Hexapod Motion Through Remote Computer Activation

Joseph Dudek

Hexapod Motion Through Remote Computer Activation

Joseph Dudek
Laboratory for Laser Energetics
University of Rochester

August 2, 2004

Abstract

This paper describes the creation of software for a motorized platform known as a 'hexapod'. This mechanism will be used to move a parabolic mirror in all 6 degrees of freedom (3 translational, 3 rotational) to focus an ultra-fast laser beam for experiments. The mathematics behind multiple coordinate systems, matrix mathematics, and vector calculations were used to develop algorithms for the movement of the hexapod. As a part of my implementation, I used both the C programming language and the Java™ programming language created by Sun Microsystems.

Outline

Figures

1 Introduction

In order to adequately maneuver a platform using six degrees of freedom, a hexapod platform is an attractive option. This small, portable machinery consists of two circular plates and six longitudinally expandable and contractible arms. By properly calculating the necessary lengths of each of the six arms, any rotation or translation of the upper platform (within accepted limits) may be accomplished. The ultimate goal of this algorithmic project is to manipulate a hexapod platform using a driver interface and a computer program.

1.1 Hexapod Platform

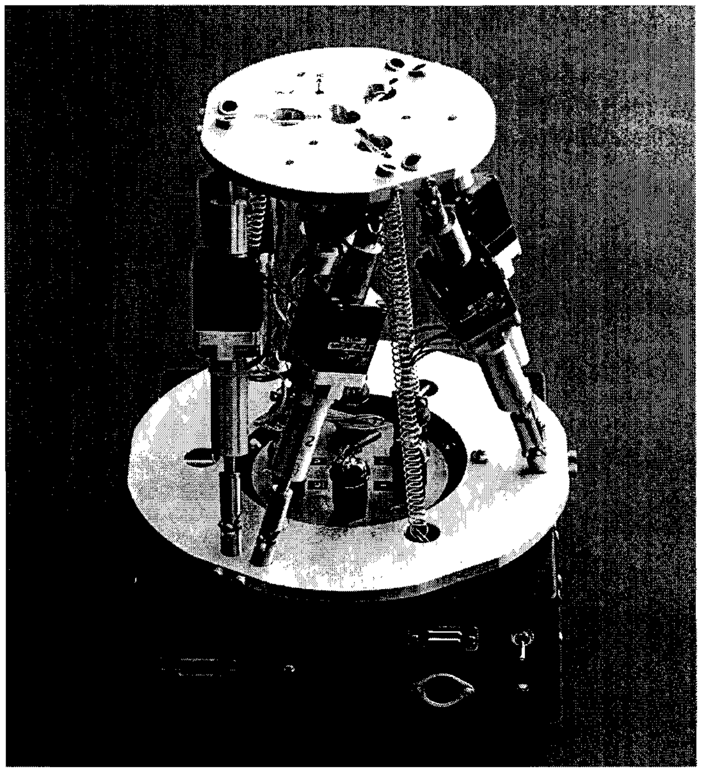

Figure 1.1
The Hexapod Platform

The hexapod itself (as seen in Fig. 1.1) is a platform structure comprised of two circular discs connected by six motorized actuators. The six actuators on the hexapod form consecutive triangles around the outside of the structure contacting each plate in three locations. As three points successfully define a plane, this method maintains stability with the geometric triangle and also needs only linear actuator motion to achieve motion in all 6 degrees of freedom of the upper disc.

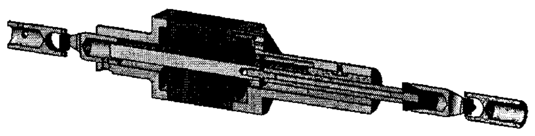

Figure 1.2
The Actuator

The motorized actuators are what provide the hexapod with extreme accuracy and flexibility. The hexapod's six actuators act as both support for the upper platform and a means for movement. Motion is achieved through the use of universal joints located at either end of the actuator. These joints allow for freedom of movement to extremely wide angles. Since the hexapod's functionality depends heavily on its flexibility, these joints are vital to the mechanism. Each actuator also contains a stepper motor that had one step equal to 3.580 μm and is activated via a control module located at the bottom of the hexapod.(Figure 1.1) The stepper motor mobilizes a shaft that leads out to one of the universal joints on the actuator.(Figure 1.2)

1.2 Hexapod Driver Unit and Protocol

Provided for this project was a custom driver unit created by Jean DePatie. This driver utilized a serial port to receive commands for the hexapod's actuators. This function utilized a rudimentary string function where the first two characters defined the actuator, the next character defined the course of action, and the final accepted character or series of characters represented a value for the function.

Function	Definition	Execution
H	Set Home Position	Creates a zero position some distance above the home switch (steps)
V	Set Velocity	Sets the velocity of the stepper motors (steps/second)
P	Move to Position...	Moves the stepper motors to a numerical position (steps)
S	Report Status	Returns a string with the current position of the motor requested (steps)

2 Mathematics

The mathematics used for this project achieved a relatively complex solution by combining more basic equations and functions. Given the goal of accomplishing six degrees of freedom and the known constraint of longitudinal motion, the most effective method of computation seemed to be using two coordinate systems. With one system at the base and another atop the plate, vectors and locations could be simply translated between the two, thus separating the rotation and translation aspect from the calculation of new arm lengths.

2.1 Managing Coordinate Systems

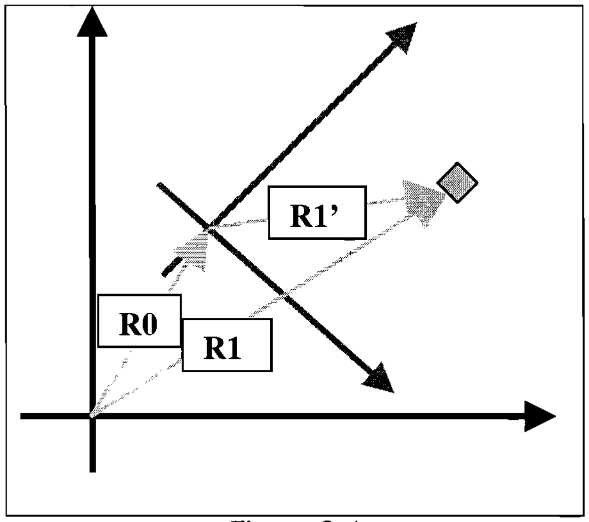

Figure 2.1
Vector Addition

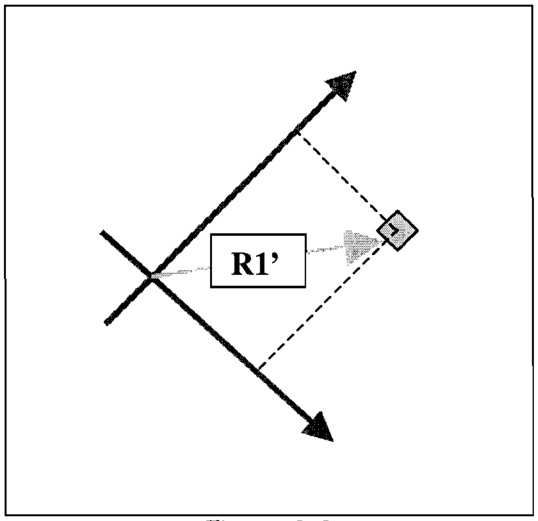

Figure 2.2
Projections Establishing Coordinates

In the case of this particular project, vectors were used as a means of translating coordinates between different coordinate systems. The hexapod must be considered to have two different sets of coordinate systems that must each be accounted for at all times. The lower, stationary platform (base) must keep track of the locations of all the parts of the hexapod as it is the only stationary point of reference. Also, the upper, mobile platform (plate) must retain its own coordinate system. Instead of constantly updating both sets of information, vector math is utilized to translate base information into the plate coordinate system, the movement is executed in the plate system, and the new coordinates are translated back into the base system. This way only the base coordinates must be retained from one movement to the next.

Vector mathematics tells us that the three vectors in Figure 2.1 are related by the equation R1-R0=R1'. This means that if a vector to the origin of the plate coordinate system(vector R0) and a vector to the point of interest were known(vector R1), the vector to the point of interest in the plate system would be a simple calculation away(vector R1'). Additionally, the magnitudes of the projections of the translated vector R1' onto the axis vectors of the plate system will create coordinates of the vector in the global system. While the demonstration above handles only two dimensional examples, this concept translates to three dimensions quite easily.

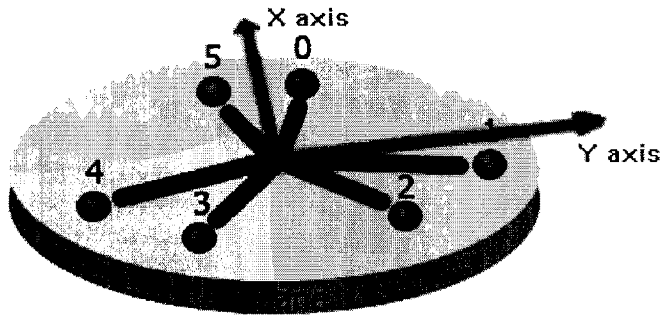

Figure 2.3
Hexapod Coordinate System Unit Vectors

Establishing the axes for translation was relatively straightforward. First, a set of axes was established at the center of the six upper universal joints. As seen in Figure 2.3, if the vectors from the center of the plane to Joint 0 and Joint 5 are added together, an x-axis is created. Additionally, if the vector to Joint 4 is subtracted from the vector to Joint 1, a y-axis is created. The cross-product of the x- and y-axes creates an orthogonal vector which becomes the z-axis. The z-axis is then elevated to compensate for the distance between the universal joints and the plate itself.

2.2 Utilizing Coordinate Translations

With coordinate system translations proven as an executable function, the ideas mentioned above can be put to use. All of the universal joints on the hexapod were tracked in the global coordinate system, here the base, and this data was stored. Whenever a user requests a rotation of the upper plate, the upper joints would be translated into the plate's perspective, thus allowing a rotation using basic rotation matrices. With the new joint locations calculated, they would be translated back into the global system and the new leg lengths would be calculated. At that point, it was just a matter of executing the movements on the hexapod.

3 Computer Programming

The primary goal of this project was not to verify a method of mobilizing the hexapod, but rather to create a fully functional computer program. Because of this, the architecture of this program must be discussed. This section will cover the programming language, target operating system, program design, and implementation.

3.1 The C Programming Language

The C Programming Language was developed in the early 1970's by Ken Thompson and Dennis Ritchie for their UNIX operating system. C has since spread to many other operating systems and builds and has become one of the most popular languages in the world. C is best known for its efficiency, one of the major reasons it was chosen for this project. C also runs several files as though they were one long program file, making long code seem simple as it is condensed into one executable file.

3.2 Code Design and Implementation

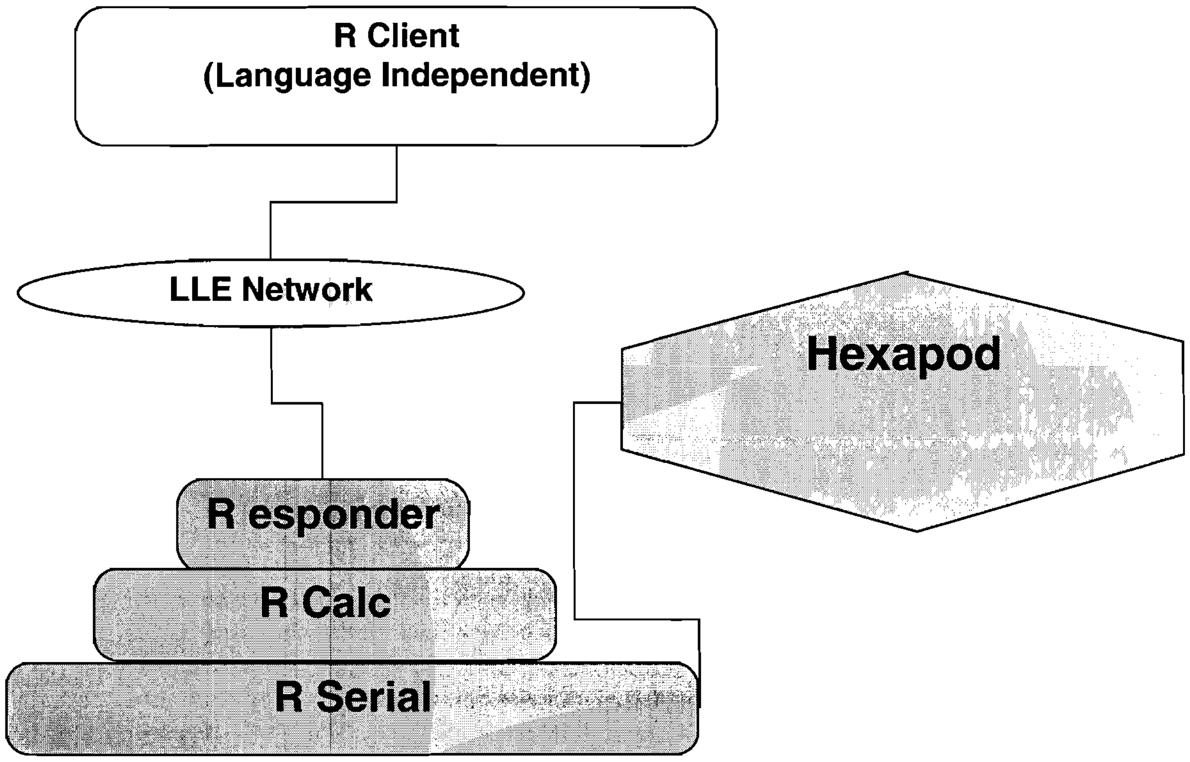

Figure 3.1
Overview of Hexapod Computer Model

In order to fulfill the requirements for the programming portion of the project, several programs were designed to work in tandem, each performing a unique function and passing its results to the next program in line. The four components used in this particular design are RClient, a remote access program, Responder, a receiver protocol, RCalc, the calculator and translator, and RSerial, the utility that sends messages through the serial port to the driver unit. Each of these programs are connected through different types of sockets established upon execution and must be initialized from the bottom up.

3.2.1 RSerial

This program was designed specifically for communication over a serial port. It first converts leg lengths given by RCalc into 'steps'. Steps are 3.850 micrometer lengths which the stepper motor recognizes as a unit of measurement. Once the distances have been converted, this program creates messages that the driver unit can understand. With that step complete, the strings are then transmitted across the serial port and a confirmation message is received following the message. The goal for the program series is to be able to interpret the messages received from the driver hardware and react accordingly, but that function has not yet been added to the software.

3.2.2 RCalc

The first segment of code that was created was the basic functionality required to use vector mathematics, rotational matrices, and other utility functions. Fortunately, the coding for these segments became rudimentary, as the coding was simply copying pre-defined equations into usable methods. Each matrix equation designed to interact with a vector or another matrix was designed only to handle vectors with three data segments and 3x3 matrices. While this restricts the reusability of the code, it was necessary to improve efficiency in this project.

The protocol for establishing several coordinate systems and translating between them was then established. Using sample universal joint coordinates, an intermediate coordinate system is set up during translation. That coordinate system exists as an average of the locations of all of the upper universal joints establishing the angle necessary to compensate for any previous platform adjustment. The axes are then created according to a preset system of equations (see Section 2.2). By increasing the position of the origin a constant number of millimeters up the z-axis, the coordinate system has now been transformed into the system at the center of the plate. The correct translation or rotation is then executed and the coordinate system is then translated back to the global position.

In accomplishing the above changes in coordinate systems, the methodology used is not to identify the coordinate system totally, but rather to edit the vectors to the universal joints so that they represent vectors from a different perspective. For example, the translation from the center of the upper universal joints to the center of the plate is, in reality, a subtraction from the z values of each of the vectors. In addition, translating back to the global position is merely adding a vector from the global origin to the plate origin to each of the universal joints' positions and using the stored axis vectors for the plate system to rotate back to vertical.

With the translation complete, taking a simple measurement of each of the arms, subtracting the length of a collapsed arm, and dividing by the distance in one motor step (3.580 μm) will establish a position number for that arm. This position is then forwarded to the serial port through a program entitled RSerial.

3.2.3 Responder

Responder was simply an intermediate between RClient and RCalc. It had to establish two different types of sockets, one to the network to receive messages from RClient and one to RCalc to transmit messages. No processing occurred in this module, but the information received was translated into a message that was then just passed down through the line. This program is run from a small LINUX computer situated near the hexapod platform. This LINUX computer has an Ethernet port allowing remote access from anywhere on the LLE network.

3.2.4 RClient

One of the most important facets of the coding for this project was remote access. The computer running this platform does not have input functionality besides a couple of Ethernet ports and three serial ports. Using a keyboard or mouse, therefore, is impossible. The computer will also exist inside the laser bay meaning that anything that can keep personnel away from the unit will be helpful. The code, therefore, is designed to execute upon a command from an outside program. RClient acts as a user client and allows a user to give the platform commands from any distance away so long as the user has access to the local network. This program does nothing more than forward commands over the network to Responder.

4 Conclusion

Manipulation of a six-legged platform requires both vector and matrix mathematics working through a computer interface. This interface has been created as a part of my project at the LLE and is ready for implementation in the Omega EP laser system.

In order to use the program to its fullest functionality, a much more accurate method of measuring the locations of the universal joints on the mechanism with respect to the center of the base must be found. Without this measurement, any calculations made by the program will be based on estimated numbers creating a significant amount of error. In addition, a more user-friendly client interface must be created to allow use from the control room by average personnel. Lastly, the positions of the arms must be logged to external files allowing the program to cope with a loss of power or a fatal error.

5 Acknowledgements

This project was facilitated by a High School Summer Intern program at the University of Rochester Laboratory for Laser Energetics. Dr. R. S. Craxton was the program supervisor and I was supervised individually by Dr. Christian Stoeckl. The driver unit for the hexapod was created and programmed by Jean DePatie and I owe each of my peers in the program for their ideas and support.

Appendix A: Vector and Matrix Operations

Function	Definition
$\vec{\mu} + \vec{v} = \langle (x_{\vec{\mu}} + x_{\vec{v}}), (y_{\vec{\mu}} + y_{\vec{v}}), (z_{\vec{\mu}} + z_{\vec{v}}) \rangle$	Sum of **u** and **v**
$\vec{\mu} \cdot \vec{v} = (x_{\vec{\mu}} \cdot x_{\vec{v}}) + (y_{\vec{\mu}} \cdot y_{\vec{v}}) + (z_{\vec{\mu}} \cdot z_{\vec{v}})$	Dot Product of **u** and **v**
$\lvert\vec{\mu}\rvert = \sqrt{[(x_{\vec{\mu}})^2 + (y_{\vec{\mu}})^2 + (z_{\vec{\mu}})^2]}$	Magnitude of **u**
$k \cdot \vec{\mu} = \langle k \cdot x_{\vec{\mu}}, k \cdot y_{\vec{\mu}}, k \cdot z_{\vec{\mu}} \rangle$	Scaling **u** k times
$proj_{\vec{v}}\, \vec{\mu} = \frac{(\vec{\mu} \cdot \vec{v})}{\lvert\vec{v}\rvert^2} \vec{v}$	Projection of **u** onto **v**
$A \times B = \det \begin{bmatrix} \hat{i} & \hat{j} & \hat{k} \\ A_x & A_y & A_z \\ B_x & B_y & B_z \end{bmatrix}$	Cross-Product of **A** and **B**