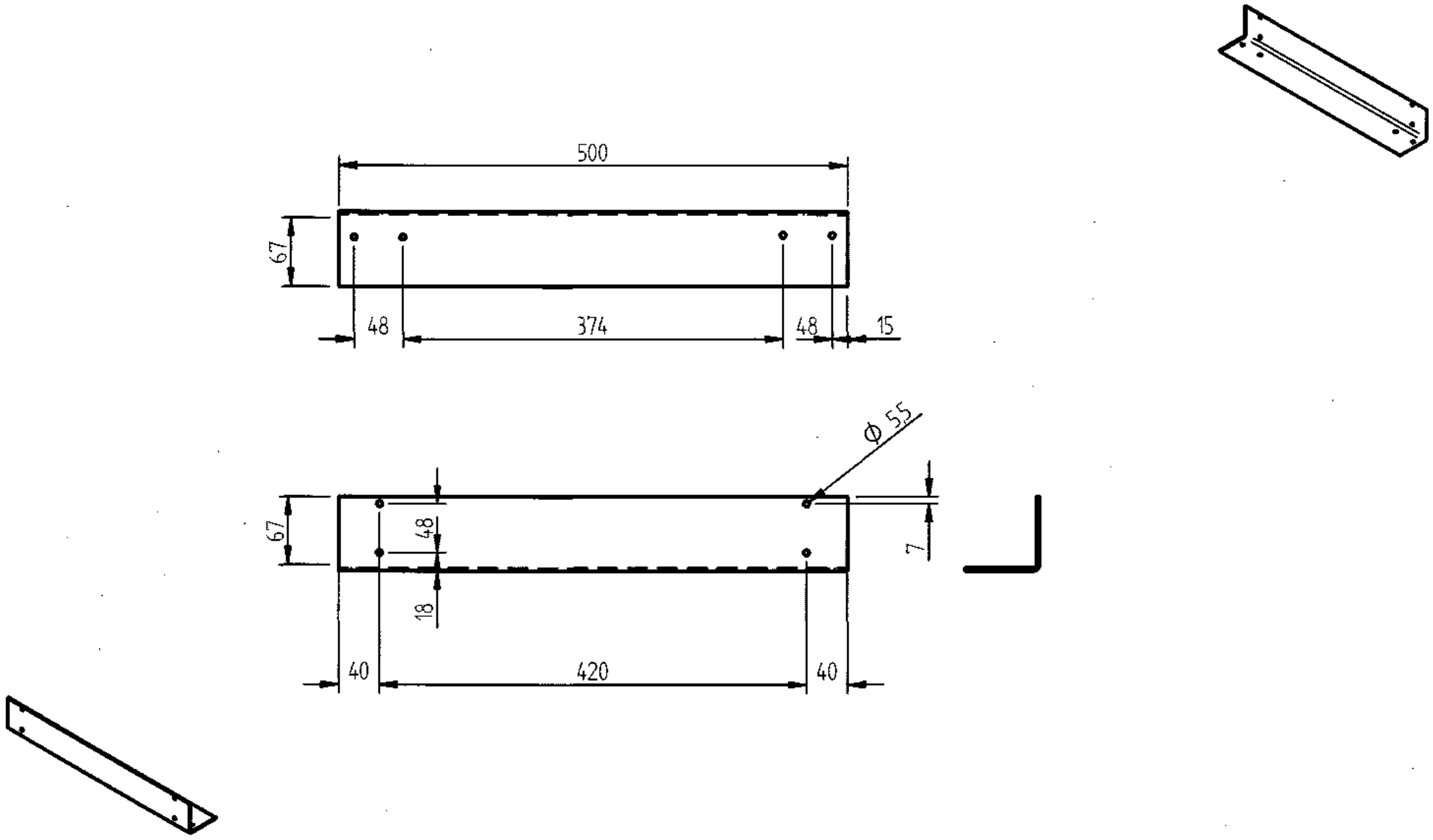
(0; 0; -20)			(-7; -7; -18)			(5; -5; -16)		
dx	dy	dz	dx	dy	dz	dx	dy	dz
1	2	-1	0	-2	1	2	1	2
2	2	3	1	-2	1	-2	1	2
1	1	-2	1	1	2	2	0	1
-1	0	-2	-2	1	1	2	0	1
1	1	1	0	2	-2	1	0	-2
-1	1	1	1	-1	-2	0	1	-2
-1	-1	1	2	-1	-1	0	-1	-2
-2	-1	1	1	-1	2	-1	-1	2
-2	1	1	1	2	3	-1	-1	2
2	-2	-1	1	2	-3	1	1	2
1	-2	1	-2	-2	-2	1	2	1
0	1	1	-2	-3	1	1	1	1
1	1	0	2	2	1	1	2	-3
1	2	0	2	2	1	-2	-3	2
1	0	1	2	1	0	1	1	1

D2. Sensor System Repeatability

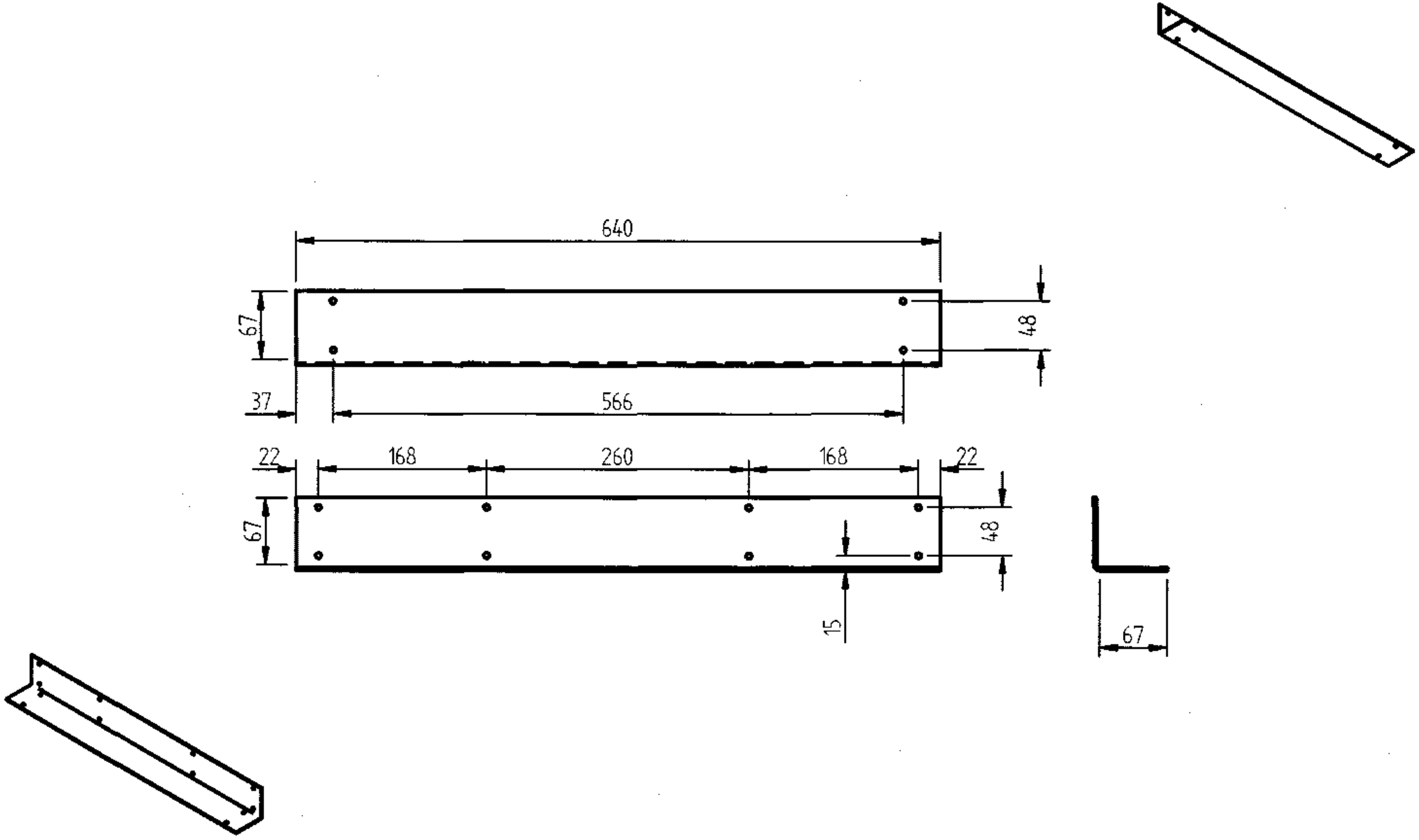
Sensor	**At 14 cm**	**At 17 cm**	**At 20 cm**
1	25	25	25
2	24	25	25
3	25	24	24
4	25	25	25
5	25	25	25
6	25	24	23
7	25	25	25
8	25	25	25
9	25	25	25
10	24	24	25
11	25	25	23
12	24	24	25
13	25	25	24
14	25	24	25
15	25	25	24
16	25	25	24
%	0.9925	0.9875	0.98



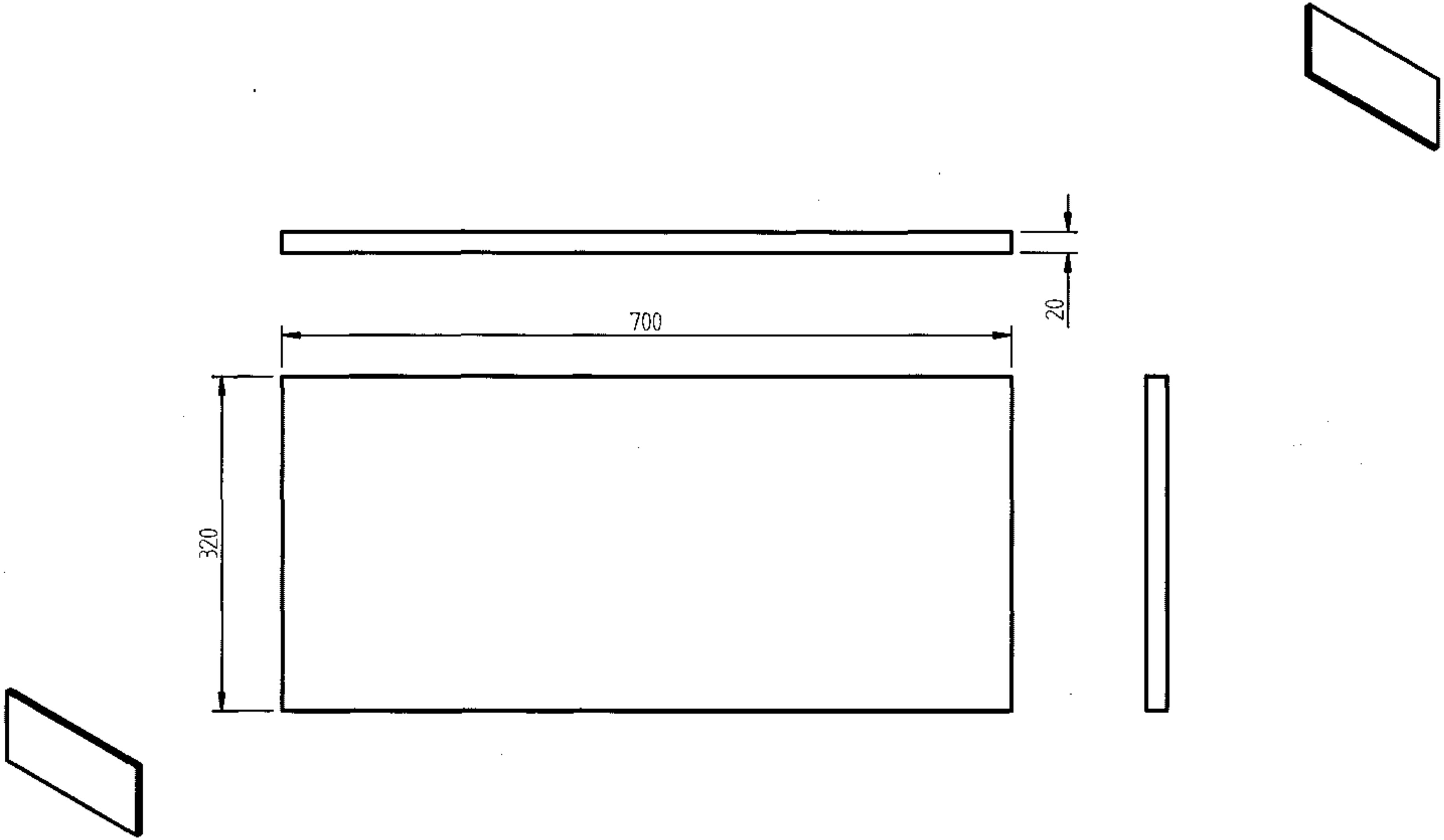
UNIVERSITY OF KWA-ZULU NATAL School of Mechanical Engineering		DATE	CHECKED	SCALE 1: 2	UNITS : mm	TITLE: Breathwise Bracket	No. 1
	W'Shop Technician			MATERIAL: Aluminium	NOTES: None		
	Draftsperson			STUDENT NAME : Ahmed Shaik		TEL No. : 072 430 0739/ 260 1226 E-MAIL : shaika@ukzn.ac.za	
	Project Supervisor			Project : Flexipicker Robot			



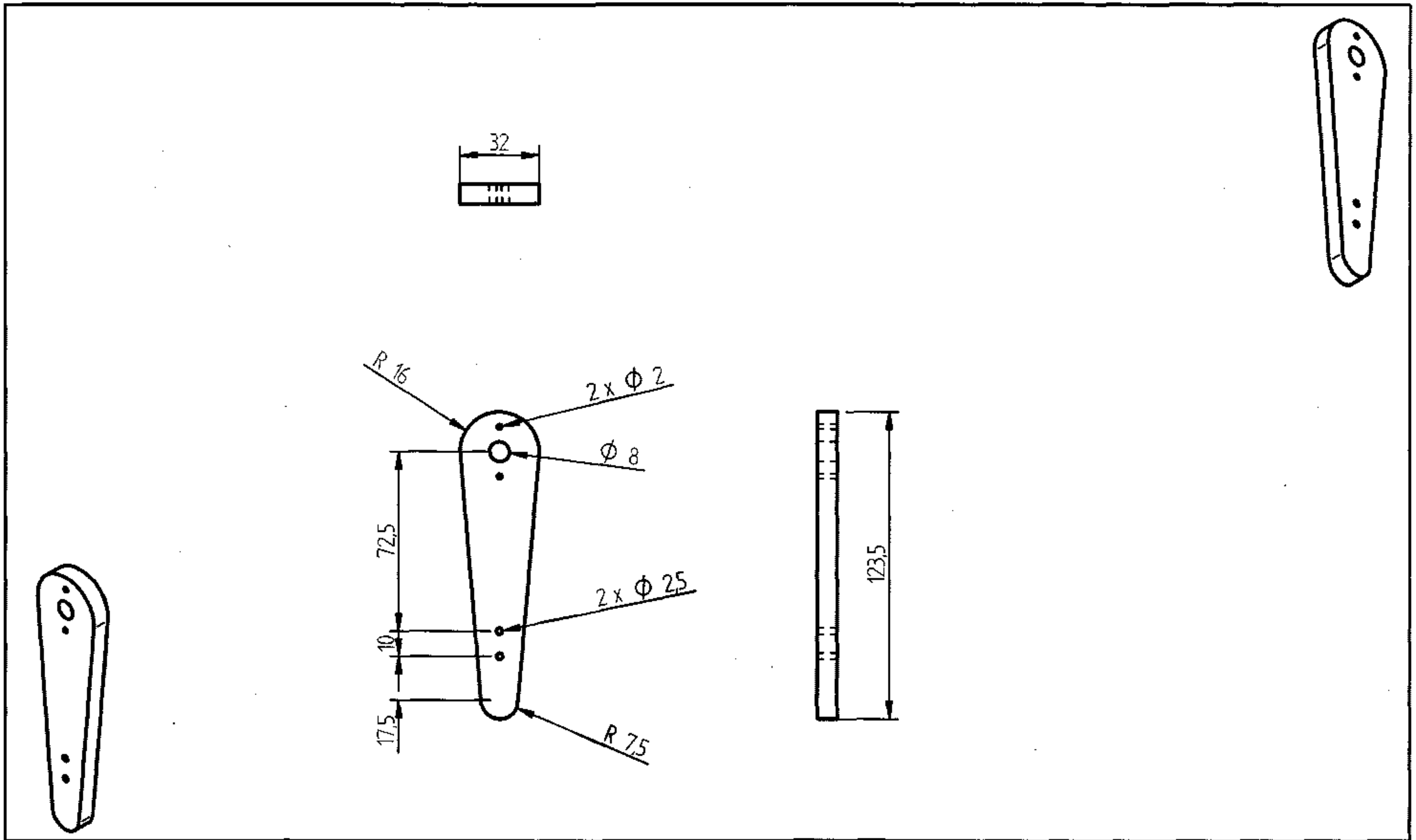
UNIVERSITY OF KWA-ZULU NATAL School of Mechanical Engineering		DATE	CHECKED	SCALE 1: 5	UNITS : mm	No. 2
	WShop Technician			MATERIAL: Aluminium	NOTES: None TITLE: Heightwise Bracket	
	Draftsperson			STUDENT NAME : Ahmed Shaik	TEL No. : 072 430 0739/ 260 1226 E-MAIL : shaika@ukzn.ac.za	
	Project Supervisor			Project : Flexipicker Robot		



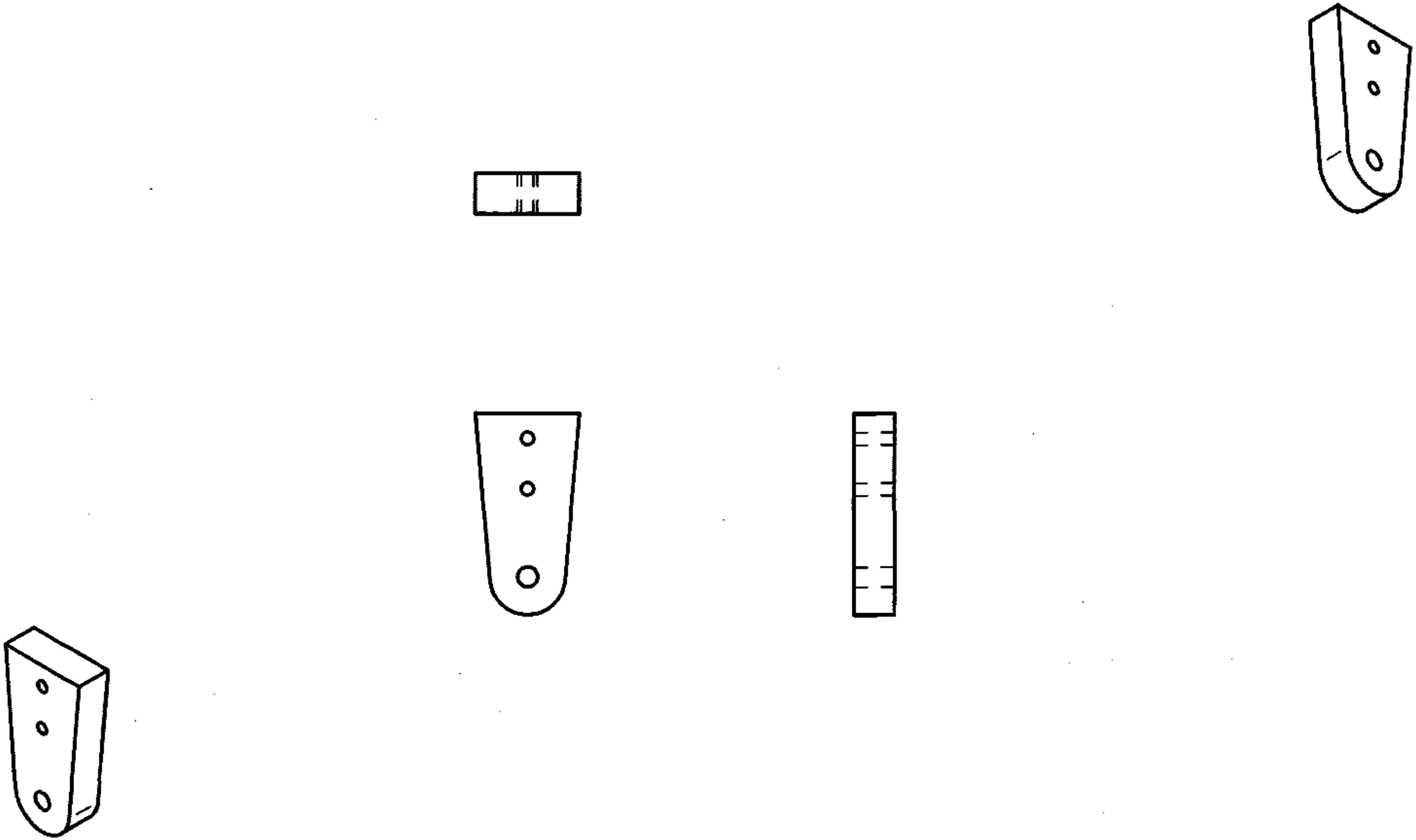
UNIVERSITY OF KWA-ZULU NATAL School of Mechanical Engineering		DATE	CHECKED	SCALE 1: 5	UNITS : mm		No. 3
	W'Shop Technician			MATERIAL: Aluminium	NOTES: None	TITLE: Lengthwise Bracket	
	Draftsperson			STUDENT NAME : Ahmed Shaik		TEL No : 072 430 0739/ 260 1226 E-MAIL : shaika@ukzn.ac.za	
	Project Supervisor			Project : Flexipicker Robot			



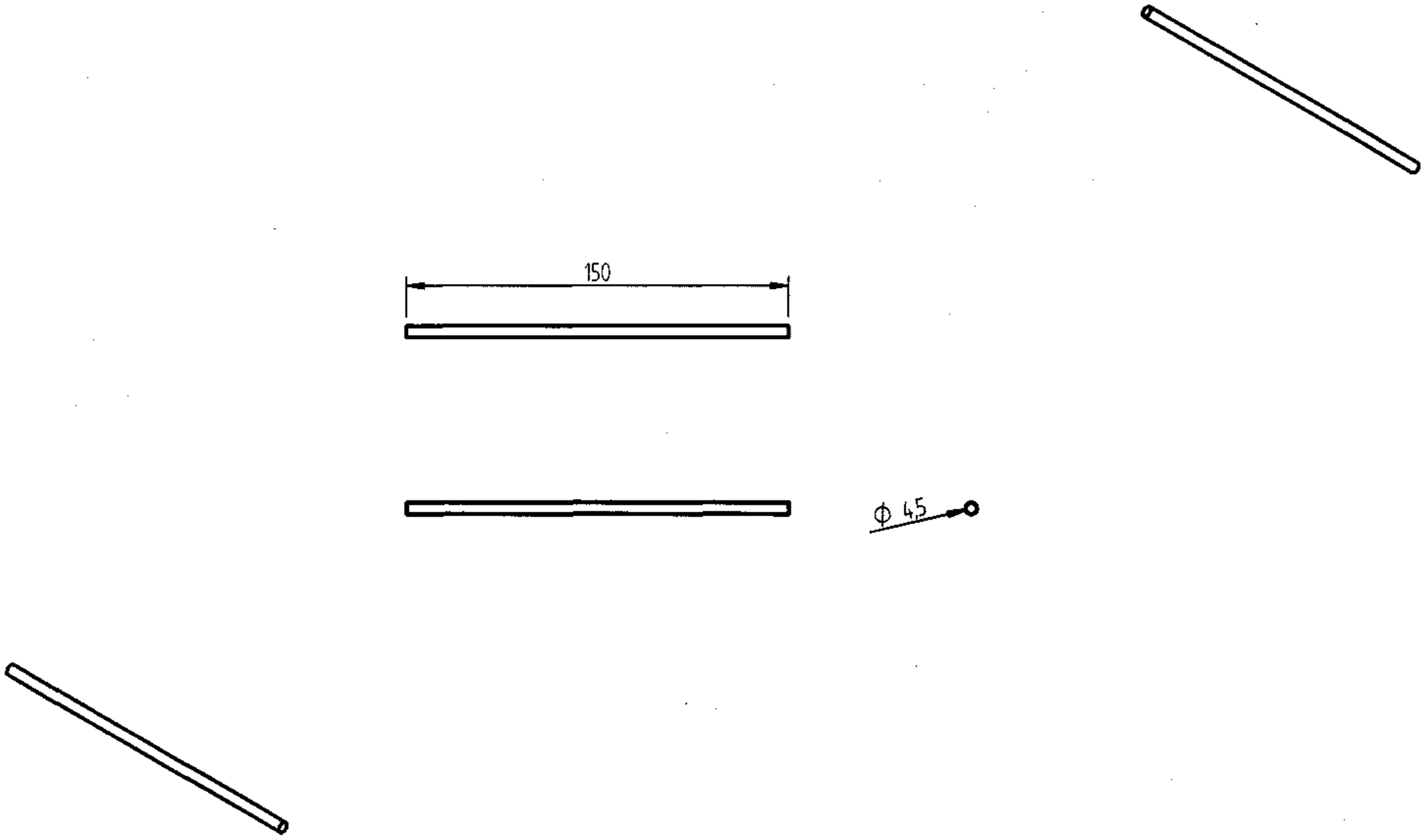
UNIVERSITY OF KWA-ZULU NATAL School of Mechanical Engineering		DATE	CHECKED	SCALE 1: 5	UNITS : mm	TITLE: Wooden Base	No. 4
	W'Shop Technician			MATERIAL: Wood	NOTES: None		
	Draftsperson			STUDENT NAME : Ahmed Shaik		TEL No. : 072 430 0739/ 260 1226 E-MAIL : shaika@ukzn.ac.za	
	Project Supervisor			Project : Flexipicker Robot			



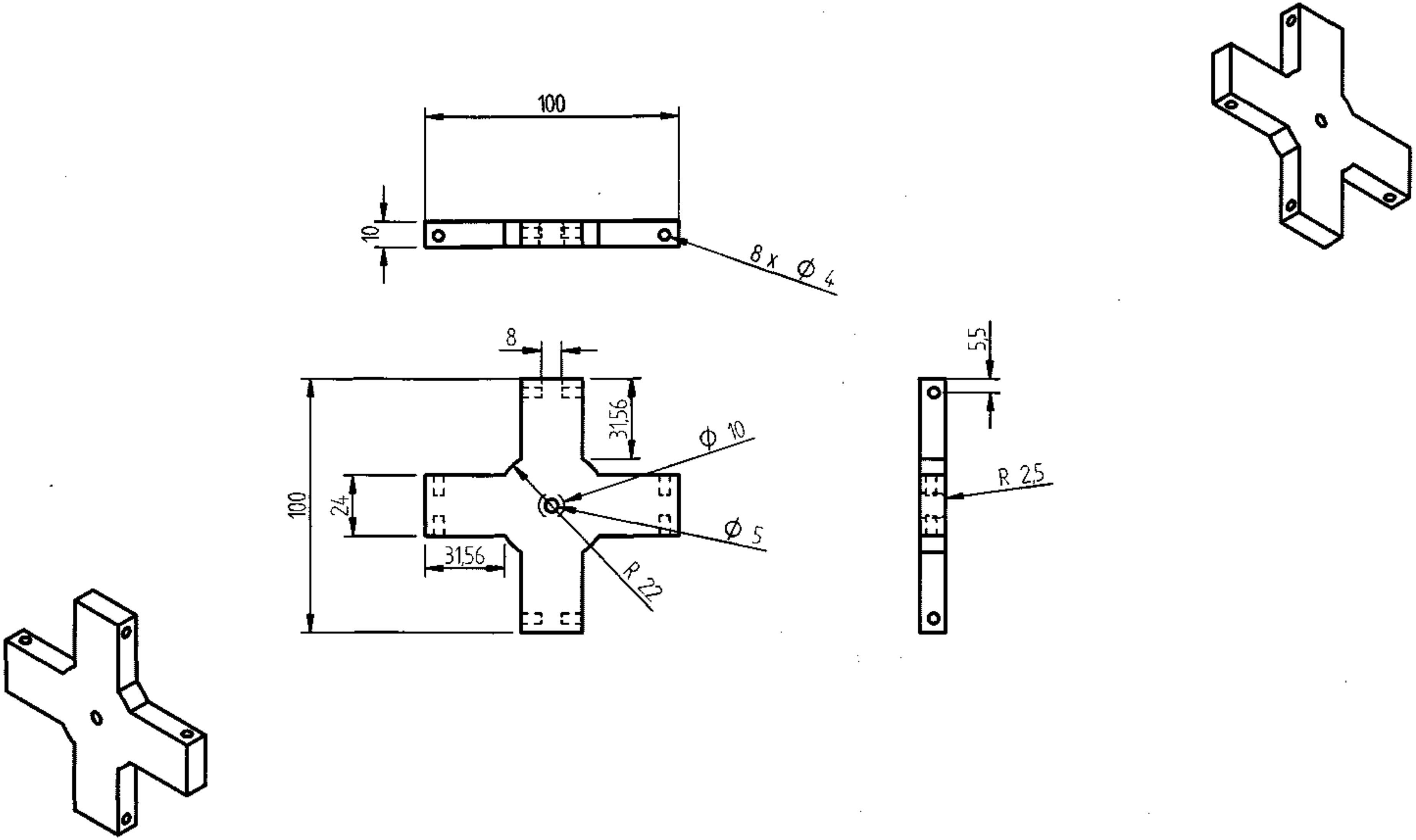
UNIVERSITY OF KWA-ZULU NATAL School of Mechanical Engineering		DATE	CHECKED	SCALE 1: 2	UNITS : mm		No. 5
	W'Shop Technician			MATERIAL: Nylon	NOTES: None	TITLE: Upper Leg	
	Draftsperson			STUDENT NAME : Ahmed Shaik		TEL No. : 072 430 0739/ 260 1226 E-MAIL : shaika@ukzn.ac.za	
	Project Supervisor			Project : Flexipicker Robot			



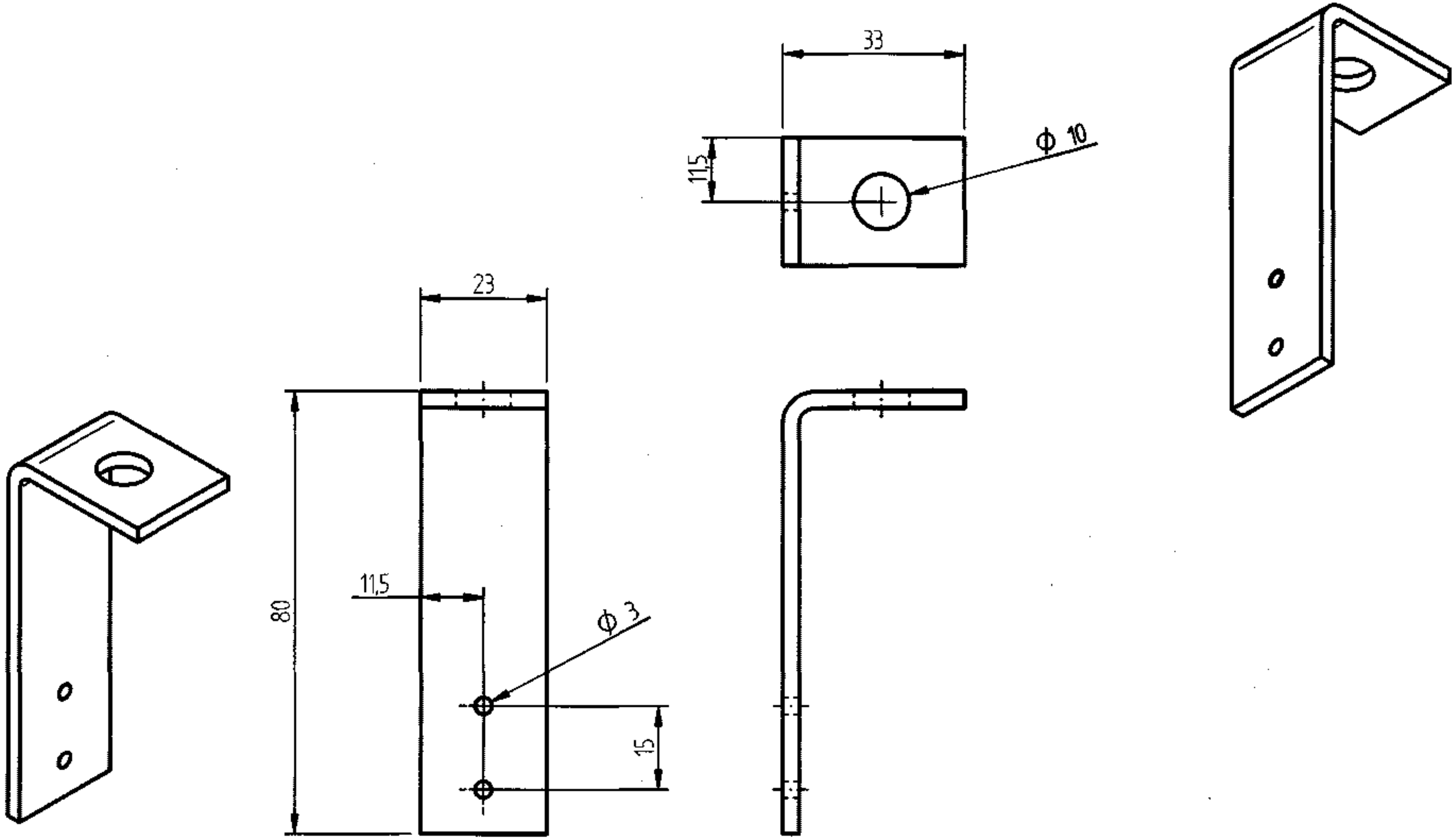
UNIVERSITY OF KWA-ZULU NATAL School of Mechanical Engineering		DATE	CHECKED	SCALE 1 : 1	UNITS : mm		
	W'Shop Technician			MATERIAL: Nylon	NOTES: None	TITLE: Upper Arm Ball Mounting	No. 6
	Draftsperson			STUDENT NAME : Ahmed Shaik		TEL No. : 072 430 0739/ 260 1226 E-MAIL : shaika@ukzn.ac.za	⊲ ⊕
	Project Supervisor			Project : Flexipicker Robot			



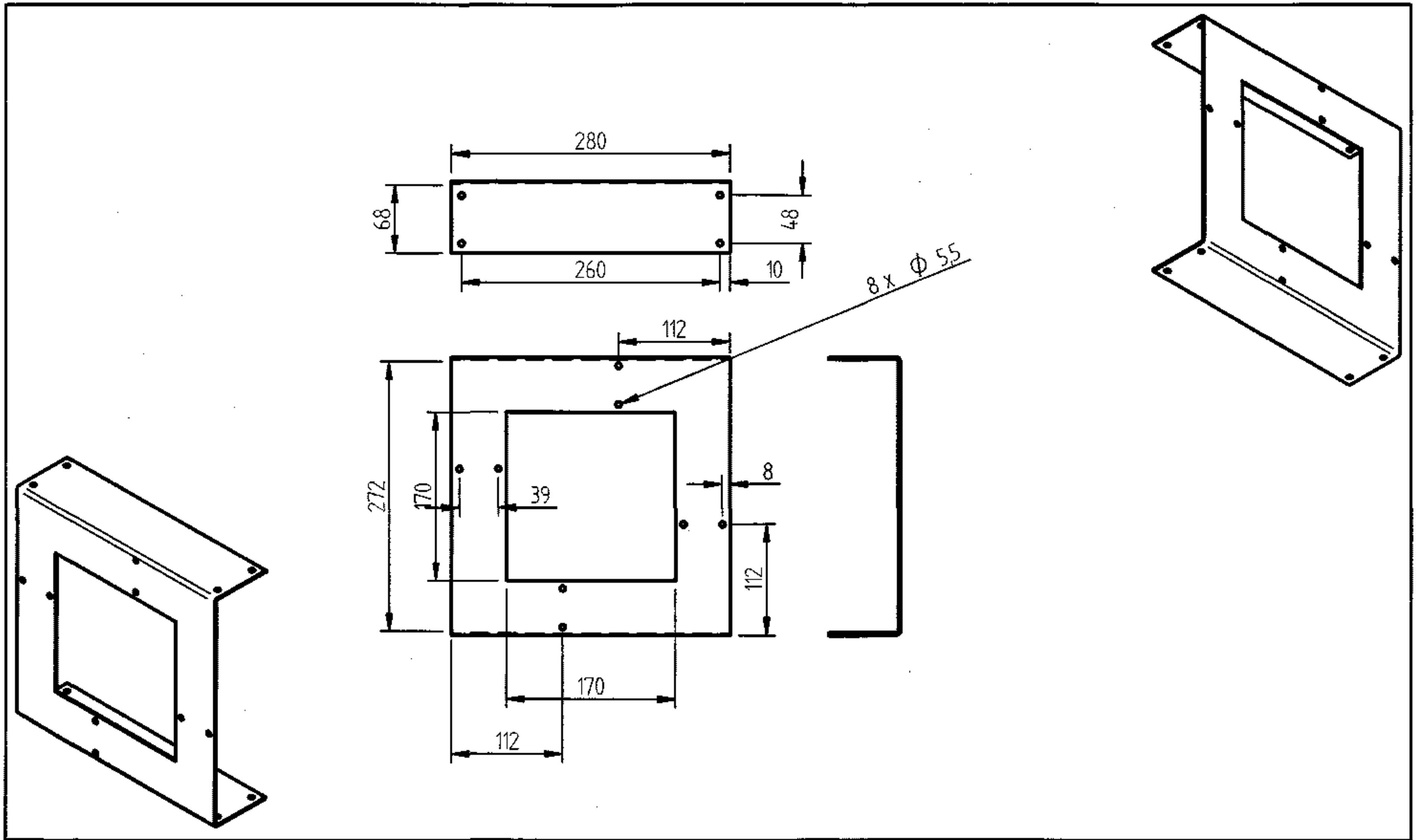
UNIVERSITY OF KWA-ZULU NATAL School of Mechanical Engineering		DATE	CHECKED	SCALE 1: 2	UNITS : mm	TITLE: Lower Leg	No. 7
	WShop Technician			MATERIAL. Steel	NOTES: None		
	Draftsperson			STUDENT NAME : Ahmed Shaik		TEL No. : 072 430 0739/ 260 1226 E-MAIL shaika@ukzn.ac.za	
	Project Supervisor			Project : Flexipicker Robot			



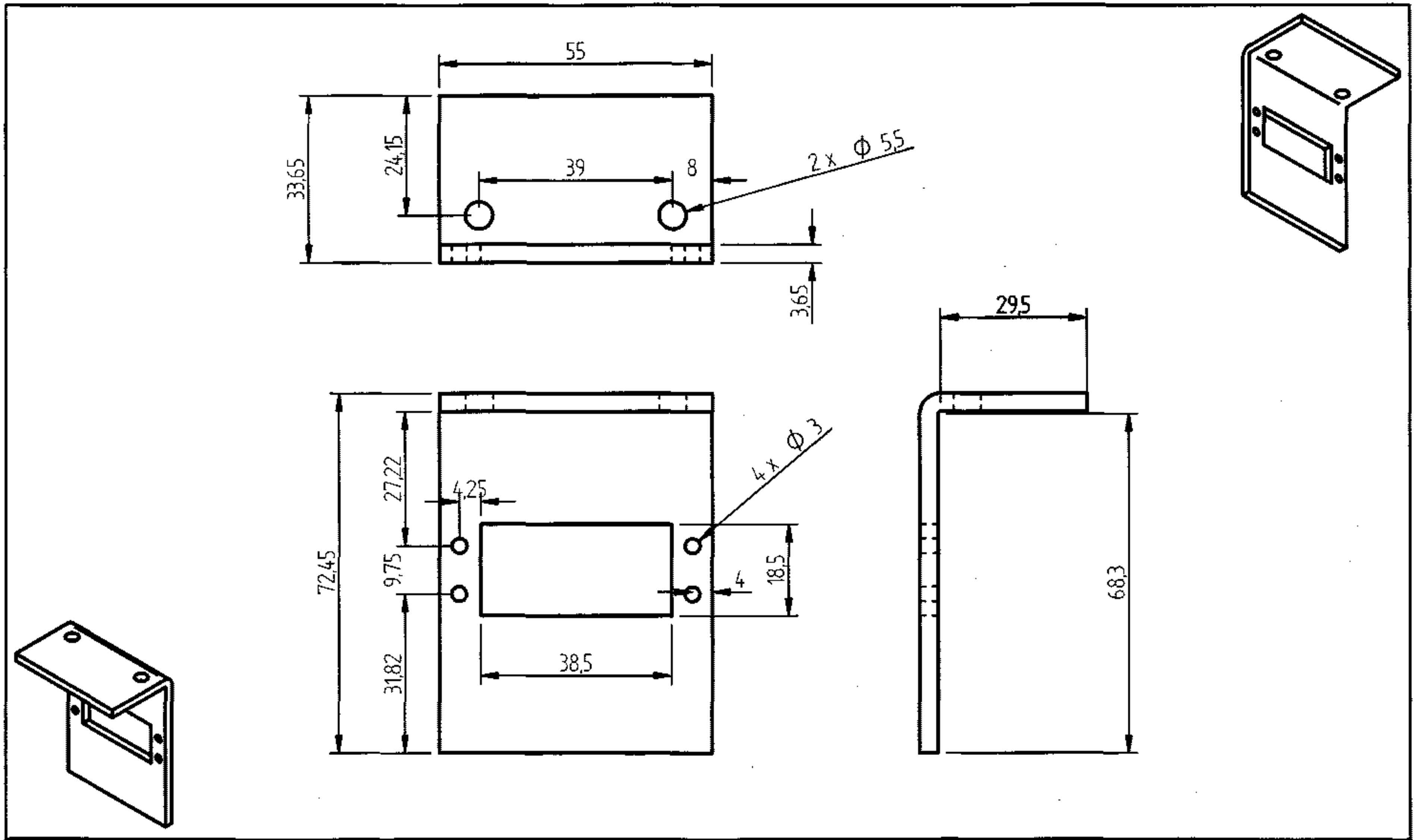
UNIVERSITY OF KWA-ZULU NATAL School of Mechanical Engineering		DATE	CHECKED	SCALE 1: 2	UNITS : mm	TITLE: End Effector	No. 8
	W'Shop Technician			MATERIAL: Nylon	NOTES: None		
	Draftsperson			STUDENT NAME : Ahmed Shaik		TEL No.: 072 430 0739/ 260 1226 E-MAIL: shaika@ukzn.ac.za	
	Project Supervisor			Project : Flexipicker Robot			



UNIVERSITY OF KWA-ZULU NATAL School of Mechanical Engineering		DATE	CHECKED	SCALE 1: 1	UNITS : mm		No. 9
	W'Shop Technician			MATERIAL: Aluminium	NOTES: None	TITLE: Second Laser Mounting	
	Draftsperson			STUDENT NAME : Ahmed Shaik		TEL No. : 072 430 0739/ 260 1226 E-MAIL : shaika@ukzn.ac.za	
	Project Supervisor			Project : Flexipicker Robot			



UNIVERSITY OF KWA-ZULU NATAL School of Mechanical Engineering		DATE	CHECKED	SCALE 1: 5	UNITS : mm		No. 10
	W'Shop Technician			MATERIAL Aluminium	NOTES: None	TITLE: Servo mounting frame	
	Draftsperson			STUDENT NAME : Ahmed Shaik		TEL No. : 072 430 0739/ 260 1226 E-MAIL : shaika@ukzn.ac.za	
	Project Supervisor			Project : Flexipicker Robot			



		DATE	CHECKED				
UNIVERSITY OF KWA-ZULU NATAL School of Mechanical Engineering				SCALE 1: 1	UNITS : mm	TITLE: Servo Mounting Bracket	No. 11
	W'Shop Technician			MATERIAL: Aluminium	NOTES: None		
	Draftsperson			STUDENT NAME : Ahmed Shaik		TEL No. : 072 430 0739/ 260 1226 E-MAIL : shaika@ukzn.ac.za	
	Project Supervisor			Project : Flexipicker Robot			

Appendix F - Code

MATLAB CODE
FLEXPICKER SIMULATION
% Solving the forward kinematics and describing the workspace

```
  % Clear Command window
  clc
  S_Angles = [0 0 0 0];
  coee = [0 0 0];

  menu_selection = 4;

  while (menu_selection ~= 0)

    fprintf('\nMENU ... ');
    fprintf('\n\n1. Inverse Kinematics (with Forward Kinematics Check) ');
    fprintf('\n2. Forward Kinematics (with Inverse Kinematics Check) ');
    menu_selection = input('\n3. Vibration Model ... ');


    %menu_selection = 1; % remove
    if (menu_selection == 1)

      fprintf('\nEnter end effector coordinates ... \t');
      fprintf('x Range: -7.5 to 7.5 \ty Range: -7.5 to 7.5 \tz Range: -25 to -15 ');
      coee(1) = input('\n\nx0 : ');
      coee(2) = input('\ny0 : ');
      coee(3) = input('\nz0 : ');

      fprintf('\n\nThe coordinates you have selected ... ');
      fprintf('[%8.4f\t%8.4f\t%8.4f]', coee);

      [S_Angles(1), S_Angles(2), S_Angles(3), S_Angles(4), kj1, kj2, kj3, kj4] = Inverse_Kinematics(coee(1), coee(2),
coee(3));
      fprintf('\n\nAngle1 = %8.4f\t\t[x1 y1 z1] = [%8.4f\t%8.4f\t%8.4f]',S_Angles(1) , kj1);
      fprintf('\nAngle2 = %8.4f\t\t[x2 y2 z2] = [%8.4f\t%8.4f\t%8.4f]', S_Angles(2), kj2);
      fprintf('\nAngle3 = %8.4f\t\t[x3 y3 z3] = [%8.4f\t%8.4f\t%8.4f]', S_Angles(3), kj3);
      fprintf('\nAngle4 = %8.4f\t\t[x4 y4 z4] = [%8.4f\t%8.4f\t%8.4f]', S_Angles(4), kj4);

      fprintf('\n\n\nChecking solution with Forward Kinematics...');
      fprintf('\n\n[Angle1  Angle2  Angle3  Angle4] = [%8.4f\t%8.4f\t%8.4f\t%8.4f]', S_Angles);

      [coee(1), coee(2), coee(3), kj1, kj2, kj3, kj4] = Forward_Kinematics(S_Angles(1), S_Angles(2), S_Angles(3),
S_Angles(4));
      fprintf('\n\nAngle1 = %8.4f\t\t[x1 y1 z1] = [%8.4f\t%8.4f\t%8.4f]',S_Angles(1) , kj1);
      fprintf('\nAngle2 = %8.4f\t\t[x2 y2 z2] = [%8.4f\t%8.4f\t%8.4f]', S_Angles(2), kj2);
      fprintf('\nAngle3 = %8.4f\t\t[x3 y3 z3] = [%8.4f\t%8.4f\t%8.4f]', S_Angles(3), kj3);
      fprintf('\nAngle4 = %8.4f\t\t[x4 y4 z4] = [%8.4f\t%8.4f\t%8.4f]', S_Angles(4), kj4);

      fprintf('\n\nConvert to Servo Rotation Angles....');
      S_Rot_Angles  =  Convert_to_servo_rotation_angles (S_Angles);
      fprintf('\n[Angle1  Angle2  Angle3  Angle4] = [%8.4f\t%8.4f\t%8.4f\t%8.4f]', S_Rot_Angles);


    elseif (menu_selection == 2)

      fprintf('\nEnter 4 actuation Angles ... \t');
      fprintf('Servo Angles Range: 30 - 170 Degrees (140 Degree Range)');
      S_Angles(1) = input('\n\nServo Angle 1 : ');
      S_Angles(2) = input('\nServo Angle 2 : ');
      S_Angles(3) = input('\nServo Angle 3 : ');
      S_Angles(4) = input('\nServo Angle 4 : ');

      fprintf('\n\nConvert to Coordinate System Angles....');
      S_Angles  =  Convert_to_coordinate_system_angles (S_Angles);
      fprintf('\n[Angle1  Angle2  Angle3  Angle4] = [%8.4f\t%8.4f\t%8.4f\t%8.4f]', S_Angles);

     [coee(1), coee(2), coee(3), kj1, kj2, kj3, kj4] = Forward_Kinematics(S_Angles(1), S_Angles(2), S_Angles(3),
S_Angles(4));

   %Forward_Kinematics
      fprintf('\n\n[x0  y0  z0] = [%8.4f\t%8.4f\t%8.4f]', coee);
      fprintf('\nAngle1 = %8.4f\t\t[x1 y1 z1] = [%8.4f\t%8.4f\t%8.4f]',S_Angles(1) , kj1);
```

```
        fprintf('\nAngle2 = %8.4f\t\t[x2 y2 z2] = [%8.4f\t%8.4f\t%8.4f]', S_Angles(2), kj2);
        fprintf('\nAngle3 = %8.4f\t\t[x3 y3 z3] = [%8.4f\t%8.4f\t%8.4f]', S_Angles(3), kj3);
        fprintf('\nAngle4 = %8.4f\t\t[x4 y4 z4] = [%8.4f\t%8.4f\t%8.4f]', S_Angles(4), kj4);

        fprintf('\n\nChecking solution with Inverse Kinematics...');
        [S_Angles(1), S_Angles(2), S_Angles(3), S_Angles(4), kj1, kj2, kj3, kj4] = Inverse_Kinematics(coee(1), coee(2),
coee(3));
        fprintf('\n\nAngle1 = %8.4f\t\t[x1 y1 z1] = [%8.4f\t%8.4f\t%8.4f]',S_Angles(1) , kj1);
        fprintf('\nAngle2 = %8.4f\t\t[x2 y2 z2] = [%8.4f\t%8.4f\t%8.4f]', S_Angles(2), kj2);
        fprintf('\nAngle3 = %8.4f\t\t[x3 y3 z3] = [%8.4f\t%8.4f\t%8.4f]', S_Angles(3), kj3);
        fprintf('\nAngle4 = %8.4f\t\t[x4 y4 z4] = [%8.4f\t%8.4f\t%8.4f]', S_Angles(4), kj4);
        fprintf('\n\n[x0  y0  z0] = [%8.4f\t%8.4f\t%8.4f]', coee);


    elseif (menu_selection == 3)

        vf = 500;
        va = 1;
        vp = [0, pi/2, pi];
        time = 2/500;
        dt = 2/(500*99);

        fprintf('\nFrequency      (Hz)     : %9.4f', vf);
        fprintf('\nAmplitude      (Degrees): %9.4f', va);

        fprintf('\nSimulation Time (S)     : %9.4f', time);

        %Call vibration function .... Vibration (vf, va, vp, time, dt)
        Vibration (vf, va, vp, time, dt);

    end

end
```

CONVERT TO SERVO ROTATION ANGLES

```
function [n] = Convert_to_servo_rotation_angles (s)
%Convert to servo angles

n(1) =  mod(405 - abs(s(1)), 360);          %from 45 to -135, clockwise,
                                            %convert to 0 to 180

n(2) =  s(2) - 135;                     %from 135 to 315, anti-clockwise,
                                            %convert to 0 to 180

n(3) =  s(3) - 135;                     %from 135 to 315, anti-clockwise,
                                            %convert to 0 to 180

n(4) =  mod(405 - abs(s(4)), 360);          %from 45 to -135, clockwise,
                                            %convert to 0 to 180
```

INVERSE KINEMATICS

```
function [servo_angle_1, servo_angle_2, servo_angle_3, servo_angle_4, kj1, kj2, kj3, kj4] = Inverse_Kinematics(x0, y0, z0)
%function [servo_angle_1, servo_angle_2, servo_angle_3, servo_angle_4, kj1, kj2, kj3, kj4] = Inverse_Kinematics(x0, y0, z0)
% This Program will solve the Inverse Kinematic equations of the modified Delta
% Robot for points in space.
% Coordinates for the centre of the end effector ... coee = [x0 y0 z0]

  coee = [0 0 0];


  coee(1) = x0;
  coee(2) = y0;
  coee(3) = z0;


if (coee(3) ~= 0) % Proceed only if z0 is not equal to 0

  %coordinates of ankle joint 1
  coaj1 = [(coee(1)+4.45), coee(2), coee(3)];

  %coordinates of ankle joint 2
```

```
coaj2 = [coee(1), (coee(2)-4.45), coee(3)];

%coordinates of ankle joint 3
coaj3 = [(coee(1)-4.45), coee(2), coee(3)];

%coordinates of ankle joint 4
coaj4 = [coee(1), (coee(2)+4.45) coee(3)];

%coordinates of thigh joint 1
cotj1 = [10 0 0];

%coordinates of thigh joint 2
cotj2 = [0 -10 0];

%coordinates of thigh joint 3
cotj3 = [-10 0 0];

%coordinates of thigh joint 4
cotj4 = [0 10 0];

%All equations have been solved --> 1 ; if any cannot be resolved --> 0
all_equations_solved = 1;

%fprintf('\nSolving the knee coordinates for each leg ...\n')
%LEG 1
%coaj1 = [x0+4.45 y0 z0];
% c1 = (5 - x0)/z0
c1 = (5.55 - coee(1))/coee(3);

% c2 = (x0^2 + y0^2 + z0^2 + 10x0 - 200)/2z0
c2 = (coee(1)^2 + coee(2)^2 + coee(3)^2 + 8.9*coee(1) - 300.607)/(2*coee(3)); %was 200 instead of 295.41

temp1 = 100 - 20*c1*c2 - c2^2;
%if temp1 < 0 then we have no real solutions


if (temp1 < 0) | (all_equations_solved == 0)
   %Place and escape sequence here
   all_equations_solved = 0;
   servo_angle_1 = -400;
   kj1 = [-400, -400, -400];

else
   %Solution set 1
   % cokj1s1 - coordinates of knee joint 1 solution 1
   % cokj1s1 = [x1 y1 z1]
   cokj1s1 = [0 0 0];

   % x1 = (-c1c2 + 10 + (temp1)^0.5)/(c1^2 + 1)
   cokj1s1(1) = (-c1*c2 + 10 + (temp1)^0.5)/(c1^2 + 1);

   % z1 = c1x1 + c2
   cokj1s1(3) = c1*cokj1s1(1) + c2;


   %Solution set 2
   % cokj1s2 - coordinates of knee joint 1 solution 2
   % cokj1s2 = [x1 y1 z1]
   cokj1s2 = [0 0 0];

   % x1 = (-c1c2 + 10 - (temp1)^0.5)/(c1^2 + 1)
   cokj1s2(1) = (-c1*c2 + 10 - (temp1)^0.5)/(c1^2 + 1);

   % z1 = c1x1 + c2
   cokj1s2(3) = c1*cokj1s2(1) + c2;


   [servo_angle_1, kj1] = Determine_correct_solution(cokj1s1, cokj1s2, coaj1, cotj1);  % (s1, s2, a, t)

end


%LEG 2
%coordinates of ankle joint 2 -- coaj2 = [x0 y0-5 z0]
%coordinates of ankle joint 2
```

```
% c1l2 = (-5 - y0)/z0
c1 = (-5.55 - coee(2))/coee(3);

% c2l1 = (x0^2 + y0^2 + z0^2 - 10y0 - 200)/2z0
c2 = (coee(1)^2 + coee(2)^2 + coee(3)^2 - 8.9*coee(2) - 300.607)/(2*coee(3));  %was 200 instead of 295.41

temp2 = 100 + 20*c1*c2 - c2^2;
%if temp2 < 0 then we have no real solutions

if (temp2 < 0) | (all_equations_solved == 0)
   %Place and escape sequence here
   all_equations_solved = 0;
   servo_angle_2 = -400;
   kj2 = [-400, -400, -400];

else
   %Solution set 1
   % cokj2s1 - coordinates of knee joint 2 solution 1
   % cokj2s1 = [x2 y2 z2]
   cokj2s1 = [0 0 0];

   % y2 = (-c1c2 - 10 + (temp1)^0.5)/(c1^2 + 1)
   cokj2s1(2) = (-c1*c2 - 10 + (temp2)^0.5)/(c1^2 + 1);

   % z2 = c1y2 + c2
   cokj2s1(3) = c1*cokj2s1(2) + c2;


   %Solution set 2
   % cokj2s2 - coordinates of knee joint 2 solution 2
   % cokj2s2 = [x2 y2 z2]
   cokj2s2 = [0 0 0];

   % y2 = (-c1c2 - 10 - (temp1)^0.5)/(c1^2 + 1)
   cokj2s2(2) = (-c1*c2 - 10 - (temp2)^0.5)/(c1^2 + 1);

   % z2 = c1y2 + c2
   cokj2s2(3) = c1*cokj2s2(2) + c2;


   [servo_angle_2, kj2] = Determine_correct_solution(cokj2s1, cokj2s2, coaj2, cotj2);  % (s1, s2, a, t)

end


%LEG 3
%coaj3 = [x0-5 y0 z0];
% c1 = (-5 - x0)/z0
c1 = (-5.55 - coee(1))/coee(3);

% c2 = (x0^2 + y0^2 + z0^2 + 10x0 - 200)/2z0
c2 = (coee(1)^2 + coee(2)^2 + coee(3)^2 - 8.9*coee(1) - 300.607)/(2*coee(3));  %was 200 instead of 295.41

temp3 = 100 + 20*c1*c2 - c2^2;
%if temp3 < 0 then we have no real solutions

if (temp3 < 0) | (all_equations_solved == 0)
   %Place and escape sequence here
   all_equations_solved = 0;
   servo_angle_3 = -400;
   kj3 = [-400, -400, -400];

else
   %Solution set 1
   % cokj3s1 - coordinates of knee joint 3 solution 1
   % cokj3s1 = [x3 y3 z3]
   cokj3s1 = [0 0 0];

   % x3 = (-c1c2 + 10 + (temp1)^0.5)/(c1^2 + 1)
   cokj3s1(1) = (-c1*c2 - 10 + (temp3)^0.5)/(c1^2 + 1);

   % z3 = c1x3 + c2
   cokj3s1(3) = c1*cokj3s1(1) + c2;
```

```
    %Solution set 2
    % cokj3s2 - coordinates of knee joint 3 solution 2
    % cokj3s2 = [x3 y3 z3]
    cokj3s2 = [0 0 0];

    % x3 = (-c1c2 + 10 - (temp1)^0.5)/(c1^2 + 1)
    cokj3s2(1) = (-c1*c2 - 10 - (temp3)^0.5)/(c1^2 + 1);

    % z3 = c1x3 + c2
    cokj3s2(3) = c1*cokj3s2(1) + c2;

    [servo_angle_3, kj3] = Determine_correct_solution(cokj3s1, cokj3s2, coaj3, cotj3);   % (s1, s2, a, t)

  end


  %LEG 4
  %coordinates of ankle joint 4 -- coaj4 = [x0 y0+5 z0]
  %coordinates of ankle joint 4

  % c1 = (5 - y0)/z0
  c1 = (5.55 - coee(2))/coee(3);

  % c2 = (x0^2 + y0^2 + z0^2 + 10y0 - 200)/2z0
  c2 = (coee(1)^2 + coee(2)^2 + coee(3)^2 + 8.9*coee(2) - 300.607)/(2*coee(3));  %was 200 instead of 295.41

  temp4 = 100 - 20*c1*c2 - c2^2;
  %if temp4 < 0 then we have no real solutions

  if (temp4 < 0) | (all_equations_solved == 0)
    %Place and escape sequence here
    all_equations_solved = 0;
    servo_angle_4 = -400;
    kj4 = [-400, -400, -400];

  else
    %Solution set 1
    % cokj4s1 - coordinates of knee joint 4 solution 1
    % cokj4s1 = [x4 y4 z4]
    cokj4s1 = [0 0 0];

    % y4 = (-c1c2 + 10 + (temp1)^0.5)/(c1^2 + 1)
    cokj4s1(2) = (-c1*c2 + 10 + (temp4)^0.5)/(c1^2 + 1);

    % z4 = c1y4 + c2
    cokj4s1(3) = c1*cokj4s1(2) + c2;


    %Solution set 2
    % cokj4s2 - coordinates of knee joint 4 solution 2
    % cokj4s2 = [x4 y4 z4]
    cokj4s2 = [0 0 0];

    % y4 = (-c1c2 - 10 - (temp1)^0.5)/(c1^2 + 1)
    cokj4s2(2) = (-c1*c2 + 10 - (temp4)^0.5)/(c1^2 + 1);

    % z4 = c1y4 + c2
    cokj4s2(3) = c1*cokj4s2(2) + c2;

    [servo_angle_4, kj4] = Determine_correct_solution(cokj4s1, cokj4s2, coaj4, cotj4);   % (s1, s2, a, t)

  end

else
  % There is a divide by 0...coordinates cannot be found
  servo_angle_1 = -400;
  kj1 = [-400, -400, -400];

  servo_angle_2 = -400;
  kj2 = [-400, -400, -400];

  servo_angle_3 = -400;
  kj3 = [-400, -400, -400];
  servo_angle_4 = -400;
  kj4 = [-400, -400, -400];

end
```

DETERMINE CORRECT SOLUTION

```
function [angle, K_coordinates] = Determine_correct_solution(s1, s2, a, t)
%Determine Correct Angle

%s1 and s2 are coordinates sets for the knee coordinates, both solutions
%s1 = (kx1, ky1, kz1)
%s2 = (kx2, ky2, kz2)

%a is the ankle joint
%a = (ax, ay, az)

%t is the thigh joint
%t = (tx, ty, tz)

rad2deg = 180/pi;


if ( t(1) == 10 ) % Leg 1

  %  function [angle] = Get_upper_leg_angle(z1, z2, r1, r2)
  ang1 = Get_upper_leg_angle(s1(3), 0, s1(1), 10);   %atan(s1(3)/(s1(1) - 10))*rad2deg;
  ang2 = Get_upper_leg_angle(s2(3), 0, s2(1), 10);   %atan(s2(3)/(s2(1) - 10))*rad2deg;

  if (t(1) ~= a(1)) % In this case the Denominator goes to 0
      % Gradient of line A1T1 = DZ/DX
    m = (t(3) - a(3))/(t(1) - a(1));
    c = -10*m;
    z1 = m*s1(1) + c;  %z at x = s1(1)
    z2 = m*s2(1) + c; %z at x = s2(1)

    %Now Test gradient of A1T1
    if (m > 0)
    % z of Knee coordinates must be less than z of A1T1 at
    % corresponding x
       if (s1(3) < z1)
          angle = ang1;
          K_coordinates = s1;
       elseif (s2(3) < z2)
          angle = ang2;
          K_coordinates = s2;
       else
       %Problems
          angle = -400;
          K_coordinates = [-400, -400, -400];
       end

    elseif (m < 0)
    % z of Knee coordinatesz1 must be more than z of A1T1 at
    % corresponding x
       if (s1(3) > z1)
          angle = ang1;
          K_coordinates = s1;
       elseif (s2(3) > z2)
          angle = ang2;
          K_coordinates = s2;
       else
       %Problems
          angle = -400;
          K_coordinates = [-400, -400, -400];
       end

    end

  else % t(1) = a(1)
    if ( abs(s1(1)) > abs(s2(1)) )
       angle = ang1;
       K_coordinates = s1;
    elseif ( abs(s1(1)) < abs(s2(1)) )
```

```
      angle = ang2;
      K_coordinates = s2;
    else
      %Problems
    end

  end

%**** LEG 3 *****************
elseif ( t(1) == -10 ) % Leg 3
  ang1 = Get_upper_leg_angle(s1(3), 0, s1(1), -10);      %atan(s1(3)/(s1(1) + 10))*rad2deg;
  ang2 = Get_upper_leg_angle(s2(3), 0, s2(1), -10);      %atan(s2(3)/(s2(1) + 10))*rad2deg;

  if (t(1) ~= a(1)) % In this case the Denominator goes to 0
     % Gradient of line A1T1 = DZ/DX
    m = (t(3) - a(3))/(t(1) - a(1));
    c = 10*m;
    z1 = m*s1(1) + c;  %z at x = s1(1)
    z2 = m*s2(1) + c; %z at x = s2(1)

    %Now Test gradient of A1T1
    if (m > 0)
    % z of Knee coordinates must be less than z of A1T1 at
    % corresponding x
      if (s1(3) > z1)
        angle = ang1;
        K_coordinates = s1;
      elseif (s2(3) > z2)
        angle = ang2;
        K_coordinates = s2;
      else
      %Problems
        angle = -400;
        K_coordinates = [-400, -400, -400];
      end

    elseif (m < 0)
    % z of Knee coordinatesz1 must be more than z of A1T1 at
    % corresponding x
      if (s1(3) < z1)
        angle = ang1;
        K_coordinates = s1;
      elseif (s2(3) < z2)
        angle = ang2;
        K_coordinates = s2;
      else
      %Problems
        angle = -400;
        K_coordinates = [-400, -400, -400];
      end

    end

  else % t(1) = a(1)
    if ( abs(s1(1)) > abs(s2(1)) )
      angle = ang1;
      K_coordinates = s1;
    elseif ( abs(s1(1)) < abs(s2(1)) )
      angle = ang2;
      K_coordinates = s2;
    else
      %Problems
    end

  end

%**** LEG 2 *****************
elseif ( t(2) == -10 ) % Leg 2

  ang1 = Get_upper_leg_angle(s1(3), 0, s1(2), -10);      %atan(s1(3)/(s1(2) + 10))*rad2deg;
  ang2 = Get_upper_leg_angle(s2(3), 0, s2(2), -10);      %atan(s2(3)/(s2(2) + 10))*rad2deg;

  if (t(2) ~= a(2)) % In this case the Denominator goes to 0
     % Gradient of line A1T1 = DZ/DX
    m = (t(3) - a(3))/(t(2) - a(2));
    c = 10*m;
    z1 = m*s1(2) + c;  %z at y = s1(2)
    z2 = m*s2(2) + c; %z at y = s2(2)

    %Now Test gradient of A1T1
    if (m > 0)
    % z of Knee coordinates must be less than z of A1T1 at
    % corresponding y
      if (s1(3) > z1)
        angle = ang1;
        K_coordinates = s1;
      elseif (s2(3) > z2)
        angle = ang2;
        K_coordinates = s2;
      else
      %Problems
        angle = -400;
        K_coordinates = [-400, -400, -400];
      end


    elseif (m < 0)
    % z of Knee coordinatesz1 must be more than z of A1T1 at
    % corresponding y
      if (s1(3) < z1)
        angle = ang1;
        K_coordinates = s1;
      elseif (s2(3) < z2)
        angle = ang2;
        K_coordinates = s2;
      else
      %Problems
        angle = -400;
        K_coordinates = [-400, -400, -400];
      end

    end

  else % t(2) = a(2)
    if ( abs(s1(2)) > abs(s2(2)) )
      angle = ang1;
      K_coordinates = s1;
    elseif ( abs(s1(2)) < abs(s2(2)) )
      angle = ang2;
      K_coordinates = s2;
    else
      %Problems
    end

  end

%**** LEG 4  *****************
elseif ( t(2) == 10 ) % Leg 4

  ang1 = Get_upper_leg_angle(s1(3), 0, s1(2), 10);      %atan(s1(3)/(s1(2) - 10))*rad2deg;
  ang2 = Get_upper_leg_angle(s2(3), 0, s2(2), 10);      %atan(s2(3)/(s2(2) - 10))*rad2deg;

  if (t(2) ~= a(2)) % In this case the Denominator goes to 0
      % Gradient of line A4T4 = DZ/DY
    m = (t(3) - a(3))/(t(2) - a(2));
    c = -10*m;
    z1 = m*s1(2) + c;  %z at x = s1(1)
    z2 = m*s2(2) + c; %z at x = s2(1)

    %Now Test gradient of A1T1
    if (m > 0)
    % z of Knee coordinates must be less than z of A4T4 at
    % corresponding y
      if (s1(3) < z1)
        angle = ang1;
        K_coordinates = s1;
      elseif (s2(3) < z2)
        angle = ang2;
        K_coordinates = s2;
      else
      %Problems
        angle = -400;
        K_coordinates = [-400, -400, -400];
      end


    elseif (m < 0)
    % z of Knee coordinatesz1 must be more than z of A4T4 at
    % corresponding y
      if (s1(3) > z1)
        angle = ang1;
        K_coordinates = s1;
      elseif (s2(3) > z2)
        angle = ang2;
        K_coordinates = s2;
      else
      %Problems
        angle = -400;
        K_coordinates = [-400, -400, -400];
      end

    end

  else % t(2) = a(2)
    if ( abs(s1(2)) > abs(s2(2)) )
      angle = ang1;
      K_coordinates = s1;
    elseif ( abs(s1(2)) < abs(s2(2)) )
      angle = ang2;
      K_coordinates = s2;
    else
      %Problems
    end

  end

end
```

GET UPPER LEG ANGLES

```
function [angle] = Get_upper_leg_angle(z1, z2, r1, r2)

rad2deg = 180/pi;
%Correct Arctangent with angle transform for leg included
num = z1 - z2;
den = r1 - r2;

if (den ~= 0)

  angle = atan( num/den )*rad2deg;

  %Determine Correct Quadrant
  %First Quadrant
  if (num == 0) & (den > 0)
    angle = 0;
  elseif (num > 0) & (den > 0)
    %No change to angle

  %Second Quadrant - aTan is negative
  elseif (num > 0) & (den < 0)
    angle = angle - 180; %force angle to go from -180 to -270

  elseif (num == 0) & (den < 0)
    angle = -180;

  %Third Quadrant - aTan is positive
  elseif (num < 0) & (den < 0)
    angle = angle - 180;  % Angle must range from -90 to -180

  %Fourth Quadrant - aTan is negative
  elseif (num < 0) & (den > 0)
    angle = angle;  % No change
```

```
   end

else % den = 0
   if (num > 0)
      angle = 90;
   elseif (num < 0)
      angle = -90;
   else
      %Problems - cannot determine angle as num = 0 and den = 0
   end
end


angle = mod(angle, 360);
% fprintf('\n After Transform Angle = %8.4f', angle);
```

FORWARD KINEMATICS

```
function [x0, y0, z0, kj1, kj2, kj3, kj4] = Forward_Kinematics(a1, a2, a3, a4)
% Solving for x0, y0 and z0 ... the forward kinematics of the FlexPicker
% Robot

kj1 = [0 0 0];
kj2 = [0 0 0];
kj3 = [0 0 0];
kj4 = [0 0 0];
coee = [0 0 0];

% ********** New coordinate system Funtion
%function [k1, k2, k3, k4] = Solve_for_knee_coordinates_fk (ang1, ang2, ang3, ang4)
[kj1, kj2, kj3, kj4] = Solve_for_knee_coordinates_fk (a1, a2, a3, a4);

c1 = 8.9 - 2*kj1(1);
c2 = -8.9*kj1(1) + kj1(1)^2 + kj1(3)^2;

c3 = -8.9 - 2*kj3(1);
c4 = 8.9*kj3(1) + kj3(1)^2 + kj3(3)^2;

c5 = -8.9 - 2*kj2(2);
c6 = 8.9*kj2(2) + kj2(2)^2 + kj2(3)^2;

c7 = 8.9 - 2*kj4(2);
c8 = -8.9*kj4(2) + kj4(2)^2 + kj4(3)^2;

m1 = (2*(kj1(3) - kj3(3)))/(c1 - c3);
n1 = (c4 - c2)/(c1 - c3);

m2 = (2*(kj2(3) - kj4(3)))/(c5 - c7);
n2 = (c8 - c6)/(c5 - c7);

a = m1^2 + m2^2 + 1;
b = 2*m1*n1 + 8.9*m1 - 2*kj1(1)*m1 + 2*m2*n2 - 2*kj1(3);
c = n1^2 + 8.9*n1 - 2*kj1(1)*n1 - 8.9*kj1(1)+ kj1(1)^2 + n2^2 + kj1(3)^2 - 300.607; %200;

z0s1 = (-b + (b^2 - 4*a*c)^.5)/(2*a);
x0s1 = (2*(kj1(3) - kj3(3))*z0s1 + c4 - c2)/(c1 - c3);
y0s1 = (2*(kj2(3) - kj4(3))*z0s1 + c8 - c6)/(c5 - c7);

z0s2 = (-b - (b^2 - 4*a*c)^.5)/(2*a);
x0s2 = (2*(kj1(3) - kj3(3))*z0s2 + c4 - c2)/(c1 - c3);
y0s2 = (2*(kj2(3) - kj4(3))*z0s2 + c8 - c6)/(c5 - c7);


if (z0s2 <= z0s1)
   coee(1) = x0s2;
   coee(2) = y0s2;
   coee(3) = z0s2;

elseif (z0s1 <= z0s2)
   coee(1) = x0s1;
   coee(2) = y0s1;
   coee(3) = z0s1;
```

```
end

x0 = coee(1);
y0 = coee(2);
z0 = coee(3);
```

SOLVE FOR KNEE COORDINATES

```
function [k1, k2, k3, k4] = Solve_for_knee_coordinates_fk (ang1, ang2, ang3, ang4)
%Getting Coordinates of ankle joints for forward kinematics
%Using new coordinate system

deg2rad = pi/180;

    %Determine coordinates from angle for leg 1 of delta modification
    angle = deg2rad*ang1;  %Convert to radians
    y1 = 0;

    %fprintf('\nAngle1 = %6.2f',ang1);

    % [x1s1 z1s1 x1s2 z1s2] = solving_for_x1_and_z1(ang1, x_low, x_high, z_low, z_high)
    if (ang1 > 270)&(ang1 < 360) % 0 > z > -10  and  20 > x > 10
       [x1 z1] = solving_for_x1_and_z1(angle, 10, 20, -10, 0);

    elseif (ang1 == 270) % z = -10 and x = 10
       x1 = 10;
       z1 = -10;

    elseif (ang1 > 180)&(ang1 < 270) % 0 > z > -10 and 10 > x > 0
       [x1 z1] = solving_for_x1_and_z1(angle, 0, 10, -10, 0);

    elseif (ang1 == 180) % z = 0 and x = 0
       x1 = 0;
       z1 = 0;

    elseif (ang1 > 90)&(ang1 < 180) % 0 < z < 10 and 10 > x > 0
       [x1 z1] = solving_for_x1_and_z1(angle, 0, 10, 0, 10);

    elseif (ang1 == 90) % z = 10 and x = 10
       x1 = 10;
       z1 = 10;

    elseif (ang1 < 90)&(ang1 > 0) % 0 < z < 10  and 20 > x > 10
       [x1 z1] = solving_for_x1_and_z1(angle, 10, 20, 0, 10);

    elseif (ang1 == 0) % z = 0 and x = 0
       x1 = 20;
       z1 = 0;

    else
       fprintf('\nError with angle, coordinates for leg 1 cannot be resolved.')
       x1 = -400;
       y1 = -400;
       z1 = -400;
    end
    k1 = [x1, y1, z1];

    %*****************************************************************
    %Determine coordinates from angle for leg 2 of delta modification
    angle = deg2rad*ang2;  %Convert to radians
    x2 = 0;

    %fprintf('\n\nAngle2 = %6.2f',ang2);

    % [y2s1 z2s1 y2s2 z2s2] = solving_for_y2_and_z2(ang2, y_low, y_high, z_low, z_high)
    if (ang2 > 270)&(ang2 < 360) % 0 > z > -10  and  0 > y > -10
       [y2 z2] = solving_for_y2_and_z2(angle, -10, 0, -10, 0);

    elseif (ang2 == 270) % z = -10 and y = -10
       y2 = -10;
       z2 = -10;
```

```
elseif (ang2 > 180)&(ang2 < 270) % 0 > z > -10 and -10 > y > -20
   [y2 z2] = solving_for_y2_and_z2(angle, -20, -10, -10, 0);

elseif (ang2 == 180) % z = 0 and y = -20
   y2 = -20;
   z2 = 0;

elseif (ang2 > 90)&(ang2 < 180) % 0 < z < 10 and -10 > y > -20
   [y2 z2] = solving_for_y2_and_z2(angle, -20, -10, 0, 10);

elseif (ang2 == 90) % z = 10 and y = -10
   y2 = -10;
   z2 = 10;

elseif (ang2 < 90)&(ang2 > 0) % 0 < z < 10  and 0 > y > -10
   [y2 z2] = solving_for_y2_and_z2(angle, -10, 0, 0, 10);

elseif (ang2 == 0) % z = 0 and y = 0
   y2 = 0;
   z2 = 0;

else
   fprintf('\nError with angle, coordinates for leg 2 cannot be resolved.')
   x2 = -400;
   y2 = -400;
   z2 = -400;

end
k2 = [x2, y2, z2];


%***************************************************************
%Determine coordinates from angle for leg 3 of delta modification
angle = deg2rad*ang3;  %Convert to radians
y3 = 0;

%fprintf('\n\nAngle3 = %6.2f',ang3);

% [x3s1 z3s1 x3s2 z3s2] = solving_for_x3_and_z3(ang3, x_low, x_high, z_low, z_high)
if (ang3 > 270)&(ang3 < 360) % 0 > z > -10  and  0 > x > -10
   [x3 z3] = solving_for_x3_and_z3(angle, -10, 0, -10, 0);

elseif (ang3 == 270) % z = -10 and x = -10
   x3 = -10;
   z3 = -10;

elseif (ang3 > 180)&(ang3 < 270) % 0 > z > -10 and -10 > x > -20
   [x3 z3] = solving_for_x3_and_z3(angle, -20, -10, -10, 0);

elseif (ang3 == 180) % z = 0 and x = -20
   x3 = -20;
   z3 = 0;

elseif (ang3 > 90)&(ang3 < 180) % 0 < z < 10 and -10 > x > -20
   [x3 z3] = solving_for_x3_and_z3(angle, -20, -10, 0, 10);

elseif (ang3 == 90) % z = 10 and x = -10
   x3 = -10;
   z3 = 10;

elseif (ang3 < 90)&(ang3 > 0) % 0 < z < 10  and 0 > x > -10
   [x3 z3] = solving_for_x3_and_z3(angle, -10, 0, 0, 10);

elseif (ang3 == 0) % z = 0 and x = 0
   x3 = 0;
   z3 = 0;

else
   fprintf('\nError with angle, coordinates for leg 3 cannot be resolved.')
   x3 = -400;
   y3 = -400;
   z3 = -400;

end
k3 = [x3, y3, z3];
```

```
%****************************************************************
%Determine coordinates from angle for leg 4 of delta modification
angle = deg2rad*ang4;  %Convert to radians
x4 = 0;

%fprintf('\n\nAngle4 = %6.2f',ang4);

% [y4s1 z4s1 y4s2 z4s2] = solving_for_y4_and_z4(ang4, y_low, y_high, z_low, z_high)
if (ang4 > 270)&(ang4 < 360) % 0 > z > -10  and  20 > y > 10
   [y4 z4] = solving_for_y4_and_z4(angle, 10, 20, -10, 0);

elseif (ang4 == 270) % z = 10 and y = 10
   y4 = 10;
   z4 = -10;

elseif (ang4 > 180)&(ang4 < 270) % 0 > z > -10 and 10 > y > 0
   [y4 z4] = solving_for_y4_and_z4(angle, 0, 10, -10, 0);

elseif (ang4 == 180) % z = 0 and y = 0
   y4 = 0;
   z4 = 0;

elseif (ang4 > 90)&(ang4 < 180) % 0 < z < 10 and 10 > y > 0
   [y4 z4] = solving_for_y4_and_z4(angle, 0, 10, 0, 10);

elseif (ang4 == 90) % z = 10 and y = 10
   y4 = 10;
   z4 = 10;

elseif (ang4 < 90)&(ang4 > 0) % 0 < z < 10  and 20 > y > 10
   [y4 z4] = solving_for_y4_and_z4(angle, 10, 20, 0, 10);

elseif (ang4 == 0) % z = 0 and y = 20
   y4 = 20;
   z4 = 0;

else
   fprintf('\nError with angle, coordinates for leg 4 cannot be resolved.')
   x4 = -400;
   y4 = -400;
   z4 = -400;

end
k4 = [x4, y4, z4];
```

SOLVE FOR X1 AND Z1

```
function [x1 z1] = solving_for_x1_and_z1(ang1, x_limit_low, x_limit_high, z_limit_low, z_limit_high)
%Determine coordinates from angle for leg 1 of delta modification

    m1 = tan(ang1);
    x1s1 = 10 + (10/((1 + m1^2)^0.5));
    z1s1 = m1*(x1s1 - 10);

    x1s2 = 10 - (10/((1 + m1^2)^0.5));
    z1s2 = m1*(x1s2 - 10);

    %fprintf('[x_limit_low \tx_limit_high] = [%6.2f\t%6.2f]\n', x_limit_low, x_limit_high);
    %fprintf('[z_limit_low \tz_limit_high] = [%6.2f\t%6.2f]\n', z_limit_low, z_limit_high);
    %fprintf('[x1s1 z1s1 x1s2 z1s2] = [%6.2f\t%6.2f\t%6.2f\t%6.2f]\n', x1s1, z1s1, x1s2, z1s2);

    %check_solution1 = (x1s1 - 10)^2 + z1s1^2;
    %check_solution2 = (x1s2 - 10)^2 + z1s2^2;

    %fprintf('[check_solution1 check_solution2] = [%6.2f\t%6.2f]\n', check_solution1, check_solution2);

    % x_limit_low < x < x_limit_high  and  z_limit_low < z <  z_limit_high
    if (z_limit_low < z1s1)&(z1s1 < z_limit_high)&(x_limit_low < x1s1)&(x1s1 < x_limit_high)
       x1 = x1s1;
       z1 = z1s1;

    elseif (z_limit_low < z1s2)&(z1s2 < z_limit_high)&(x_limit_low < x1s2)&(x1s2 < x_limit_high)
```

```
        x1 = x1s2;
        z1 = z1s2;

    else
        fprintf('Coordinates for leg 1 cannot be resolved.\n')
        x1 = -400;
        z1 = -400;

    end
```

SOLVE FOR Y2 AND Z2

```
function [y2 z2] = solving_for_y2_and_z2(ang2, y_limit_low, y_limit_high, z_limit_low, z_limit_high)
%Determine coordinates from angle for leg 2 of delta modification

    m2 = tan(ang2);
    y2s1 = -10 + (10/((1 + m2^2)^0.5));
    z2s1 = m2*(y2s1 + 10);  %-m2  previously

    y2s2 = -10 - (10/((1 + m2^2)^0.5));
    z2s2 = m2*(y2s2 + 10);  %-m2  previously

    %fprintf('\n[y_limit_low \ty_limit_high] = [%6.2f\t%6.2f]', y_limit_low, y_limit_high);
    %fprintf('\n[z_limit_low \tz_limit_high] = [%6.2f\t%6.2f]', z_limit_low, z_limit_high);
    %fprintf('\n[y2s1 z2s1 y2s2 z2s2] = [%6.2f\t%6.2f\t%6.2f\t%6.2f]', y2s1, z2s1, y2s2, z2s2);

    %check_solution1 = (y2s1 + 10)^2 + z2s1^2;
    %check_solution2 = (y2s2 + 10)^2 + z2s2^2;

    %fprintf('\n[check_solution1 check_solution2] = [%6.2f\t%6.2f]', check_solution1, check_solution2);

    % y_limit_low < y < y_limit_high  and  z_limit_low < z <  z_limit_high
    if (z_limit_low < z2s1)&(z2s1 < z_limit_high)&(y_limit_low < y2s1)&(y2s1 < y_limit_high)
        y2 = y2s1;
        z2 = z2s1;

    elseif (z_limit_low < z2s2)&(z2s2 < z_limit_high)&(y_limit_low < y2s2)&(y2s2 < y_limit_high)
        y2 = y2s2;
        z2 = z2s2;

    else
        fprintf('\nCoordinates for leg 2 cannot be resolved.')
        y2 = -400;
        z2 = -400;

    end
```

SOLVE FOR X3 AND Z3

```
function [x3 z3] = solving_for_x3_and_z3(ang3, x_limit_low, x_limit_high, z_limit_low, z_limit_high)
%Determine coordinates from angle for leg 3 of delta modification

    m3 = tan(ang3);
    x3s1 = -10 + (10/((1 + m3^2)^0.5));
    z3s1 = m3*(x3s1 + 10);  %-m3  previously

    x3s2 = -10 - (10/((1 + m3^2)^0.5));
    z3s2 = m3*(x3s2 + 10);  %-m3  previously

    %fprintf('[x_limit_low \tx_limit_high] = [%6.2f\t%6.2f]\n', x_limit_low, x_limit_high);
    %fprintf('[z_limit_low \tz_limit_high] = [%6.2f\t%6.2f]\n', z_limit_low, z_limit_high);
    %fprintf('[x3s1 z3s1 x3s2 z3s2] = [%6.2f\t%6.2f\t%6.2f\t%6.2f]\n', x3s1, z3s1, x3s2, z3s2);

    %check_solution1 = (x3s1 + 10)^2 + z3s1^2
    %check_solution2 = (x3s2 + 10)^2 + z3s2^2

    %fprintf('[check_solution1 check_solution2] = [%6.2f\t%6.2f]\n', check_solution1, check_solution2);

    % x_limit_low < x < x_limit_high  and  z_limit_low < z <  z_limit_high
    if (z_limit_low < z3s1)&(z3s1 < z_limit_high)&(x_limit_low < x3s1)&(x3s1 < x_limit_high)
        x3 = x3s1;
```

```
           z3 = z3s1;

        elseif (z_limit_low < z3s2)&(z3s2 < z_limit_high)&(x_limit_low < x3s2)&(x3s2 < x_limit_high)
           x3 = x3s2;
           z3 = z3s2;

        else
           fprintf('Coordinates for leg 3 cannot be resolved.\n')
           x3 = -400;
           z3 = -400;

        end
```

SOLVE FOR Y4 AND Z4

```
function [y4 z4] = solving_for_y4_and_z4(ang4, y_limit_low, y_limit_high, z_limit_low, z_limit_high)
%Determine coordinates from angle for leg 4 of delta modification

        m4 = tan(ang4);
        y4s1 = 10 + (10/((1 + m4^2)^0.5));
        z4s1 = m4*(y4s1 - 10);

        y4s2 = 10 - (10/((1 + m4^2)^0.5));
        z4s2 = m4*(y4s2 - 10);

        %fprintf('[y_limit_low \ty_limit_high] = [%6.2f\t%6.2f]\n', y_limit_low, y_limit_high);
        %fprintf('[z_limit_low \tz_limit_high] = [%6.2f\t%6.2f]\n', z_limit_low, z_limit_high);
        %fprintf('[y4s1 z4s1 y4s2 z4s2] = [%6.2f\t%6.2f\t%6.2f\t%6.2f]\n', y4s1, z4s1, y4s2, z4s2);

        %check_solution1 = (y4s1 - 10)^2 + z4s1^2;
        %check_solution2 = (y4s2 - 10)^2 + z4s2^2;

        %fprintf('[check_solution1 check_solution2] = [%6.2f\t%6.2f]\n', check_solution1, check_solution2);

        % y_limit_low < y < y_limit_high  and  z_limit_low < z <  z_limit_high
        if (z_limit_low < z4s1)&(z4s1 < z_limit_high)&(y_limit_low < y4s1)&(y4s1 < y_limit_high)
           y4 = y4s1;
           z4 = z4s1;

        elseif (z_limit_low < z4s2)&(z4s2 < z_limit_high)&(y_limit_low < y4s2)&(y4s2 < y_limit_high)
           y4 = y4s2;
           z4 = z4s2;

        else
           fprintf('Coordinates for leg 4 cannot be resolved.\n')
           y4 = -400;
           z4 = -400;

        end
```

VIBRATION

```
function [] = Vibration (vf, va, vp, time, dt)

        fprintf('\n\nEnter end effector coordinates ... \t')
        fprintf('x Range: -7.5 to 7.5 \t\ty Range: -7.5 to 7.5 \t\tz Range: -22 to -12 ')

        coee(1) = input('\n\nx0 : ');
        coee(2) = input('\ny0 : ');
        coee(3) = input('\nz0 : ');

        fprintf('\n\nThe coordinates you have selected ... [%8.4f\t%8.4f\t%8.4f]', coee);
        [S_Angles(1), S_Angles(2), S_Angles(3), S_Angles(4), kj1, kj2, kj3, kj4] = Inverse_Kinematics(coee(1), coee(2),
coee(3));

        fprintf('\n\nAngle1 = %8.4f\t\t[x1 y1 z1] = [%8.4f\t%8.4f\t%8.4f]',S_Angles(1) , kj1);
        fprintf('\nAngle2 = %8.4f\t\t[x2 y2 z2] = [%8.4f\t%8.4f\t%8.4f]', S_Angles(2), kj2);
        fprintf('\nAngle3 = %8.4f\t\t[x3 y3 z3] = [%8.4f\t%8.4f\t%8.4f]', S_Angles(3), kj3);
        fprintf('\nAngle4 = %8.4f\t\t[x4 y4 z4] = [%8.4f\t%8.4f\t%8.4f]', S_Angles(4), kj4);

        t = zeros(1, 100);
        y1 = zeros(1, 100);
```

```
y2 = zeros(1, 100);
y3 = zeros(1, 100);
y4 = zeros(1, 100);
x0 = zeros(1, 100);
y0 = zeros(1, 100);
z0 = zeros(1, 100);

x0_min = coee(1);
x0_max = coee(1);

y0_min = coee(2);
y0_max = coee(2);

z0_min = coee(3);
z0_max = coee(3);

fprintf('\n\nConvert to Servo Rotation Angles....');
S_Rot_Angles = Convert_to_servo_rotation_angles (S_Angles);
fprintf('\n[Angle1  Angle2  Angle3  Angle4] = [%8.4f\t%8.4f\t%8.4f\t%8.4f]', S_Rot_Angles);


for m = 2 : 2 : 2   % Was 6 to get more phase differences
  %For a Phase difference of vp1 = 0 radians
  k = m/2;

  for i = 1 : 100
    t(i) =  (i - 1)*dt;
    y1(i) = S_Rot_Angles(1) + va*sin (2*pi*vf*t(i));
    y2(i) = S_Rot_Angles(2) + va*sin (2*pi*vf*t(i) + vp(k));
    y3(i) = S_Rot_Angles(3) + va*sin (2*pi*vf*t(i));
    y4(i) = S_Rot_Angles(4) + va*sin (2*pi*vf*t(i) + vp(k));

    a1 = S_Angles(1) + va*sin (2*pi*vf*t(i));
    a2 = S_Angles(2) + va*sin (2*pi*vf*t(i) + vp(k));
    a3 = S_Angles(3) + va*sin (2*pi*vf*t(i));
    a4 = S_Angles(4) + va*sin (2*pi*vf*t(i) + vp(k));

    [x0(i), y0(i), z0(i), kj1, kj2, kj3, kj4] = Forward_Kinematics(a1, a2, a3, a4); %Forward_Kinematics

    if (x0(i) > x0_max)
      x0_max = x0(i);
    elseif (x0(i) < x0_min)
      x0_min = x0(i);
    end

    if (y0(i) > y0_max)
      y0_max = y0(i);
    elseif (y0(i) < y0_min)
      y0_min = y0(i);
    end

    if (z0(i) > z0_max)
      z0_max = z0(i);
    elseif (z0(i) < z0_min)
      z0_min = z0(i);
    end

  end

  if (coee(1) - x0_min < 0.0000001)
    x0_min = coee(1) - 0.0000001;
  end

  if (x0_max - coee(1) < 0.0000001)
    x0_max = coee(1) + 0.0000001;
  end

  if (coee(2) - y0_min < 0.0000001)
    y0_min = coee(2) - 0.0000001;
  end

  if (y0_max - coee(2) < 0.0000001)
    y0_max = coee(2) + 0.0000001;
  end
```

```
    if (coee(3) - z0_min < 0.0000001)
       z0_min = coee(3) - 0.0000001;
    end

    if (z0_max - coee(3) < 0.0000001)
       z0_max = coee(3) + 0.0000001;
    end

    fprintf('\n\n[x0_min  x0_max] = [%14.10f \t %14.10f]\t\tx0_max - x0_min = %14.10f', x0_min, x0_max, (x0_max -
x0_min));
    fprintf('\n[y0_min  y0_max] = [%14.10f \t %14.10f]\t\ty0_max - y0_min = %14.10f', y0_min, y0_max, (y0_max -
y0_min));
    fprintf('\n[z0_min  z0_max] = [%14.10f \t %14.10f]\t\tz0_max - z0_min = %14.10f', z0_min, z0_max, (z0_max -
z0_min));

    %Now that we have the angles we may add the vibration to them
    Figure(m);
    clf;
    subplot(2,2,1)
    hold on;

    plot(t, y1, '-b', 'LineWidth',2);
    ymin = S_Rot_Angles(1) - 1.5*va;
    ymax = S_Rot_Angles(1) + 1.5*va;
    axis([0, time, ymin, ymax]);
    grid on;
    title('a. Servo 1 Vibration')
    xlabel('Time (s)')
    ylabel('Angle (Degrees)')

    subplot(2,2,2)
    plot(t, y2, '-b','LineWidth',2);
    ymin = S_Rot_Angles(2) - 1.5*va;
    ymax = S_Rot_Angles(2) + 1.5*va;
    axis([0, time, ymin, ymax]);
    grid on;
    title('b. Servo 2 Vibration')
    xlabel('Time (s)')
    ylabel('Angle (Degrees)')

    subplot(2,2,3)
    plot(t, y3, '-b', 'LineWidth',2);
    ymin = S_Rot_Angles(3) - 1.5*va;
    ymax = S_Rot_Angles(3) + 1.5*va;
    axis([0, time, ymin, ymax]);
    grid on;
    title('c. Servo 3 Vibration')
    xlabel('Time (s)')
    ylabel('Angle (Degrees)')

    subplot(2,2,4)
    plot(t, y4, '-b','LineWidth',2);
    ymin = S_Rot_Angles(4) - 1.5*va;
    ymax = S_Rot_Angles(4) + 1.5*va;
    axis([0, time, ymin, ymax]);
    grid on;
    title('d. Servo 4 Vibration')
    xlabel('Time (s)')
    ylabel('Angle (Degrees)')
    hold off;

    Figure(m+1);
    clf;
    subplot(3,1,1)
    hold on;
    plot(t, x0, '-b', 'LineWidth',2);
    axis([0, time, x0_min , x0_max ]);
    title('a. X0 Vibration')
    grid on;
    ylabel('Position (cm)')

    subplot(3,1,2)
    plot(t, y0, '-b', 'LineWidth',2);
    axis([0, time, y0_min, y0_max]);
    grid on;
```

```
        title('b. Y0 Vibration')
        ylabel('Position (cm)')

        subplot(3,1,3)
        plot(t, z0, '-b', 'LineWidth',2);
        axis([0, time, z0_min, z0_max]);
        grid on;
        title('c. Z0 Vibration')
        xlabel('Time (s)')
        ylabel('Position (cm)')
        hold off;

    end
```

WORKSPACE CALCULATION

```
function [x1, y1, zmin, x2, y2, zmax, last_valid_x, last_valid_y, correspond_z] = Workspace_Coordinate_Calculation (m, n,
last_valid_x, last_valid_y, correspond_z)
%Workspace coordinate calculation


    zmax_global = 0;
    zmin_global = -25;
    xmin_global = -16;
    xmax_global = 16;
    ymin_global = -16;
    ymax_global = 16;

    actual_minimum_z_reached = 10;
    % Set it high so that we ensure we actually find the minimum

    % We want to find the actual minimum and actual maximum
    xmin = xmax_global;
    xmax = xmin_global;

    % We want to find the actual minimum and actual maximum
    ymin = ymax_global;
    ymax = ymin_global;

    zmin = zmin_global;
    zmax = zmax_global;

    S_Angles_min = [0 0 0 0];
    S_Angles_max = [0 0 0 0];

    zmin_found = 0; % reset ... searching for new min z
    zmax_found = 0; % reset ... searching for new max z

    xmin_found = 0; % reset ... searching for new min x
    xmax_found = 0; % reset ... searching for new max x

    ymin_found = 0; % reset ... searching for new min y
    ymax_found = 0; % reset ... searching for new max y

    x1 = m;
    x2 = m;
    y1 = n;
    y2 = n;

    while (zmin_found == 0)|(zmax_found == 0)

        % function [servo_angle_1, servo_angle_2, servo_angle_3, servo_angle_4, kj1, kj2, kj3, kj4] =
Inverse_Kinematics(x0, y0, z0)

        if (zmin_found == 0)
            [S_Angles_min(1), S_Angles_min(2), S_Angles_min(3), S_Angles_min(4), kj1_min, kj2_min, kj3_min, kj4_min] =
Inverse_Kinematics(m, n, zmin);

            if (S_Angles_min(1) == -400)|(S_Angles_min(2) == -400)|(S_Angles_min(3) == -400)|(S_Angles_min(4) == -400)
                zmin = zmin + 0.25;
                if (zmin >= zmax)
                    % Passed range
                    zmin_found = 1;
```

```
              zmin = correspond_z; %zmin cannot be found
              x1 = last_valid_x;
              y1 = last_valid_y;

            end
          else
            % Found zmin
            zmin_found = 1;
            last_valid_x = m;
            last_valid_y = n;
            correspond_z = zmin;
            % We have found a value here, now we need to check
            % whether the x and y values for the minimum and
            % maximum
            if (m > xmax)
              xmax = m;
            end

            if (m < xmin)
              xmin = m;
              last_valid_x = m;
            end

            if (n > ymax)
              ymax = n;
              last_valid_y = n;
            end

            if (n < ymin)
              ymin = n;
            end

            if (actual_minimum_z_reached > zmin)
              actual_minimum_z_reached = zmin;
            end

          end
        end

        if (zmax_found == 0)
          [S_Angles_max(1), S_Angles_max(2), S_Angles_max(3), S_Angles_max(4), kj1_max, kj2_max, kj3_max,
kj4_max] = Inverse_Kinematics(m, n, zmax);

          if (S_Angles_max(1) == -400)|(S_Angles_max(2) == -400)|(S_Angles_max(3) == -400)|(S_Angles_max(4) == -
400)
            zmax = zmax - 0.25;
            if (zmax <= zmin)
              % Passed range
              zmax_found = 1;
              zmax = correspond_z;
              x2 = last_valid_x;
              y2 = last_valid_y;
            end
          else
            % Found zmax
            zmax_found = 1;
            last_valid_x = m;
            last_valid_y = n;
            correspond_z = zmax;
            % We have found a value here, now we need to check
            % whether the x and y values for the minimum and
            % maximum
            if (m > xmax)
              xmax = m;
            end

            if (m < xmin)
              xmin = m;
            end

            if (n > ymax)
              ymax = n;
            end

            if (n < ymin)
```

```
                ymin = n;
            end
        end
    end
end
```

VISUAL BASIC CODE

```
Option Explicit
  ' Determining which servo values we are receiving
  Dim bool_all_servo_mvnt_comp As Boolean

  ' This variable will hold the incoming data
  Dim received_data As String

  ' End effector Coordinates
  Dim x0 As Single, y0 As Single, z0 As Single
  Dim x0_old As Single, y0_old As Single, z0_old As Single
  Dim z0_max As Single

  Dim attach_circle_to_mouse As Boolean


  ' Upper Legs of Servos
  Dim upper_leg(3) As leg

  ' End effector ankle joint coordinates
  Dim end_effector As EEAJ

  Dim r_angles As rotation_angles
  Dim byte_r_angles As rotation_angles ' 0 - 160 -> 0 - 255


Private Sub value_to_ascii(d As Integer)

' This routine sends the ascii characters of the correspondinge decimal digits
  Select Case d

    Case 0
      MSComm1.Output = "0"
    Case 1
      MSComm1.Output = "1"
    Case 2
      MSComm1.Output = "2"
    Case 3
      MSComm1.Output = "3"
    Case 4
      MSComm1.Output = "4"
    Case 5
      MSComm1.Output = "5"
    Case 6
      MSComm1.Output = "6"
    Case 7
      MSComm1.Output = "7"
    Case 8
      MSComm1.Output = "8"
    Case 9
      MSComm1.Output = "9"

  End Select

End Sub

Private Sub send_digits_of_PWM_value(temp As Integer)

  Dim digit As Integer
    "send Ten thousands digit
    'digit = temp \ 10000   ' Integer division yields, integer result
    'temp = temp - 10000 * digit
    'Call value_to_ascii(digit)

    " Send thousands digit of coordinate
    'digit = temp \ 1000   ' Integer division yields, integer result
    'temp = temp - 1000 * digit
    'Call value_to_ascii(digit)

  ' Send hundreds digit of coordinate
  digit = temp \ 100   ' Integer division yields, integer result
  temp = temp - 100 * digit
  Call value_to_ascii(digit)

  ' Send tens digit of coordinate
```

```
    digit = temp \ 10
    temp = temp - 10 * digit
    Call value_to_ascii(digit)

    ' Send units digit of coordinate
    digit = temp
    Call value_to_ascii(digit)

    ' Send completion character
    MSComm1.Output = "X"

End Sub


Private Sub btn_start_Click()

    If (btn_start.Caption = "Start Capturing Servo Feedback") Then
        btn_start.Caption = "Stop Capturing Servo Feedback"
        'Send start command...open commport first then send start signal
        If MSComm1.PortOpen = False Then
            MSComm1.PortOpen = True
            'Send start signal
            MSComm1.Output = "E"
            'Calculate Inverse Kinematics
        Call do_inverse_kinematics_calculation(0, 0, -15)

        End If
    Else
        btn_start.Caption = "Start Capturing Servo Feedback"
        'Send stop command
        If MSComm1.PortOpen = True Then
            'Send stop command then disable port
            MSComm1.Output = "G"
            MSComm1.PortOpen = False
        End If
    End If

End Sub

Private Sub Command1_Click()

    ' Displays the source dialog for the video control
    'If (ezVidCap1.HasDlgSource) Then
        'ezVidCap1.ShowDlgVideoSource
    'End If

    ' Toggles preview and visibility
    'If ezVidCap1.Preview = False Then

        'ezVidCap1.Preview = True
        'ezVidCap1.Visible = True
    'Else

        'ezVidCap1.Preview = False
        'ezVidCap1.Visible = False

    'End If

End Sub

Private Sub Command2_Click()

     If MSComm1.PortOpen = True Then
        Text1.Text = Text1.Text & vbNewLine
        MSComm1.Output = "T"
    End If

End Sub

Private Sub Command3_Click()

    'Save angles to variables
    x0 = Val(txt_x0.Text)
    y0 = Val(txt_y0.Text)
    z0 = Val(txt_z0.Text)
    Text3.Text = "The Coordinates You have selected : x0 =" & x0 & "; y0 = " & y0 & "; z0 = " & z0
    Text3.Text = Text3.Text & vbNewLine
    Call Calculate_Inverse_Kinematics(x0, y0, z0, upper_leg(), end_effector, r_angles, Text3)

    Call Write_Upper_leg_values(Text3)

End Sub

Private Sub Write_Upper_leg_values(t As TextBox)

    Dim i As Integer

    'Format$(COKJS2(1), "Fixed")

    For i = 0 To 3 Step 1
        t.Text = t.Text & vbNewLine & "Upper Leg " & (i + 1) & " Angle: " & _
        Format$(upper_leg(i).angle, "Fixed") & "   Knee Coordinates: (" & _
        Format$(upper_leg(i).knee_coordinates.x, "Fixed") & _
        "; " & Format$(upper_leg(i).knee_coordinates.y, "Fixed") & "; " & _
        Format$(upper_leg(i).knee_coordinates.z, "Fixed") & ")"
    Next i

    t.Text = t.Text & vbNewLine

    For i = 0 To 3 Step 1
        t.Text = t.Text & vbNewLine & "End Effector Ankle Joints: AJ" & (i + 1) & _
        "- (" & Format$(end_effector.AJ(i).x, "Fixed") & "; " & _
        Format$(end_effector.AJ(i).y, "Fixed") & "; " & Format$(end_effector.AJ(i).z, "Fixed") & ")"
    Next i

End Sub

Private Sub Command4_Click()

    'x0 = 0
    'y0 = 0
    'z0 = -10

    'Calculate Inverse Kinematics
    Call do_inverse_kinematics_calculation(0, 0, -10)

    'Enable Timer and send data out port
    Timer1.Enabled = True

End Sub


Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)

    Text1.Text = KeyCode

    If (KeyCode = 90) Then
        bool_all_servo_mvnt_comp = True
    End If

End Sub

Private Sub Form_Load()

    'Initialize variables
    bool_all_servo_mvnt_comp = False

    'Set to empty string
    received_data = ""

    'Set Thigh Joint Coordinates - once only
    Call Set_Thigh_Joint_Coordinates(upper_leg())

    Call initialize_variables


    'Initialize x0, y0, z0
    z0_max = -10
```

```
  'x0 = 0
  'y0 = 0
  'z0 = z0_max - 5

  'Calculate Inverse Kinematics
  Call do_inverse_kinematics_calculation(0, 0, z0_max - 5)
  txt_z0.Text = z0

  'Scale Picture boxes
  Call scale_pic_boxes

  'initialize mouse control picture box
  Call draw_mouse_control_indicator(0, 0)

  attach_circle_to_mouse = False


End Sub

Private Sub Form_Unload(Cancel As Integer)

  'Close comport if it is open
  If MSComm1.PortOpen = True Then
    'Send stop command then disable port
    MSComm1.Output = "G"
    MSComm1.PortOpen = False
  End If

End Sub

Private Sub MSComm1_OnComm()

  'Holds character values
  Dim code As String
  code = MSComm1.Input

  '*************************************************
  'Remove Text1 control...just checking data received
  'Text1.Text = Text1.Text & code
  If (code = "X") Then
    Text1.Text = Text1.Text & "      "
  End If
  '*************************************************

  If (code = "z") Then
    'All servos have moved to their angles
    bool_all_servo_mvnt_comp = True
    MSComm1.Output = "z"   'Received completion character

  ElseIf (code = "X") Then


    'Reset string containing data
    received_data = ""

  ElseIf (code = "0") Or (code = "1") Or (code = "2") Or (code = "3") Or (code = "4") Or (code = "5") Or (code = "6") Or (code
= "7") Or (code = "8") Or (code = "9") Then
      ' Make sure code is a numerical ascii value - i.e. 0 to 9
      ' Appending a text string is similar to appending a text box
      received_data = received_data & code
  Else
      'Some code character, or unknown value
  End If

End Sub

Private Sub move_indicator_if_clicked(xt As Single, yt As Single)

  'Modify
  pic_mouse_control.Cls
  Call draw_mouse_control_indicator(0, 0)

  If opn_mouse.Value = True Then
  ' The first click attaches the circle to the mouse, once attached a second click will then
  ' release the mouse
```

```
        If (attach_circle_to_mouse = True) Then
            ' if its true we need to release the mouse
            attach_circle_to_mouse = False
            'Reset
            'x0 = 0
            'y0 = 0
            'z0 = z0_max - 5

            Call do_inverse_kinematics_calculation(0, 0, z0_max - 5)

            x0_old = 0
            y0_old = 0
            z0_old = z0

            txt_x0.Text = Format$(x0, "Fixed")
            txt_y0.Text = Format$(y0, "Fixed")
            txt_z0.Text = Format$(z0, "Fixed")

        Else
            ' if its false we need to check if the pointer is ontop of the circle
            If ((xt >= -0.1) And (xt <= 0.1)) Then
                If ((yt >= -0.1) And (yt <= 0.1)) Then
                    ' At this point the mouse pointer is over the circle
                    attach_circle_to_mouse = True
                End If
            End If
        End If
    End If


End Sub

Private Sub opn_demo_Click()

    opn_demo.Value = True

    Timer1.Enabled = True


End Sub

Private Sub opn_mouse_Click()

    Timer1.Enabled = False

End Sub

Private Sub pic_mouse_control_KeyDown(KeyCode As Integer, Shift As Integer)

    If (z0 <= z0_max) Or (z0 >= z0_max - 10) Then

        If (KeyCode = 38) Then
        'Up
            z0 = z0 + 0.1

        ElseIf (KeyCode = 40) Then
        'Down
            z0 = z0 - 0.1

        End If

    End If

    If (z0 > z0_max) Then
        z0 = z0_max
    ElseIf (z0 < z0_max - 10) Then
        z0 = z0_max - 10
    End If

    txt_z0.Text = Format$(z0, "Fixed")
    'draw data into picboxes
    Call draw_picbox_data(pic_horizontal_data, Text3, x0, y0, x0_old, y0_old, bool_all_servo_mvnt_comp)
    Call draw_picbox_data(pic_vertical_data, Text3, z0, y0, z0_old, y0_old, bool_all_servo_mvnt_comp)
```

```
End Sub

Private Sub pic_mouse_control_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)

  If Button = 1 Then  ' Left click
    Call move_indicator_if_clicked(x, y)
    'draw sensor grid data
    'draw data into picboxes
    Call draw_picbox_data(pic_horizontal_data, Text3, x0, y0, x0_old, y0_old, bool_all_servo_mvnt_comp)
    Call draw_picbox_data(pic_vertical_data, Text3, z0, y0, z0_old, y0_old, bool_all_servo_mvnt_comp)

  ElseIf Button = 2 Then 'Right Click
    'Only calculate inverse kinematics if we have the mouse attached to end of the cursor
    If (attach_circle_to_mouse = True) Then
      'Calculate Inverse Kinematics
      Call do_inverse_kinematics_calculation(x0, y0, z0)
      x0_old = x0
      y0_old = y0
      z0_old = z0

    End If
  Else

  End If

End Sub

Private Sub pic_mouse_control_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)

  If (attach_circle_to_mouse = True) Then
    pic_mouse_control.Cls

    x0 = x
    y0 = y

    Call draw_mouse_control_indicator(x, y)
    txt_x0.Text = Format$(x0, "Fixed")
    txt_y0.Text = Format$(y0, "Fixed")

    'draw sensor grid data
    'draw data into picboxes
    Call draw_picbox_data(pic_horizontal_data, Text3, x0, y0, x0_old, y0_old, bool_all_servo_mvnt_comp)
    Call draw_picbox_data(pic_vertical_data, Text3, z0, y0, z0_old, y0_old, bool_all_servo_mvnt_comp)

  End If

End Sub

Private Sub sld_servo1_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)

  Call servo1_control

End Sub


Private Sub sld_servo2_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)

  Call servo2_control

End Sub

Private Sub sld_servo3_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)

  Call servo3_control

End Sub


Private Sub sld_servo4_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)

  Call servo4_control

End Sub

Private Sub servo1_control()
```

```
    'Display value in textbox
    txt_servo1.Text = sld_servo1.Value

    If MSComm1.PortOpen = True Then
       'Let Controller know Servo 1's value is coming
       MSComm1.Output = "P"
       'Convert 8 bit value to 16 bit value, then send digits out of port
       send_digits_of_PWM_value (sld_servo1.Value)
       Text1.Text = Text1.Text & " **P" & convert_8_bit_slider_value_to_16_bit_PWM_value(sld_servo1.Value) & "P** "
    End If

End Sub

Private Sub servo2_control()

    'Display value in textbox
    txt_servo2.Text = sld_servo2.Value

    If MSComm1.PortOpen = True Then
       'Let Controller know Servo 1's value is coming
       MSComm1.Output = "Q"
       'Convert 8 bit value to 16 bit value, then send digits out of port
       send_digits_of_PWM_value (sld_servo2.Value)
       Text1.Text = Text1.Text & " **Q" & convert_8_bit_slider_value_to_16_bit_PWM_value(sld_servo2.Value) & "Q** "
    End If

End Sub

Private Sub servo3_control()

    txt_servo3.Text = sld_servo3.Value

    If MSComm1.PortOpen = True Then
       'Let Controller know Servo 1's value is coming
       MSComm1.Output = "R"
       'Convert 8 bit value to 16 bit value, then send digits out of port
       send_digits_of_PWM_value (sld_servo3.Value)
       Text1.Text = Text1.Text & " **R" & convert_8_bit_slider_value_to_16_bit_PWM_value(sld_servo3.Value) & "R** "
    End If

End Sub

Private Sub servo4_control()

    txt_servo4.Text = sld_servo4.Value

    If MSComm1.PortOpen = True Then
       'Let Controller know Servo 1's value is coming
       MSComm1.Output = "S"
       'Convert 8 bit value to 16 bit value, then send digits out of port
       send_digits_of_PWM_value (sld_servo4.Value)
       Text1.Text = Text1.Text & " **S" & convert_8_bit_slider_value_to_16_bit_PWM_value(sld_servo4.Value) & "S** "
    End If

    bool_all_servo_mvnt_comp = False

End Sub


Private Sub scale_pic_boxes()

    'Scale picboxes - Workspace 16x16x16
    pic_mouse_control.Scale (-7, 7)-(7, -7)

    'Boards are 16 cm
    pic_horizontal_data.Scale (-8, 8)-(8, -8)
    pic_vertical_data.Scale (z0_max + 3, 8)-(z0_max - 13, -8)  'z0_max - 5 + 8; z0_max - 5 - 8

End Sub

Private Sub draw_mouse_control_indicator(x As Single, y As Single)

    'Draw X axis, Y axis
    'Set line thickness, and style
    pic_mouse_control.FillColor = vbBlack
    pic_mouse_control.DrawStyle = 1    'Dash
    pic_mouse_control.Line (0, 7)-(0, -7) ' y axis
    pic_mouse_control.Line (-7, 0)-(7, 0) ' x axis

    pic_mouse_control.FillColor = vbBlue
    pic_mouse_control.DrawStyle = 0    'Solid
    pic_mouse_control.Line (x, 7)-(x, -7) '
    pic_mouse_control.Line (-7, y)-(7, y) '

    pic_mouse_control.Circle (x, y), 0.1

End Sub

Private Sub do_inverse_kinematics_calculation(x As Single, y As Single, z As Single)

    Dim xt As Single, yt As Single, zt As Single

    xt = x
    yt = y
    zt = z

    x0 = xt
    y0 = yt
    z0 = zt

    'Calculate Inverse Kinematics
    txt_x0.Text = Format$(x0, "Fixed")
    txt_y0.Text = Format$(y0, "Fixed")
    txt_z0.Text = Format$(z0, "Fixed")


    Text3.Text = "The Coordinates You have selected : x0 =" & Format$(x0, "Fixed") & _
    "; y0 = " & Format$(y0, "Fixed") & "; z0 = " & Format$(z0, "Fixed")
    Text3.Text = Text3.Text & vbNewLine
    Call Calculate_Inverse_Kinematics(x0, y0, z0, upper_leg(), end_effector, r_angles, Text3)
    Call Write_Upper_leg_values(Text3)

    Call rotation_angles_to_byte_conversion 'Convert angles and send data out port

End Sub

Private Sub rotation_angles_to_byte_conversion()

    '0 to 160 = 0 to 255
    Dim conversion_factor As Single

    conversion_factor = 255 / 190

    byte_r_angles.ang1 = Round(r_angles.ang1 * conversion_factor)
    byte_r_angles.ang2 = Round(r_angles.ang2 * conversion_factor)
    byte_r_angles.ang3 = Round(r_angles.ang3 * conversion_factor)
    byte_r_angles.ang4 = Round(r_angles.ang4 * conversion_factor)

    If (r_angles.ang1 <= 160) And (r_angles.ang2 <= 160) And (r_angles.ang3 <= 160) And (r_angles.ang4 <= 160) Then

        sld_servo1.Value = byte_r_angles.ang1
        sld_servo2.Value = byte_r_angles.ang2
        sld_servo3.Value = byte_r_angles.ang3
        sld_servo4.Value = byte_r_angles.ang4

        Call servo1_control
        Call servo2_control
        Call servo3_control
        Call servo4_control

    Else

        Text1.Text = "Angles out of Mechanicallly imposed limits.... 0 - 160 degrees."

    End If

End Sub

Private Sub Timer1_Timer()
```

```
  If (opn_demo.Value = False) Then

    'Disable timer
    Timer1.Enabled = False

    'Calculate Inverse Kinematics
    Call do_inverse_kinematics_calculation(0, 0, -18)

  Else ' Demo has been selected

    'Call demo routine
    Call Demo_Movement

  End If

End Sub

Private Sub Demo_Movement()

  Static demo_step As Integer

  demo_step = demo_step + 1

  If (demo_step = 1) Then

    Call Move_To_Coordinates(0, 0, -10)

  ElseIf (demo_step = 2) Then

    Call Move_To_Coordinates(5, 0, -18)

  ElseIf (demo_step = 3) Then

    Call Move_To_Coordinates(0, 0, -10)

  ElseIf (demo_step = 4) Then

    Call Move_To_Coordinates(-5, 0, -18)

  ElseIf (demo_step = 5) Then

    Call Move_To_Coordinates(0, 0, -10)

  ElseIf (demo_step = 6) Then

    Call Move_To_Coordinates(0, 5, -18)

  ElseIf (demo_step = 7) Then

    Call Move_To_Coordinates(0, 0, -10)

  ElseIf (demo_step = 8) Then

    Call Move_To_Coordinates(0, -5, -18)

  ElseIf (demo_step = 9) Then

    Call Move_To_Coordinates(0, 0, -10)

  ElseIf (demo_step = 10) Then

    Call Move_To_Coordinates(5, 0, -18)

  ElseIf (demo_step = 11) Then

    Call Move_To_Coordinates(5, 5, -18)

  ElseIf (demo_step = 12) Then

    Call Move_To_Coordinates(0, 5, -18)

  ElseIf (demo_step = 13) Then

    Call Move_To_Coordinates(-5, 5, -18)
  ElseIf (demo_step = 14) Then

    Call Move_To_Coordinates(-5, 0, -18)

  ElseIf (demo_step = 15) Then

    Call Move_To_Coordinates(-5, -5, -18)

  ElseIf (demo_step = 16) Then

    Call Move_To_Coordinates(0, -5, -18)

  ElseIf (demo_step = 17) Then

    Call Move_To_Coordinates(5, -5, -18)

  ElseIf (demo_step = 18) Then

    Call Move_To_Coordinates(5, 0, -18)

  '*****************************************
  ElseIf (demo_step = 19) Then

    Call Move_To_Coordinates(0, 0, -10)

  ElseIf (demo_step = 20) Then

    Call Move_To_Coordinates(5, 5, -18)

  ElseIf (demo_step = 21) Then

    Call Move_To_Coordinates(0, 0, -10)

  ElseIf (demo_step = 22) Then

    Call Move_To_Coordinates(-5, -5, -18)

  ElseIf (demo_step = 23) Then

    Call Move_To_Coordinates(0, 0, -10)

  ElseIf (demo_step = 24) Then

    Call Move_To_Coordinates(-5, 5, -18)

  ElseIf (demo_step = 25) Then

    Call Move_To_Coordinates(0, 0, -10)

  ElseIf (demo_step = 26) Then

    Call Move_To_Coordinates(5, -5, -18)

  Else

    Call Move_To_Coordinates(0, 0, -15)

    demo_step = 0
    opn_demo.Value = False
    opn_mouse.Value = True
    Timer1.Enabled = False

  End If

End Sub

Private Sub Move_To_Coordinates(xm As Single, ym As Single, zm As Single)

  Call do_inverse_kinematics_calculation(xm, ym, zm)
  Call draw_picbox_data(pic_horizontal_data, Text3, x0, y0, x0_old, y0_old, bool_all_servo_mvnt_comp)
  Call draw_picbox_data(pic_vertical_data, Text3, z0, y0, z0_old, y0_old, bool_all_servo_mvnt_comp)

End Sub
```

```
Option Explicit

Public Sub draw_picbox_data(pb As PictureBox, t As TextBox, xq As Single, yq As Single, xo As Single, yo As Single,
bool_stimulated As Boolean)

   Dim width As Single, height As Single, xp As Single, yp As Single
   Dim step_x As Single, step_y As Integer

   'Clear picture box
   pb.Cls

   width = pb.ScaleWidth
   height = pb.ScaleHeight

   'First sensor position
   xp = pb.ScaleLeft + 0.5
   yp = pb.ScaleTop - 0.5

   step_x = width / 16
   step_y = height / 16


   'Draw Reference Axes
   'Set line thickness, and style
   pb.FillColor = vbBlack
   pb.DrawStyle = 1     'Dash
   pb.Line (pb.ScaleLeft + width / 2, pb.ScaleTop) _
   -(pb.ScaleLeft + width / 2, pb.ScaleTop + height) ' y axis

   pb.Line (pb.ScaleLeft, pb.ScaleTop + height / 2) _
   -(pb.ScaleLeft + width, pb.ScaleTop + height / 2)   ' x axis

   ' check Scale values
   't.Text = pb.ScaleHeight & vbTab & pb.ScaleWidth & vbTab & xp & vbTab & yp & vbTab & step_x & vbTab & step_y


   'Fillcolour and fill style
   pb.FillColor = &HFFFF00   ' a light blue
   pb.DrawStyle = 0          ' Solid

   'Distance between centres of sensors is 10 mm, 16 by 16 grid of sensors
   For yp = pb.ScaleTop - 0.5 To (pb.ScaleTop - 0.5 + height) Step step_y
   'Rows - Represented by y
      For xp = pb.ScaleLeft + 0.5 To (pb.ScaleLeft + 0.5 + width) Step step_x
      'Columns - represented by x

         'xo, yo - old coordinates
         If (Abs(xo - xp) <= 0.1) And (Abs(yo - yp) <= 0.1) Then
            'Change fillcolour to red, then draw circle, and change fillcolour back
            If (bool_stimulated = True) Then
               pb.FillColor = vbRed
            End If

            pb.Circle (xp, yp), 0.2
            pb.FillColor = &HFFFF00

         Else
            ' Just draw the circle
            pb.Circle (xp, yp), 0.2
         End If

         'Draw bulls eye
         If (Abs(xq - xp) <= 0.1) And (Abs(yq - yp) <= 0.1) Then
            pb.FillColor = vbBlue
            pb.Circle (xp, yp), 0.1
            pb.FillColor = &HFFFF00
         End If

      Next xp

   Next yp

   'draw crosshair
   pb.Line (xq, pb.ScaleTop)-(xq, pb.ScaleTop + pb.ScaleHeight) '
   pb.Line (pb.ScaleLeft, yq)-(pb.ScaleLeft + pb.ScaleWidth, yq) '


End Sub


Option Explicit

Dim rad2deg As Single
Dim angle_leg1_min As Single
Dim angle_leg1_max As Single
Dim angles As rotation_angles

Public Sub initialize_variables()
   rad2deg = 180 / 3.141592654

   angle_leg1_min = 25
   angle_leg1_max = 245


End Sub

Private Sub Coordinates_not_found(L As leg)
   'Sets values to impossible values, because coordinates cannot be found
   L.angle = -400
   L.knee_coordinates.x = -400
   L.knee_coordinates.y = -400
   L.knee_coordinates.z = -400
End Sub

Private Sub Reset_Knee_Joint_Solutions(x() As Single)
   x(0) = 0
   x(1) = 0
   x(2) = 0
End Sub


Public Sub Calculate_Inverse_Kinematics(x As Single, y As Single, z As Single, L() As leg, ee As EEAJ, r_a As rotation_angles,
t As TextBox)

   Dim x_0 As Single, y_0 As Single, z_0 As Single, angle1 As Single, angle2 As Single
   Dim solution As angle_and_coordinates
   Dim c1 As Single, c2 As Single
   Dim temp As Single
   Dim all_equations_solved As Boolean
   Dim COKJS1(2) As Single, COKJS2(2) As Single
   Dim i As Integer

   'Set Conversion constant
   'rad2deg

   x_0 = x
   y_0 = y
   z_0 = z
   i = 1

   ' Set end effector ankle joints
   ee.AJ(0).x = x + 4.45
   ee.AJ(0).y = y
   ee.AJ(0).z = z

   ee.AJ(1).x = x
   ee.AJ(1).y = y - 4.45
   ee.AJ(1).z = z

   ee.AJ(2).x = x - 4.45
   ee.AJ(2).y = y
   ee.AJ(2).z = z

   ee.AJ(3).x = x
   ee.AJ(3).y = y + 4.45
   ee.AJ(3).z = z

   all_equations_solved = True
```

```
  'To Prevent division by 0
  If (z_0 <> 0) Then
    'Solving Leg 1 Knee coordinates
    c1 = (5.55 - x_0) / z_0
    c2 = (x_0 ^ 2 + y_0 ^ 2 + z_0 ^ 2 + 8.9 * x_0 - 300.607) / (2 * z_0)

    temp = 100 - 20 * c1 * c2 - c2 ^ 2
    If (temp < 0) Or (all_equations_solved = False) Then
      all_equations_solved = False
      Call Coordinates_not_found(L(0))
    Else
      'Coordinates can be found
      'Solutions set 1
      Call Reset_Knee_Joint_Solutions(COKJS1())
      'x Value
      COKJS1(0) = (-c1 * c2 + 10 + temp ^ 0.5) / (c1 ^ 2 + 1)
      'z Value
      COKJS1(2) = c1 * COKJS1(0) + c2

      'Solution Set 2
      Call Reset_Knee_Joint_Solutions(COKJS2())
      'x Value
      COKJS2(0) = (-c1 * c2 + 10 - temp ^ 0.5) / (c1 ^ 2 + 1)
      'z Value
      COKJS2(2) = c1 * COKJS2(0) + c2

      t.Text = t.Text & vbNewLine & vbNewLine & "Solution Sets for leg:" & i
      t.Text = t.Text & vbNewLine & "SS1 - (" & Format$(COKJS1(0), "Fixed") & "; " & Format$(COKJS1(1), "Fixed") & ";
" & Format$(COKJS1(2), "Fixed") & ")" & _
      vbTab & "SS2 - (" & Format$(COKJS2(0), "Fixed") & "; " & Format$(COKJS2(1), "Fixed") & "; " &
Format$(COKJS2(2), "Fixed") & ")"

      't.Text = t.Text & vbNewLine & ((COKJS1(0) - ee.AJ(0).x) ^ 2 + (COKJS1(1) - ee.AJ(0).y) ^ 2 + (COKJS1(2) -
ee.AJ(0).z) ^ 2)
      't.Text = t.Text & vbNewLine & ((COKJS2(0) - ee.AJ(0).x) ^ 2 + (COKJS2(1) - ee.AJ(0).y) ^ 2 + (COKJS2(2) -
ee.AJ(0).z) ^ 2)

      '****************************************************************************************
      solution = Determine_Correct_Solution(COKJS1, COKJS2, ee.AJ(0), L(0).thigh_coordinates)
      t.Text = t.Text & vbNewLine & "Angle: " & Format$(solution.angle, "Fixed")
      t.Text = t.Text & vbTab & "Coordinates - (" & Format$(solution.coordinates.x, "Fixed") & "; " & _
      Format$(solution.coordinates.y, "Fixed") & "; " & Format$(solution.coordinates.z, "Fixed") & ")"

      L(0).angle = solution.angle
      L(0).knee_coordinates = solution.coordinates

    End If

    i = i + 1

    'Sloving Leg 2 Knee Coordinates
    c1 = (-5.55 - y_0) / z_0
    c2 = (x_0 ^ 2 + y_0 ^ 2 + z_0 ^ 2 - 8.9 * y_0 - 300.607) / (2 * z_0)

    temp = 100 + 20 * c1 * c2 - c2 ^ 2
    If (temp < 0) Or (all_equations_solved = False) Then
      all_equations_solved = False
      Call Coordinates_not_found(L(1))
    Else
      'Coordinates can be found
      'Solutions set 1
      Call Reset_Knee_Joint_Solutions(COKJS1())
      'y Value
      COKJS1(1) = (-c1 * c2 - 10 + temp ^ 0.5) / (c1 ^ 2 + 1)
      'z Value
      COKJS1(2) = c1 * COKJS1(1) + c2

      'Solution Set 2
      Call Reset_Knee_Joint_Solutions(COKJS2())
      'y Value
      COKJS2(1) = (-c1 * c2 - 10 - temp ^ 0.5) / (c1 ^ 2 + 1)
      'z Value
      COKJS2(2) = c1 * COKJS2(1) + c2

      t.Text = t.Text & vbNewLine & vbNewLine & "Solution Sets for leg:" & i
      t.Text = t.Text & vbNewLine & "SS1 - (" & Format$(COKJS1(0), "Fixed") & "; " & Format$(COKJS1(1), "Fixed") & ";
" & Format$(COKJS1(2), "Fixed") & ")" & _
      vbTab & "SS2 - (" & Format$(COKJS2(0), "Fixed") & "; " & Format$(COKJS2(1), "Fixed") & "; " &
Format$(COKJS2(2), "Fixed") & ")"
      'Check for Correct Solutions

      '*******************************************************************************
      solution = Determine_Correct_Solution(COKJS1, COKJS2, ee.AJ(1), L(1).thigh_coordinates)
      t.Text = t.Text & vbNewLine & "Angle: " & Format$(solution.angle, "Fixed")
      t.Text = t.Text & vbTab & "Coordinates - (" & Format$(solution.coordinates.x, "Fixed") & "; " & _
      Format$(solution.coordinates.y, "Fixed") & "; " & Format$(solution.coordinates.z, "Fixed") & ")"

      L(1).angle = solution.angle
      L(1).knee_coordinates = solution.coordinates

    End If

    i = i + 1

    'Solving Leg 3 Knee Coordinates
    c1 = (-5.55 - x_0) / z_0
    c2 = (x_0 ^ 2 + y_0 ^ 2 + z_0 ^ 2 - 8.9 * x_0 - 300.607) / (2 * z_0)

    temp = 100 + 20 * c1 * c2 - c2 ^ 2
    If (temp < 0) Or (all_equations_solved = False) Then
      all_equations_solved = False
      Call Coordinates_not_found(L(2))
    Else
      'Coordinates can be found
      'Solutions set 1
      Call Reset_Knee_Joint_Solutions(COKJS1())
      'x Value
      COKJS1(0) = (-c1 * c2 - 10 + temp ^ 0.5) / (c1 ^ 2 + 1)
      'z Value
      COKJS1(2) = c1 * COKJS1(0) + c2

      'Solution Set 2
      Call Reset_Knee_Joint_Solutions(COKJS2())
      'x Value
      COKJS2(0) = (-c1 * c2 - 10 - temp ^ 0.5) / (c1 ^ 2 + 1)
      'z Value
      COKJS2(2) = c1 * COKJS2(0) + c2

      t.Text = t.Text & vbNewLine & vbNewLine & "Solution Sets for leg:" & i
      t.Text = t.Text & vbNewLine & "SS1 - (" & Format$(COKJS1(0), "Fixed") & "; " & Format$(COKJS1(1), "Fixed") & ";
" & Format$(COKJS1(2), "Fixed") & ")" & _
      vbTab & "SS2 - (" & Format$(COKJS2(0), "Fixed") & "; " & Format$(COKJS2(1), "Fixed") & "; " &
Format$(COKJS2(2), "Fixed") & ")"
      'Check for Correct Solutions

      '*******************************************************************************
      solution = Determine_Correct_Solution(COKJS1, COKJS2, ee.AJ(2), L(2).thigh_coordinates)
      t.Text = t.Text & vbNewLine & "Angle: " & Format$(solution.angle, "Fixed")
      t.Text = t.Text & vbTab & "Coordinates - (" & Format$(solution.coordinates.x, "Fixed") & "; " & _
      Format$(solution.coordinates.y, "Fixed") & "; " & Format$(solution.coordinates.z, "Fixed") & ")"

      L(2).angle = solution.angle
      L(2).knee_coordinates = solution.coordinates

    End If

    i = i + 1

    'Solving Leg 4 Knee Coordinates
    c1 = (5.55 - y_0) / z_0
    c2 = (x_0 ^ 2 + y_0 ^ 2 + z_0 ^ 2 + 8.9 * y_0 - 300.607) / (2 * z_0)

    temp = 100 - 20 * c1 * c2 - c2 ^ 2
    If (temp < 0) Or (all_equations_solved = False) Then
      all_equations_solved = False
      Call Coordinates_not_found(L(1))
    Else
      'Coordinates can be found
      'Solutions set 1
      Call Reset_Knee_Joint_Solutions(COKJS1())
```

```
            'y Value
            COKJS1(1) = (-c1 * c2 + 10 + temp ^ 0.5) / (c1 ^ 2 + 1)
            'z Value
            COKJS1(2) = c1 * COKJS1(1) + c2

            'Solution Set 2
            Call Reset_Knee_Joint_Solutions(COKJS2())
            'y Value
            COKJS2(1) = (-c1 * c2 + 10 - temp ^ 0.5) / (c1 ^ 2 + 1)
            'z Value
            COKJS2(2) = c1 * COKJS2(1) + c2

            t.Text = t.Text & vbNewLine & vbNewLine & "Solution Sets for leg:" & i
            t.Text = t.Text & vbNewLine & "SS1 - (" & Format$(COKJS1(0), "Fixed") & "; " & Format$(COKJS1(1), "Fixed") & ";
" & Format$(COKJS1(2), "Fixed") & ")" & _
            vbTab & "SS2 - (" & Format$(COKJS2(0), "Fixed") & "; " & Format$(COKJS2(1), "Fixed") & "; " &
Format$(COKJS2(2), "Fixed") & ")"
            'Check for Correct Solutions

            '**************************************************************************************

            solution = Determine_Correct_Solution(COKJS1, COKJS2, ee.AJ(3), L(3).thigh_coordinates)
            t.Text = t.Text & vbNewLine & "Angle: " & Format$(solution.angle, "Fixed")
            t.Text = t.Text & vbTab & "Coordinates - (" & Format$(solution.coordinates.x, "Fixed") & "; " & _
            Format$(solution.coordinates.y, "Fixed") & "; " & Format$(solution.coordinates.z, "Fixed") & ")"

            L(3).angle = solution.angle
            L(3).knee_coordinates = solution.coordinates

        End If

        'Converting to servo rotation angles

        angles = Convert_to_servo_rotation_angles(angles)

        t.Text = t.Text & vbNewLine
        t.Text = t.Text & vbNewLine & "Servo Rotation Angles : (" & Format$(angles.ang1, "Fixed") & "; " & _
        Format$(angles.ang2, "Fixed") & "; " & Format$(angles.ang3, "Fixed") & "; " & _
        Format$(angles.ang4, "Fixed") & ")" & vbNewLine

        r_a = angles

    End If

End Sub

Public Sub Set_Thigh_Joint_Coordinates(L() As leg)

    L(0).thigh_coordinates.x = 10
    L(0).thigh_coordinates.y = 0
    L(0).thigh_coordinates.z = 0

    L(1).thigh_coordinates.x = 0
    L(1).thigh_coordinates.y = -10
    L(1).thigh_coordinates.z = 0

    L(2).thigh_coordinates.x = -10
    L(2).thigh_coordinates.y = 0
    L(2).thigh_coordinates.z = 0

    L(3).thigh_coordinates.x = 0
    L(3).thigh_coordinates.y = 10
    L(3).thigh_coordinates.z = 0

End Sub

Private Function Get_upper_leg_angle(z1 As Single, z2 As Single, r1 As Single, r2 As Single)

    Dim num As Single, den As Single, angle As Single

    num = z1 - z2
    den = r1 - r2

    If (den <> 0) Then

        angle = rad2deg * Atn(num / den)
    ' First Quadrant
    If (num = 0) And (den > 0) Then
      angle = 0

    ElseIf (num > 0) And (den > 0) Then
      'No change to angle

    ' Second Quadrant - atn is negative
    ElseIf (num > 0) And (den < 0) Then
      angle = angle - 180 ' Force angle to go from -180 to -270

    ElseIf (num = 0) And (den < 0) Then
      angle = -180

    ' Third Quadrant
    ElseIf (num < 0) And (den < 0) Then
      angle = angle - 180 ' Angle must range from -90 to -180

    ' Fourth Quadrant
    ElseIf (num < 0) And (den > 0) Then
      angle = angle   ' No change

    End If

  Else ' den = 0

    If (num > 0) Then
      angle = 90
    ElseIf (num < 0) Then
      angle = -90
    Else
      'Problems cannot determine angle as num = 0 and den = 0
    End If

  End If

  angle = (angle + 360) Mod 360
  Get_upper_leg_angle = angle ' Return Angle

End Function


Private Function Determine_Correct_Solution(s1() As Single, s2() As Single, a As XYZ_coordinates, t As XYZ_coordinates) As
angle_and_coordinates

  Dim ang1 As Single, ang2 As Single
  Dim temp As angle_and_coordinates
  Dim m As Single, c As Single, z1 As Single, z2 As Single

  '************** LEG 1 *****************************
  If t.x = 10 Then ' For leg 1
    ang1 = Get_upper_leg_angle(s1(2), 0, s1(0), 10)
    ang2 = Get_upper_leg_angle(s2(2), 0, s2(0), 10)

    If t.x <> a.x Then ' else den is 0
      'Gradient of line A1T1 = DZ/DX
      m = (t.z - a.z) / (t.x - a.x)
      c = -10 * m
      z1 = m * s1(0) + c
      z2 = m * s2(0) + c

      'Now test gradient of A1T1
      If (m > 0) Then
      'z of Knee coordinate must be less than z of A1T1 at corresponding x
        If (s1(2) < z1) Then
          temp.angle = ang1
          temp.coordinates.x = s1(0)
          temp.coordinates.y = s1(1)
          temp.coordinates.z = s1(2)

        ElseIf (s2(2) < z2) Then
          temp.angle = ang2
          temp.coordinates.x = s2(0)
          temp.coordinates.y = s2(1)
        temp.coordinates.z = s2(2)
      Else
        'Problems - Singularity
        temp.angle = -400
        temp.coordinates.x = -400
        temp.coordinates.y = -400
        temp.coordinates.z = -400
      End If

    ElseIf (m < 0) Then
    'z of knee coodinate must be more than A1T1 at corresponding x
      If (s1(2) > z1) Then
        temp.angle = ang1
        temp.coordinates.x = s1(0)
        temp.coordinates.y = s1(1)
        temp.coordinates.z = s1(2)
      ElseIf (s2(2) > z2) Then
        temp.angle = ang2
        temp.coordinates.x = s2(0)
        temp.coordinates.y = s2(1)
        temp.coordinates.z = s2(2)
      Else
        'Problems - Singularity
        temp.angle = -400
        temp.coordinates.x = -400
        temp.coordinates.y = -400
        temp.coordinates.z = -400
      End If

    End If

  Else ' t.x = a.x -> den goes to 0
    If (Abs(s1(0)) > Abs(s2(0))) Then
      temp.angle = ang1
      temp.coordinates.x = s1(0)
      temp.coordinates.y = s1(1)
      temp.coordinates.z = s1(2)

    ElseIf (Abs(s1(0)) < Abs(s2(0))) Then
      temp.angle = ang2
      temp.coordinates.x = s2(0)
      temp.coordinates.y = s2(1)
      temp.coordinates.z = s2(2)
    Else
      'Problems
    End If

  End If

  angles.ang1 = temp.angle

'***************** LEG 3 *********************************
ElseIf t.x = -10 Then ' leg 3
  ang1 = Get_upper_leg_angle(s1(2), 0, s1(0), -10)
  ang2 = Get_upper_leg_angle(s2(2), 0, s2(0), -10)

  If t.x <> a.x Then ' else den is 0
    'Gradient of line A3T3 = DZ/DX
    m = (t.z - a.z) / (t.x - a.x)
    c = 10 * m
    z1 = m * s1(0) + c
    z2 = m * s2(0) + c

    'Now test gradient of A3T3
    If (m > 0) Then
    'z of Knee coordinate must be less than z of A3T3 at corresponding x
      If (s1(2) > z1) Then
        temp.angle = ang1
        temp.coordinates.x = s1(0)
        temp.coordinates.y = s1(1)
        temp.coordinates.z = s1(2)

      ElseIf (s2(2) > z2) Then
        temp.angle = ang2
        temp.coordinates.x = s2(0)
```

```
            temp.coordinates.y = s2(1)
            temp.coordinates.z = s2(2)
         Else
            'Problems - Singularity
            temp.angle = -400
            temp.coordinates.x = -400
            temp.coordinates.y = -400
            temp.coordinates.z = -400
         End If

      ElseIf (m < 0) Then
      'z of knee coodinate must be more than A3T3 at corresponding x
         If (s1(2) < z1) Then
            temp.angle = ang1
            temp.coordinates.x = s1(0)
            temp.coordinates.y = s1(1)
            temp.coordinates.z = s1(2)
         ElseIf (s2(2) < z2) Then
            temp.angle = ang2
            temp.coordinates.x = s2(0)
            temp.coordinates.y = s2(1)
            temp.coordinates.z = s2(2)
         Else
            'Problems - Singularity
            temp.angle = -400
            temp.coordinates.x = -400
            temp.coordinates.y = -400
            temp.coordinates.z = -400
         End If

      End If

   Else ' t.x = a.x -> den goes to 0
      If (Abs(s1(0)) > Abs(s2(0))) Then
         temp.angle = ang1
         temp.coordinates.x = s1(0)
         temp.coordinates.y = s1(1)
         temp.coordinates.z = s1(2)

      ElseIf (Abs(s1(0)) < Abs(s2(0))) Then
         temp.angle = ang2
         temp.coordinates.x = s2(0)
         temp.coordinates.y = s2(1)
         temp.coordinates.z = s2(2)
      Else
         'Problems
      End If

   End If

   angles.ang3 = temp.angle

'************************* LEG 2 ********************************************
ElseIf t.y = -10 Then ' leg 2

   ang1 = Get_upper_leg_angle(s1(2), 0, s1(1), -10)
   ang2 = Get_upper_leg_angle(s2(2), 0, s2(1), -10)

   If t.y <> a.y Then  ' else den is 0
      'Gradient of line A2T2 = DZ/DY
      m = (t.z - a.z) / (t.y - a.y)
      c = 10 * m
      z1 = m * s1(1) + c
      z2 = m * s2(1) + c

      'Now test gradient of A2T2
      If (m > 0) Then
      'z of Knee coordinate must be less than z of A2T2 at corresponding y
         If (s1(2) > z1) Then
            temp.angle = ang1
            temp.coordinates.x = s1(0)
            temp.coordinates.y = s1(1)
            temp.coordinates.z = s1(2)

         ElseIf (s2(2) > z2) Then
```

```
        temp.angle = ang2
        temp.coordinates.x = s2(0)
        temp.coordinates.y = s2(1)
        temp.coordinates.z = s2(2)
      Else
        'Problems - Singularity
        temp.angle = -400
        temp.coordinates.x = -400
        temp.coordinates.y = -400
        temp.coordinates.z = -400
      End If

    ElseIf (m < 0) Then
    'z of knee coodinate must be more than A2T2 at corresponding y
      If (s1(2) < z1) Then
        temp.angle = ang1
        temp.coordinates.x = s1(0)
        temp.coordinates.y = s1(1)
        temp.coordinates.z = s1(2)
      ElseIf (s2(2) < z2) Then
        temp.angle = ang2
        temp.coordinates.x = s2(0)
        temp.coordinates.y = s2(1)
        temp.coordinates.z = s2(2)
      Else
        'Problems - Singularity
        temp.angle = -400
        temp.coordinates.x = -400
        temp.coordinates.y = -400
        temp.coordinates.z = -400
      End If

    End If

  Else ' t.y = a.y -> den goes to 0
    If (Abs(s1(1)) > Abs(s2(1))) Then
      temp.angle = ang1
      temp.coordinates.x = s1(0)
      temp.coordinates.y = s1(1)
      temp.coordinates.z = s1(2)

    ElseIf (Abs(s1(1)) < Abs(s2(1))) Then
      temp.angle = ang2
      temp.coordinates.x = s2(0)
      temp.coordinates.y = s2(1)
      temp.coordinates.z = s2(2)
    Else
      'Problems
    End If

  End If

  angles.ang2 = temp.angle

'************************* LEG 4 *********************************************
ElseIf t.y = 10 Then ' leg 4

  ang1 = Get_upper_leg_angle(s1(2), 0, s1(1), 10)
  ang2 = Get_upper_leg_angle(s2(2), 0, s2(1), 10)

  If t.x <> a.x Then ' else den is 0
    'Gradient of line A4T4 = DZ/DY
    m = (t.z - a.z) / (t.y - a.y)
    c = -10 * m
    z1 = m * s1(1) + c
    z2 = m * s2(1) + c

    'Now test gradient of A4T4
    If (m > 0) Then
    'z of Knee coordinate must be less than z of A4T4 at corresponding y
      If (s1(2) < z1) Then
        temp.angle = ang1
        temp.coordinates.x = s1(0)
        temp.coordinates.y = s1(1)
        temp.coordinates.z = s1(2)
        ElseIf (s2(2) < z2) Then
          temp.angle = ang2
          temp.coordinates.x = s2(0)
          temp.coordinates.y = s2(1)
          temp.coordinates.z = s2(2)
        Else
          'Problems - Singularity
          temp.angle = -400
          temp.coordinates.x = -400
          temp.coordinates.y = -400
          temp.coordinates.z = -400
        End If

      ElseIf (m < 0) Then
      'z of knee coodinate must be more than A4T4 at corresponding y
        If (s1(2) > z1) Then
          temp.angle = ang1
          temp.coordinates.x = s1(0)
          temp.coordinates.y = s1(1)
          temp.coordinates.z = s1(2)
        ElseIf (s2(2) > z2) Then
          temp.angle = ang2
          temp.coordinates.x = s2(0)
          temp.coordinates.y = s2(1)
          temp.coordinates.z = s2(2)
        Else
          'Problems - Singularity
          temp.angle = -400
          temp.coordinates.x = -400
          temp.coordinates.y = -400
          temp.coordinates.z = -400
        End If

      End If

    Else ' t.y = a.y -> den goes to 0
      If (Abs(s1(1)) > Abs(s2(1))) Then
        temp.angle = ang1
        temp.coordinates.x = s1(0)
        temp.coordinates.y = s1(1)
        temp.coordinates.z = s1(2)

      ElseIf (Abs(s1(1)) < Abs(s2(1))) Then
        temp.angle = ang2
        temp.coordinates.x = s2(0)
        temp.coordinates.y = s2(1)
        temp.coordinates.z = s2(2)
      Else
        'Problems
      End If

    End If

    angles.ang4 = temp.angle

  End If


  Determine_Correct_Solution = temp

End Function

Private Function Convert_to_servo_rotation_angles(b As rotation_angles) As rotation_angles

  b.ang1 = (Abs(b.ang1 - 405)) Mod 360 ' From 45 to -135, clockwise convert to 0 to 180

  b.ang2 = (b.ang2 - 135)             ' From 135 to 315, clockwise convert to 0 to 180

  b.ang3 = (b.ang3 - 135)             ' From 135 to 315, clockwise convert to 0 to 180

  b.ang4 = (Abs(b.ang4 - 405)) Mod 360 ' From 45 to -135, clockwise convert to 0 to 180

  'Return values
  Convert_to_servo_rotation_angles = b
```

```
End Function

Option Explicit

Public Type XYZ_coordinates

    'May need to change these to single
    x As Single
    y As Single
    z As Single

End Type


Public Type leg

    angle As Single
    knee_coordinates As XYZ_coordinates
    thigh_coordinates As XYZ_coordinates

End Type


Public Type EEAJ

    'End effector Ankle Joints
    AJ(3) As XYZ_coordinates

End Type


Public Type angle_and_coordinates

    angle As Single
    coordinates As XYZ_coordinates

End Type


Public Type rotation_angles

    ang1 As Single
    ang2 As Single
    ang3 As Single
    ang4 As Single

End Type
```