3D stepper motor system and its GUI design

A report for project-in-lieu-of-thesis for masters degree

Student Name: Meng Lian

Advisor: M.A. Abidi

Imaging, Robotics, and Intelligent Systems Laboratory

Dept. of Electrical and Computer Engineering

The University of Tennessee

Email: mlian@utk.edu

Table of Content

Abstract

The theory of motion control has evolved since the late 18th century. Simply, motion control is defined as accurately controlling the movement of an object based on speed, distance, load, inertia or a combination of all these factors. Due to high system complexity and difficult software language implementation, the traditional programmable logic controller based motion control systems have gradually been replaced by PC based control systems. In this project, the control to a 3D stepper motor system is accomplished from a PC and an intuitive and easy to use graphical user interface is designed using Visual Basic. The hardware system involved in this project contains an ISA pc card, an indexer, four drive controllers, three linear motion tables and one rotary table which as a whole allows users to implement linear and rotary motion in the x, y, z and θ directions. The purpose of the GUI design is to simplify the control process, save users time from learning a control language and to provide a visualized window for control under the Windows2000 and XP operating system. The functions include basic, real time click-go control and motor status feedback. Automatic parameter calculations of loop motion and hardware system parameter setting windows are integrated into the software. Also the velocity profile on each axis can be created by user clicks on the specified picture area. As part of this project, a data acquisition experiment is included to testify the usability of the GUI and collect human face data that could be used in other face recognition projects. In this experiment, the rotary axis is controlled to conduct 180 degree half circle motion with a 4.5 foot long shaft. In this structure the different modalities of cameras are mounted so that when sitting in the center, a person can have his/her face scanned and different face views can be acquired. As a whole, the stepper motor system, together with its graphical user interface, provides a high-accuracy, easily controlled platform with comparatively low system complexity that is suitable for future applications such as data collection, robot navigation or camera controlling.

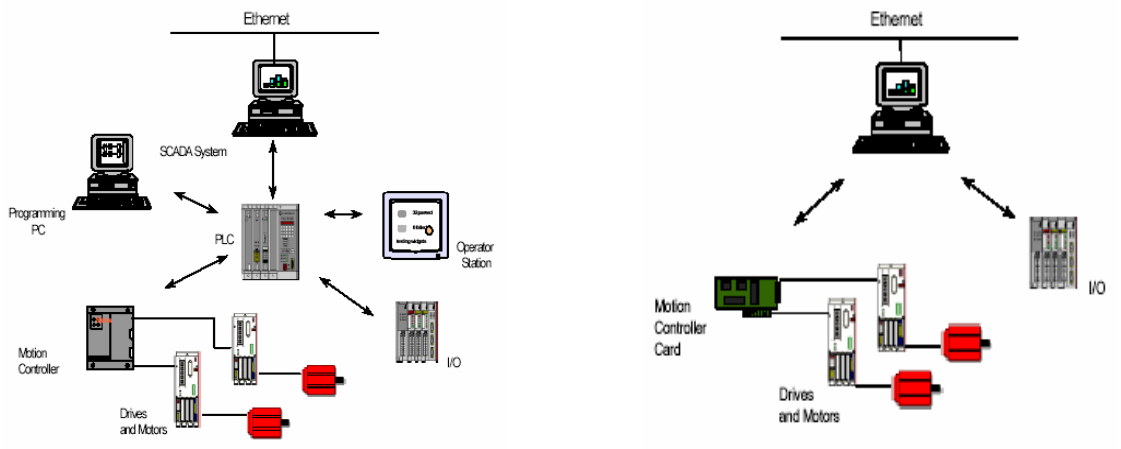
1. Introduction

Since human society entered the industrial age in the 18th century, motion control, especially precision motion control, has steadily gathered attention in terms of research, development and its application to produce innovation. Precision motion control, in electronic terms, means to accurately control the movement of an object based on speed, distance, load, inertia or a combination of all these factors. Driven by the requirements for much higher product performance, higher reliability, longer life and lower cost, numerous advances have been made recently, especially with the help of digital computers. With the emergence of nanoscience at the end of the last century, technology in this field advanced a higher level. Today, high precision motion control has become an essential requirement in advanced manufacturing systems such as machine tools, micromanipulators, surface mounted robots, etc.

Generally, motion control systems can be separated into several parts: the mechanical device being moved, the motor (servo or stepper) with or without feedback, motion control I/O, motor driver, intellectual controller unit, and programming/operating interface software. Traditional motion control systems utilize PLC technology to fulfill the control task which typically comprises a number of hardware and software elements: PC for process visualization, hard PLC with coprocessor cards, I/O via field bus, motion control via parallel cabling and a selection of software operating systems and programming languages. The standard PLC system is shown by the Figure 1. (a).

Widely used in electrical systems, the PLC based control system has many advantages. First, the photoelectric isolation on all I/O modules, self-diagnosis function and the redundant system achieved by the dual-cpu in large scale PLC systems lends the whole platform high reliability. Additionally, the PLC system has abundant I/O interface module capabilities according to different signal requirements so that it is adaptive in various situations. However, despite its advantages, the challenges of the traditional motion controller system are many. As shown in Figure 1, the PLC system with motion control has a number of separate components that are required for each individual function. For instance, PLC is used for logic control; the motion controller platform is

used to control the motor drives; a programming PC runs the various configuration software packages; an operator interface provides graphic information; separate software packages are used to program the PLC, motion controller, and operator interface. In many cases, an additional PC is required to run supervisory control software or data acquisition (SCADA) to collect or filter production data [1]. Generally, the PLC control system contains multiple hardware platforms, multiple databases, and multiple software packages to program and configure each part of the hardware which causes complexity in connecting components and exchanging data and commands between them. Such complexity notably prevents PLC from being used in recent highly integrated industrial applications. In recent years, as microprocessors made computers less expensive and more competitive in performance, the shift from PLC to PC-based control has been accomplished for applications requiring single axis or multi-axis coordinated motion control by combining PLC-based logic and motion control into a single PC platform with common programming language [1]. The PC control system takes advantage of clear visualization, easy data manipulation, storage and reporting characteristics, which may be difficult to implement in PLC. These advantages lead to the fact that numerous small scale PC control systems are in operation today, especially in the areas of data acquisition, process monitoring and batch control. The sample composition of a PC control system is shown in Figure 1(b).



(a) PLC motion control (b) PC-based motion control

Figure 1.1 Comparison of PLC and PC-based motion control [1]

In motion systems, the motor is one of the key devices for the reason that it provides brute motive force. There are many types of electric motors, each with its own power drive requirements. Three common types are: alternating current (AC) servo motors, direct current (DC) servo motors and stepper motors.

AC servo motors operate from alternating current power resources. They can be inexpensive to build and operate. They are reliable and usually operate at standard line voltages and frequencies. However the control to the speed of AC motors is difficult to implement and limits AC motors to most simple motion.

DC servo motors are suitable for complex motion tasks because the speed and torque in DC motors are easy to control by varying the voltage and current. Yet, due to the feedback characteristic, motion systems using this kind of motor are expensive and highly complex. This is also the main reason to use a step motor when considering the trade-off of price and system performance.

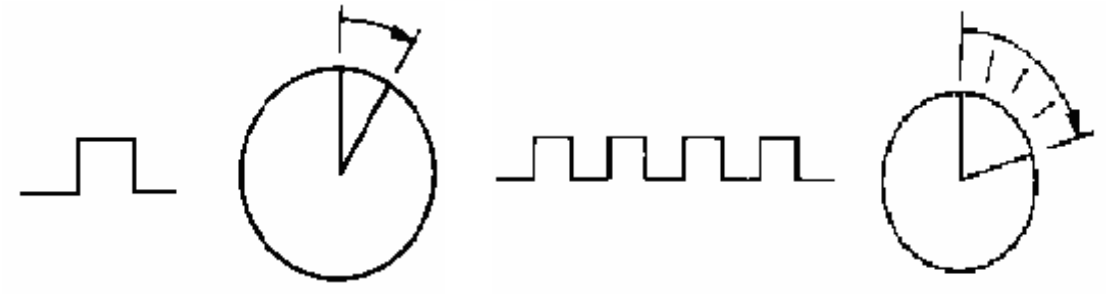
After many years' innovations in the attempt to provide higher resolution and lower cost, the step motor has become a popular solution for achieving controllable motion due to its unique feature of the output shaft rotating in a discrete number of steps. Now advanced stepper drives can provide microstepping, a relatively new stepper motor technology that controls the current in the motor winding to a degree that further subdivides the number of positions between inner motor poles. This greatly increases the resolution of the stepper motor while at the same time retains the comparatively low price. When combined with a suitable controller, a step motor system can be tailored to meet the requirement of a wide variety of applications, such as X, Y, Z positioning and rotary indexing tables; the accuracy makes step motors particularly attractive for scientific and laboratory applications. Furthermore, they provide a simple open-loop technique for positioning with easy configuration and no need for tuning. Also the fact that step motors are brushless means the maintenance requirements are minimal and the relatively high torque produced by these motors makes the system stable when stopped.

Other benefits such as simplicity and low cost make step motor system even more competitive in small control systems.

The remainder of this report is organized as follows: In chapter 2, an introduction to the step motor system is provided. A brief working theory and classification of step motors will be included in this chapter. Chapter 3 mainly describes graphical user interface design essentials including those does and don'ts, followed by the 3D motor system (from Compumotor Division, Parker Hannifin Corporation) specifications and interface implementation in chapter 4. Chapter 5 shows the system setup for the human face data acquisition experiment. Conclusions and references are in the last of this paper.

2. Technology of the step motor system

Step motors (sometimes also known as stepping motors, or steppers) can be defined as "electromagnetic incremental-motion actuators which convert digital pulse input to analog output motion" [2]. Thus, step motors can be viewed as electrical motors that are driven by digital pulses rather than a continuous voltage or current. When used in a open-loop control design, step motors translate a train of pulses into shaft revolutions and each pulse equals one rotary increment, which is a portion of one complete rotation. This process is clearly shown in Figure 2.1.



(a) One pulse equals one step (b) Pulse count equals step count

Figure 2.1 Theory of the stepper motor system [3]

Unlike DC or servo motors, step motors are synchronous devices. Any torque generated by the system is only the result of the load applied. These motors rely on input signals to step the rotor through discrete angles. A step motor system often contains three basic elements, as shown in the following figure:

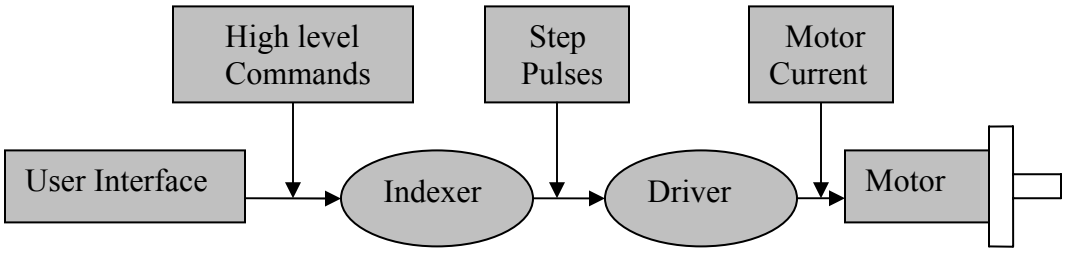


Figure 2.2 Stepper motor system overview

The indexer (also called the controller) is a microprocessor that directly processes the sophisticated high-level commands and provides step pulses and direction outputs to

the driver. Usually, it is a programmable device and the programming process is accomplished with hardware switch settings. Most of applications require that the indexer manage other functions including acceleration, deceleration, drive resolution and distance as well. Microprocessor-based indexers offer flexibility that they can operate in either stand-alone mode or interfaced to a host computer.

The main function of the driver (also called the amplifier) is to convert the indexer output signal into the power (voltage or current) necessary to drive the motor winding. One step pulse from the driver is required by one step advance of the motor shaft. In other word, the speed and torque performance of the step motor is based on the flow of current from the driver to the motor winding. Inductance, the most important parameter of the driver, refers to the time it takes for the current to energized the winding. In most industrial applications, driver circuits are designed to supply a greater amount of voltage than the voltage that drives the motors.

Generally speaking, the step motor itself consists of two parts, the stator and the rotor. The winding is the main part of the stator. Three-phase, four-phase and five-phase step motors have three, four and five windings, respectively. When in the working mode, the windings are energized in a specific order that is called the phase sequence. The rotor mainly composes a magnetic shaft. When the windings are energizing-deenergizing under the effect of the phase sequence signal, an electromagnetic field is generated around the rotor. Hence, the rotor starts rotating driven by the regular varied electromagnetic force.

Precision is probably one of the most important requirements to justify a step motor and is mainly determined by the numbers of step per revolution. Usually a step motor can work under full, half and microstep work modes which are dependent on the design of the driver. Normally full step mode is achieved by energizing both windings while reversing the current alternatively. Essentially, one digital input from the driver is equivalent to one step. The standard step motor has 200 rotor teeth, which means 200 full steps per revolution, or 1.8 degrees/full step angle. In half step mode, one winding is

energized and then two windings are energized alternately, causing the rotor to rotate half the distance, making the half step angle 0.9 degrees. Furthermore, microstepping is a new technology that emerged recently and it controls the current in the motor winding to a degree that further subdivides the number of positions between poles. In most of the advanced control systems, microstepping technology is used to achieve smaller step angles and higher accuracy. On the whole, the fractional stepping method effectively lowers vibration that can be exhibited when step motors are working under full step operation at certain speeds. It also reduces the amount of jumpiness that is inherent in running in full step mode.

Depending on different classification methods, there are several types of step motors. Basically, from the point of view of the design method and working theory, there are three types of step motors that are more popularly used than others: the variable-reluctance type of step motor (VR), permanent-magnet step motor (PM), and hybrid step motor.

2.1 Variable-reluctance (VR) step motor

The variable reluctance step motor is characterized as having soft iron multiple rotors and a wound stator. The following figure shows the simplest design of VR step motors

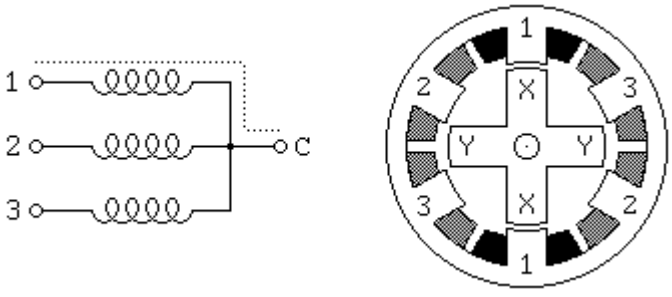


Figure 2.3 VR stepper motor [4]

As shown by the Figure 2.3, the rotor (inner part) in this motor has 4 teeth and the stator has 6 poles, which each winding wraps around two opposite poles. When winding number 1 is energized, the x teeth are attracted to poles numbered 1. When winding 1 is

turned off and winding 2 is turned on, the rotor will rotate 30 degrees so that the y teeth will line up with poles numbered 2. Simply applying power to 3 windings in the sequence will make this rotor rotate continuously. Based on the difference in construction methods, there are two basic types of VR step motor. The first one is a multiple-stator-stack VR step motor. Motors usually come in the form of three-phase or four-phase construction. The three-phase, multiple-stator-stack, VR step motor has three electrically and magnetically independent stator and rotor segments mounted on a common shaft. In order to make the motor rotate, the stator or rotor must be radically offset from each other. Most of the early VR step motors are of this type of construction. However, in recent years, the multiple-stator-stack, VR step motors have become less popular compared to single-stator-stack types due to their high cost of manufacturing and inefficiency. Compared with multiple-stator-stack, VR step motors, the single-stator-stack step motor is more compact in size and simpler to construct. However, only certain step resolutions can be obtained with this arrangement by specific combinations of the rotor and stator teeth. Realistically, it can provide step angles that range from 1.8 degrees to 30 degrees. The biggest difference compared with multiple-stator-stack, VR step motors is that, in the single stack version, the winding of each electrical phase shares the same lamination stack, whereas in a multiple stack motor, each electrical phase is wound on a separate stack length of laminations. This construction method lends single-stator-stack motor characteristics such as:

- Small step angle possible
- No holding torque with the winding de-energized
- High slew speed possible.

2.2 Permanent-magnet step motor

The permanent-magnet step motor is also referred to a “can-stack” motor. Usually PM step motors incorporate a permanent magnet rotor (referred to PMR), coil winding and magnetically conductive stators as shown in the Figure 2.4. When the coil winding is energized, an electromagnetic field is formed on the stator which causes the rotor to align itself with the field. The magnetic field can be altered by sequentially energizing the stator coils which generate rotary motion.

Another PM type step motor is the permanent-magnetic-stator (PMS) motor. The difference is quite obvious. The rotor of this kind of motor is a toothed, soft iron structure that completes the magnetic circuit

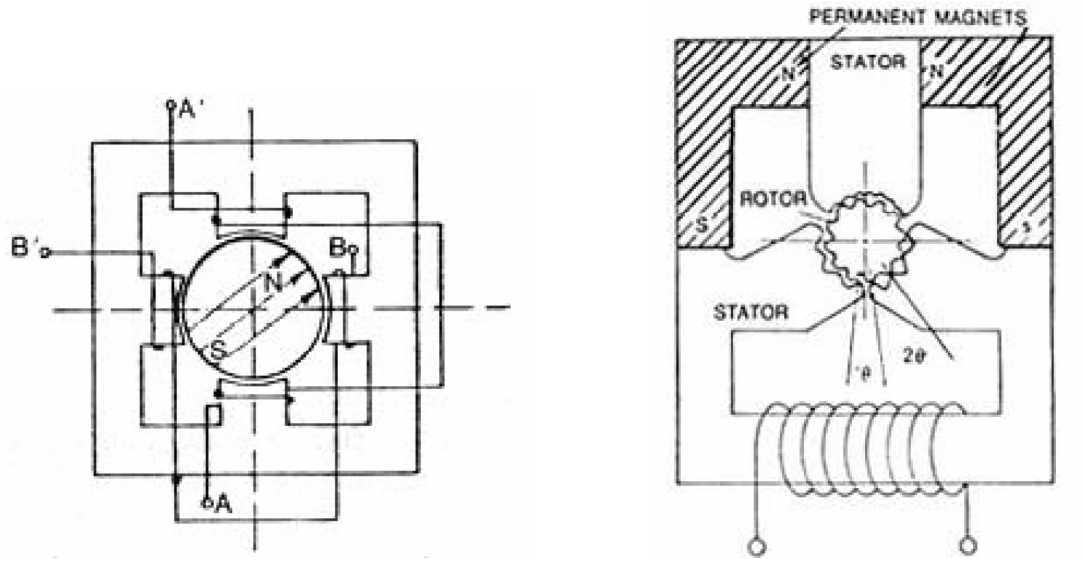


Figure 2.4 PMR and PMS stepper motor [5]

The most commercially realistic types of PM motor are in two-to-24-pole configurations, which means the motor has 24 poles on the rotor and two stator sections. PM step motors are characterized by their simplicity in construction and are low in cost. Usually they are only used for low-speed applications due to their relatively low torque.

2.3 Hybrid step motor

Besides the two step motor types discussed previously, hybrid step motors are the most widely used in all variety of industrial applications. Hybrid motors combine the best characteristics of the variable reluctance and permanent magnet motors. They are constructed with multi-toothed stator poles and a serrated permanent magnet rotor. When energized, electrical current in the coils around each stator creates an electromagnetic pole in the stator and the teeth in the rotor line up with the serrated teeth in the stator. Compared with the other two types, hybrid step motors offer finer resolution. Standard hybrid motors have 200 to 400 rotor teeth and rotate at 0.9- 1.8 step angles. They also exhibit high static and dynamic torque.

3 GUI design essentials

A GUI (graphical user interface) can be defined as a program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. Well-designed graphical user interfaces can free the user from understanding intricate theories, learning complex command languages, and implementing complicated algorithms, etc. According to the encyclopedia, the first graphical user interface was designed in the 1970s by Xerox Corporation's Palo Alto Research Center. But it was not until the 1980s and the emergence of the Apple Macintosh that graphical user interfaces became popular. One of the main reasons for their slow acceptance was the fact that they require considerable CPU power and a high-quality monitor, which was not cheap until recently.

Nowadays, the GUI has become a widely accepted standard. To some extent, the operating system, such as Windows 2000 or Macintosh, can also be looked at as a large GUI. From the user's point of view, the interface stands for the software. Since an intelligent interface allows users to perform tasks in their own ways, it must be easy to learn and use. Despite their popularity, there are still some basic principles for all good interfaces that only are exhibited by a few programs. Common problems that are easily neglected by the designer can be summarized by the following three aspects:

1. Forgetting the user

Developers often design for what they know, not what the users know. This problem occurs in many areas but even is more fatal in the interface design because it immediately makes the user feel incapable of using the product.

2. Controlling the user

One evidence of the designer's preference for control is to continually attempt to gray or blacken items during user navigation. This is contradictory to event-driven design in which the user rather than the software dictates what events will occur.

3. Too many features at the top level

Sometimes the designer's attempt to put everything on the first screen not only prevents users from using a desired feature, but also makes the interface look disorganized and disturbing. A good GUI with an abundance of functions may only have the most frequently used features on the top window, with most others hidden in a drop-down panel, accessible when needed.

From the designer's point of view, to avoid such problems is quite critical when making the interface friendly and useful for the user. As a matter of fact, the GUI is the type of software that makes the computer transparent to those users who may not want to be a programmer. The best way to achieve this transparency is to understand the user's own model of what the task requires and interpret that into the user interface. It requires more than graphic design skills to create quality user interfaces. Clarity and consistency are important factors at the design time. Using some general representations such as "File", "Edit" or "View" is always a good way to provide convenience for those who have prior knowledge from other successful applications. Also visual or acoustic feedback maybe very helpful to let people how much longer their operation will take.

In order to make the final product meet the user's expectations and requirements, there are three necessary phases that constitute the process of intelligent interface design [6]. They are: Analysis, Design and Construction. Although these names can also be used to refer to software development, they have different meanings and steps when referring to interface design [7]. Their relationship is shown in Figure3.1.

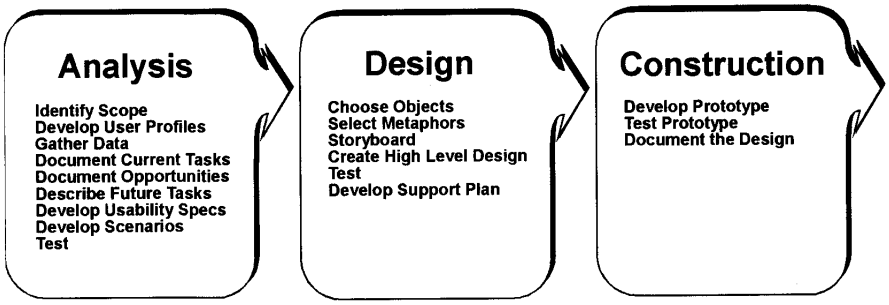


Figure 3.1 The phases of intelligent interface design [6]

Analysis

The purpose of the analysis phase is to document and verify information about people who will be using the interface, their current tasks, their concepts and terminology when performing work, requirements for the interface, and vision of their future work when the new interface is in place. Work in this phase can be further divided into four steps:

1. Identify the current state and scope. The designer may not be the first person to do the analysis work, so it is critical to know what has and has not been done to avoid redundancy. Additionally, to decide the scope of the analysis means to identify high-level user activities, not the concrete flow or steps when performing tasks. The design constraints, for example, from the hardware, software, programming environment and project features, should also be taken into account.

2. Develop user profiles. As mentioned earlier, one of the common mistakes at design time is for the designer to assume users know what they know. User profiles are documents that help the designer understand those important design trade-offs, for example, the user experience with hardware and software environments that the project will use; user experience with the kind of software that the project will develop; user task experience; expected frequency of use and job turnover period. If the turnover frequency is quite high and frequency of use is low, the user will never be an expert with the software. Thus, the ease of learning feature should be more important. Otherwise, the designer should pay more attention to ease of use.

3. Gather data and document tasks, problems and opportunities. This is a more detailed step compared to step 1. The designer is now able to gather information that contains important clues to what the interface organization and conceptual design should be. One must also clarify how much of the future work is going to be ideal and how much is going to conform to constraints.

4. Develop usability specifications and user scenarios. A usability specification is defined as “usable or user-friendly for the interface” [6]. It provides powerful means for communicating what usability, easy-of-use, and user-friendly really mean for a software and its interface [6]. Many key attributes are included, such as ease of learning, rapid task performance, accurate task performance and perceived ease of use.

A user case scenario is an outline of tasks that describe how users will do their work; the purpose is to give the designer a hand in conceptual design so that the clear flow of screens can be developed. In order to create the best flow with good efficiency, information about frequency of tasks must be documented. Additionally exceptions and critical tasks should also be noted in these scenarios.

Design

As shown in the Figure 3.1, the next step after analysis is design. In this phase, the designer takes everything that has been learned from analysis and creates a high level construction. It is important for the users to be able to predict what will happen next and decide what they will do next. They need to have a mental model of how the software works. Hence, a conceptual model needs to be created for the screen where users will perform their tasks, and then translate into appearance and organization.

The main objective of the design phase includes identifying the user objects, actions and metaphors that should be represented in the interface so that the user can understand the interface.

1. Choose major user objects. Major user objects are those that the users have to manipulate as they move through their workflow. These are usually related to underlying software objects or the objects described in analysis and design. It’s of the most importance that those objects that are critical and frequently used are obvious and clear to the users.

2. Select Metaphors and Representations. Metaphors are the tools designer use to "link complex software with the user's world and are visual or conceptual representations of major objects and their associated actions" [6]. It's helpful when the interface contains any functions new to the users. A simple form can be seen as a metaphor but it's critical for the designer to decide which represents the underlying functions best. Thus the designer must make sure the metaphor supports the user's primary assumption and reflects the major objects.

3. Create a high-level interface design. So far, the designer holds a scene in his mind about objects, metaphors, and so on. Thus, the task is to convey these ideas into things such as style, the main window and mockups on the computer screen. In effect, the first thing to do is to select or adapt a style for the whole interface. The next few steps are concerned with the main windows. Main windows provide not only a set of user-performed actions, but also one or a few recognizable home bases. As a consequence, main windows and related user actions should be identified to represent those major user objects and major actions. Also, it should be decided how flexible the interface is for different users and tasks.

The last point in high-level design is to review and revise. That is, improve the interface before trying it out with users. This includes identifying conceptual design problems such as confusing objects or action names, high-level design problems such as confusing organization of windows and task flow and interface designer misunderstanding about system scope and design constraints.

Construction

The final step of GUI design is construction. Now the designer has identified the users information, their current working concept and requirements for the GUI. The major objects and representations have been selected and the designer already has an idea in mind about the GUI style. The remaining task is to construct a detailed and realistic version of the interface. The main processes in this step can be included: (1) create a computer-based prototype, (2) further iterate design and test with users, (3) document the complete and final design.

4. System integration and GUI design

This and the next chapter conclude my accomplishments to this project. In 4.1, I will describe how individual hardware components are combined to form the complete motion system. The process of GUI design and functions is presented in Chapter 4.2.

4.1 Hardware description

The 3D multi-axis step motor system used in this project consists of the following components:

- Four S57-102 series rotary stepper motors
- Three linear motion tables (100 series) and one rotary motion table (300 series)
- Four Digiplan PDS series controllers
- One AT6400-Aux1 step and direction indexer
- One AT6400 PC ISA card

The motor, controller, indexer and the PC card are manufactured by Compumotor Division, Parker Hannifin Corporation. The positioning tables are from the Lintech Company. The motors are physically mounted on the positioning tables with transmission shafts. The terminal user controls the motion of the stepper motor by sending parameters and commands through the controller using a set of particular syntax defined by the Parker Hannifin Corporation. The complete system connection is shown in Figure 4.1.

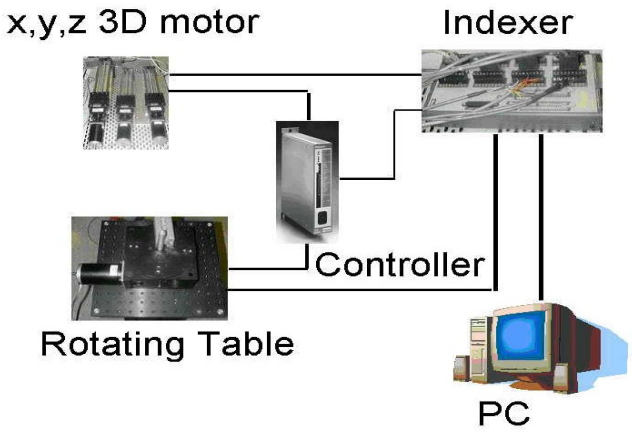


Figure 4.1 Stepper motor system

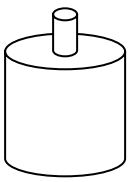
This is a typical step motor system as described earlier and contains regular components such as an indexer and driver (same as controller). A PC card is installed in the host computer functioning as a connection between the hardware system and the computer. Here, a high level command is translated to machine language that is understandable to the indexer. Step pulse output is generated in the indexer and sent to the driver. Finally, the analog current from the driver is input to the step motor to produce motion. The system delay is confined to an unnoticeable scale so that real time control can be achieved. The main features of this system are:

- 3D system with freedom in the x, y, z directions
- Base rotation function provided by the rotary motor
- Total travel length on linear positioning table: 10 inches
- Highly accurate. The minimum step: 1.05 microns (linear table), 0.0814 degrees (rotating table).
- Maximum speed: 5 inches on linear axis (120000 steps), about 20 degrees on rotary axis (40000 steps).

These step motors are of permanent magnet design. The technical specification of the motor is as follows:

Compumotor S series rotary step motor (model: s57-102):

Dimensions:



Length: 4.0 (+0.72) inches

Radius: 1.118 (+0.084) inches

Step pulse input:

200 nanosecond pulse minimum

40% duty cycle

Accuracy: 5 arcminutes (0.0833 degree) typically (unloaded)

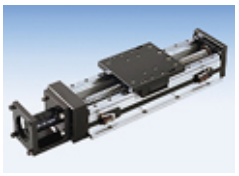
Repeatability: 5 arcseconds (0.00138 degree) typically (unloaded)

Hysteresis: < 2 arcminutes (0.0334 degree) unloaded

Maximum speed: 50 rps

Lintech 100 series linear positioning table:

Dimensions:

 16.0x3.188 inches

Travel Length: 10 inches

Load Capacity: 703 kg within 50 km of travel

238 kg within 1270 km of travel

Orthogonality: < 30 arcseconds in multi-axis system

+ Lintech 300 series rotary positioning table:

Dimensions:

 6x6 inches

Table Top Radius: 3 inches

Load Capacity: 102 kg within 1 million revolutions

45 kg within 1 million revolutions when inverted

Maximum Speed: 10 rpm

4.2 GUI design process

This subsection describes the process of GUI design of the 3D stepper motor system and will include some sample images.

4.2.1 Introduction of motion architect

As mentioned earlier, the system manufacturer provides a set of software called motion architect. It is a windows-based application development system to help design, develop and debug programs for the 6000 series motion controller. This software can be downloaded from www.compumotor.com. The manual is located under

http://www.compumotor.com/manuals/software/MAmanual.pdf. The software provides functions including system configuration, combination code generator, program editor, terminal emulator, and program tester. With communication to the PC card and the indexer, the program can be used to control the system upon selection of the type of motor controller. The generator automatically generates controller code for the basic system set-up parameters such as distances and speeds. The editor is used to provide a programmable environment to create blocks or lines of code. The source code files with a prg extension can be downloaded to the bus controller for immediate execution. Copying the code contents to the terminal window can also reach the same goal. The most frequently used tool in the set is the terminal emulator. Similar to the DOS command window, the terminal module allows users to type in and execute controller code with direct communication with the hardware. Another function is that code files can be transferred to and from the controller. Lastly, the test panel provides users a function to customize the panel with multiple windows to monitor controller output and programmable buttons for input. The sample picture of this software packages is shown in Figure 4.2.

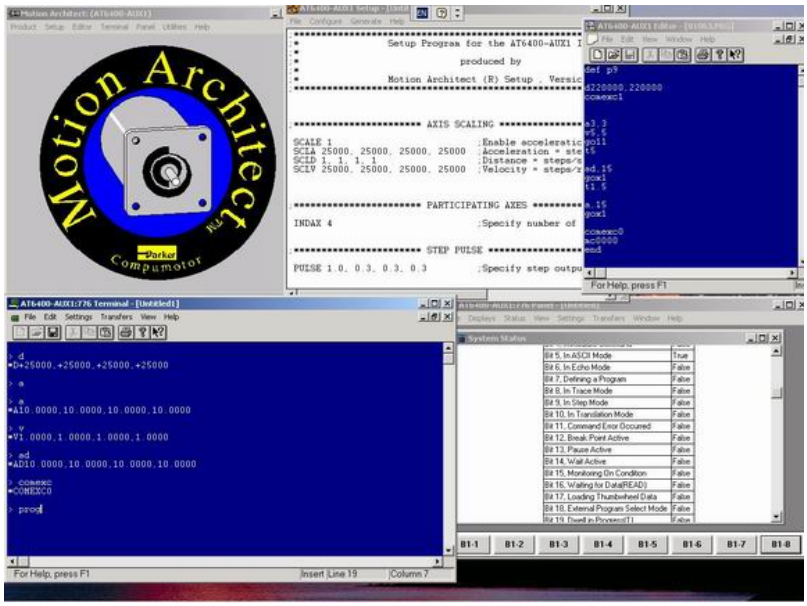


Figure 4.2 Motion architect interface, windows (from top to bottom, left to right): Main window, code generator, editor, terminal and test panel

Overall, the motion architect provides a set of tools to help users achieve basic control of the motor system. However, despite its simplicity and ease of use, the deficiencies are obvious. For instance, the equipments in the IRIS lab are configured with three linear motion axises and a 360 degrees rotating table to form a 3-dimensional system with freedom in the x, y, z directions and an angle (θ) which specifies rotating degrees. To control such a complicated system for a given distance and speed, a more intelligent and convenient tool is needed to avoid typing hundreds of commands at the terminal windows to conduct a loop motion. Also the motion architect software was developed in the 90's at the early stage of computer-aided design, which led to interface development in Windows32 style. Although a patch for the latest version provides compatibility to the windows 9x and 2000, the interface still looks out of date and some polishing work needed to be accomplished. Hence, the objective of GUI design is to make it suitable for complicated motion control and provide an intuitive and visualized window.

4.2.2 New interface design

Programming of motion control software, together with interface design, is accomplished in Visual Basic 6.0 under Windows 2000. VB was chosen because it is easier to learn and easier to use in interface designing processes. By using user selected parameters or settings, the software will generate a series of sentences in MCL (motion control language) syntax. They are downloaded to the motion controller by calling a function: SendAT6400Block (%device address, $command, 0). There are several dynamic link library files that are provided by the manufacturer involved in this process. The most important ones, nt6400.dll and win6400.dll, have to be put in system folders to ensure the proper operation of the program. The other necessary files include winrt.sys and at6400.bas. In this case, VB is more suitable than other programming environment such as VC++ because no bottom level communication with hardware is needed.

According to the Chapter 2 GUI design essentials, there are three design steps. When matched with his project, the results in the analysis and design steps can be summarized by the following description. The users of this GUI are the graduate or

undergraduate students working in the IRIS Lab and they should have a background in engineering and computer operation. The GUI aims at dealing with applications such as using one or more axises to achieve precision control of a robot or mount various kinds of cameras to capture 3D image data. Most of the time, the real time high precision control of the distance and speed parameters are of more concern than other functions. Thus, these two factors should be considered as major parameters at design time. Also the style of the interface is selected to comply with commonly used Microsoft Windows products. XP-style buttons are applied in each sub window. The main window contains most, frequently-used functions including the current status of each axis, click and go button and distance setting. At the same time, the problem of "too many features at top level" is avoided and the interface looks clear so that the user will not feel confused at first glance.

There are several individual windows in the GUI. In Visual Basic, each window corresponds to a file with a frm extension name. Among them, much time and endeavor were devoted to the main window (AT6400.frm), loop control window (ln.frm) and speed control window (spdctrl1-4.frm). In order to make the GUI clear to other students who are totally new to the 3D system, I will describe how this GUI works step by step and make code comments in appendix. Figure 4.3 shows the flowchart for this software.

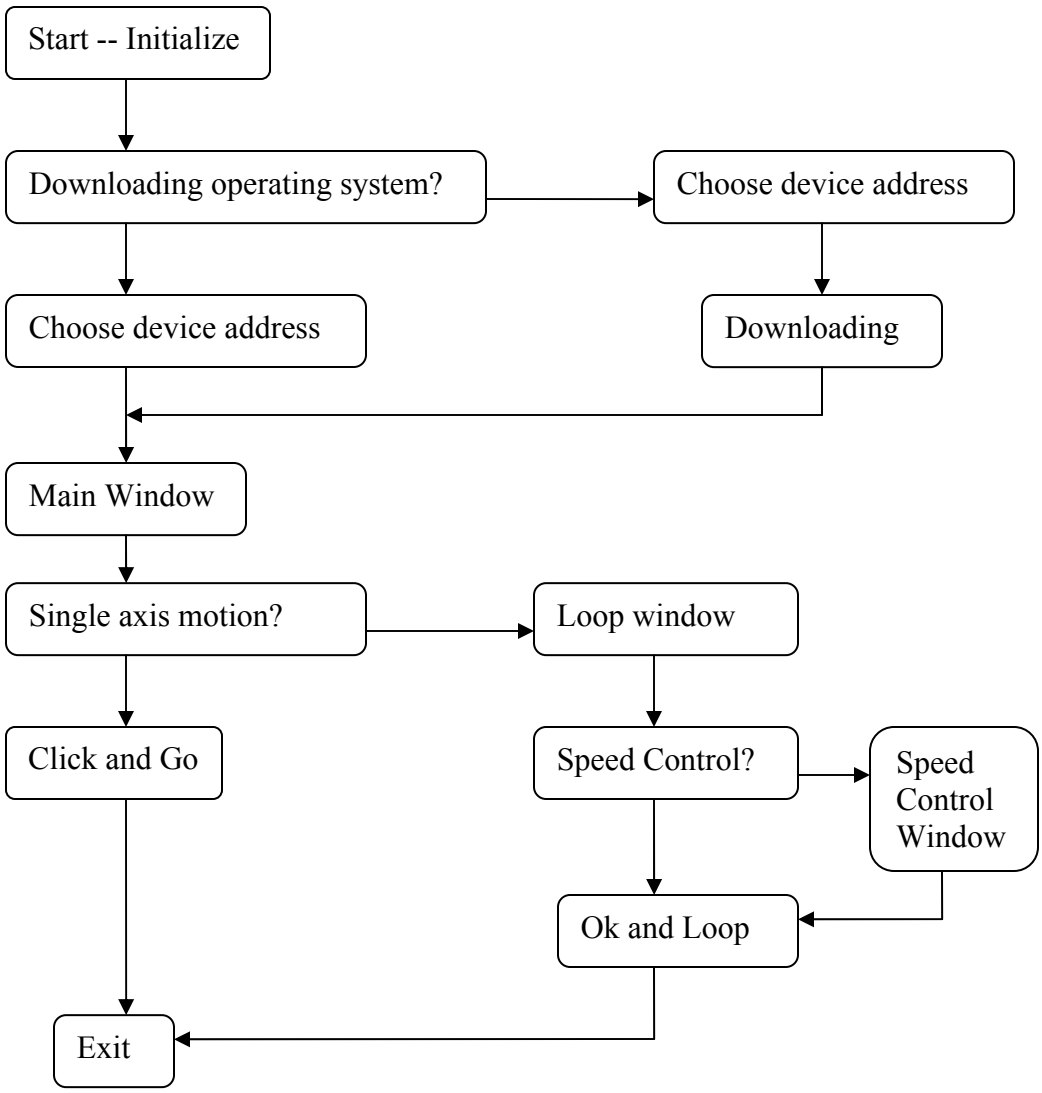


Figure 4.3 Flowchart of the GUI

Step 1

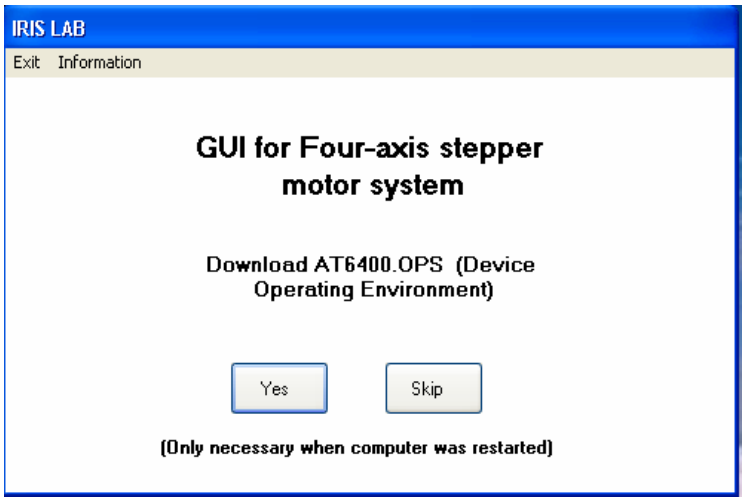


Figure 4.4 Initializing window

Figure 4.4 is the first window shown after launching the executable file. The user will be asked to make a choice if downloading the device operating environment is needed. This downloading is the prerequisite step to connect, program or send commands to the controller and should be done every time the host computer is restarted (i.e. when the LED on AT6400 indexer is red). After downloading, the LED will turn to green. The device environment file AT6400.ops should be put under c:\windows\system path.

Step 2

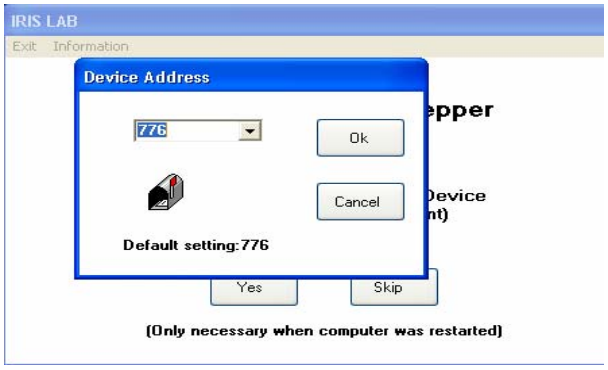


Figure 4.5 Address setting

No matter what choice the user selects in the previous window, the next window will ask the user to set the device hardware address, as shown in Figure 4.5. This address is where the users want the computer to find the PC card and should match with the switch setting located on the PC card (DIP Switch sw1). This varies on different

computers. In the testing computer, it is set to 776 (decimal). Occasionally, a card controller error message will occur due to a conflict in memory or specified address dominated by other peripheral cards. If there is no conflict in the computer system, the user can download the environment file from the terminal window of the motion architect or simply restart the computer. Otherwise, a new I/O address needs to be allocated to the PC card.

Step 3

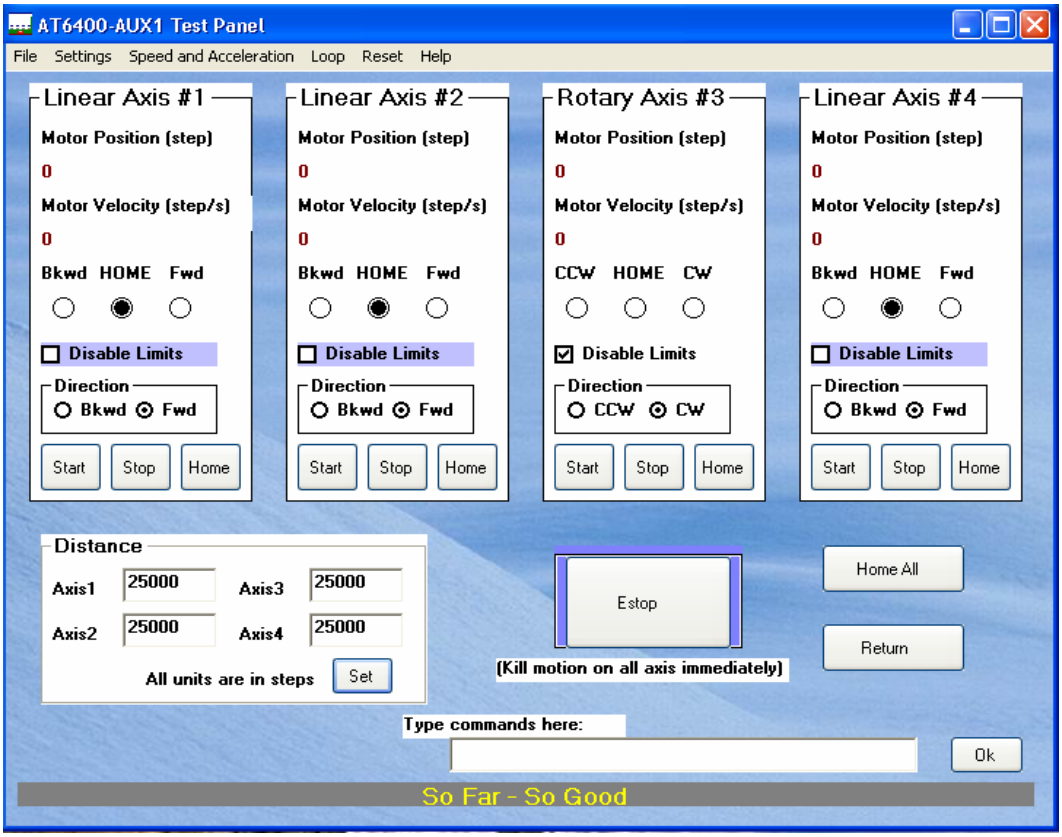


Figure 4.6 Main control window

Figure 4.6 displays the main window of the GUI developed. If communication has been set up with the device, the load position status will be shown on the screen. There are four columns which dominate the upper part of the interface. Each corresponds to a motion axis. Load position and moving speed of each axis are shown on the screen. Three end-of-travel hardware limits are located in each linear table (none in the rotary table). They are backward, home and forward. The backward and forward switch is located at the end of the axis and the home switch is close to the backward switch. Usually this is

set as original starting point (also referred to home position). Three buttons at the bottom of each column simply control the single motion behavior of each axis. The units for the motor position are in steps. This complies with the default manufacturer hardware settings. The complete moving distance of each linear table ranges from approximately –5,000 to +235,000 steps (about 8 inches, hence 30,000 steps=1 inch). All limits are default enabled which means when the load hits the end limit, continuing motion in the same direction will be inhibited.

From the main window, the user can implement basic movement control of the system. The next moving distance is easy to set through the distance frame located at the bottom-left corner. Also if the user has learned the system's motion control language, more commands can be issued through the command textbox. The "Estop" button is used to kill the motion on all axises immediately and the "Home all" button returns all axises to their original points. In sometime particular cases, for instance, the user disables the hardware limits and sets the motor to continuous mode (the motor will keep turning at a constant speed until a stop command is issued), the program will disregard the limit and the motor will keep spinning even if the load has reached the end of the table. In such a case, the position shown on the screen keeps increasing as the motor spins. Thus it is no longer accurate. Then the "Return" button is used to clear the encoded position to 0 while the load does not move.

Step 4

Other than distance, there are more parameters such as velocity or acceleration rate that users need to change. They can all be accessed from the menu. More settings such as the pulse width and driver resolution are under "setting". According to the manufacturer's manual, pulse width is defined as the time the pulse is active. In order to drive the motor in our system (model s57-102), this parameter should be no less than 1.0 μs. Another setting is driver resolution, which is how many steps the motor will advance per revolution (default 4000 steps/rev). In fact, values in the speed and acceleration window are all comparative values. They must be multiplied by driver resolution to get real values.

Step 5

As a major advantage compared with motion architect, the loop motion with specified speed control is an important part in this GUI. This feature simplifies the whole process by embedding parameter calculation and hundreds of commands can be downloaded into the program. The user needs to specify several values, for instance, starting position, ending position and loop times, to achieve loop control. For some simple cases, three speed profiles has been included into my GUI, as shown in the Figure 4.7.

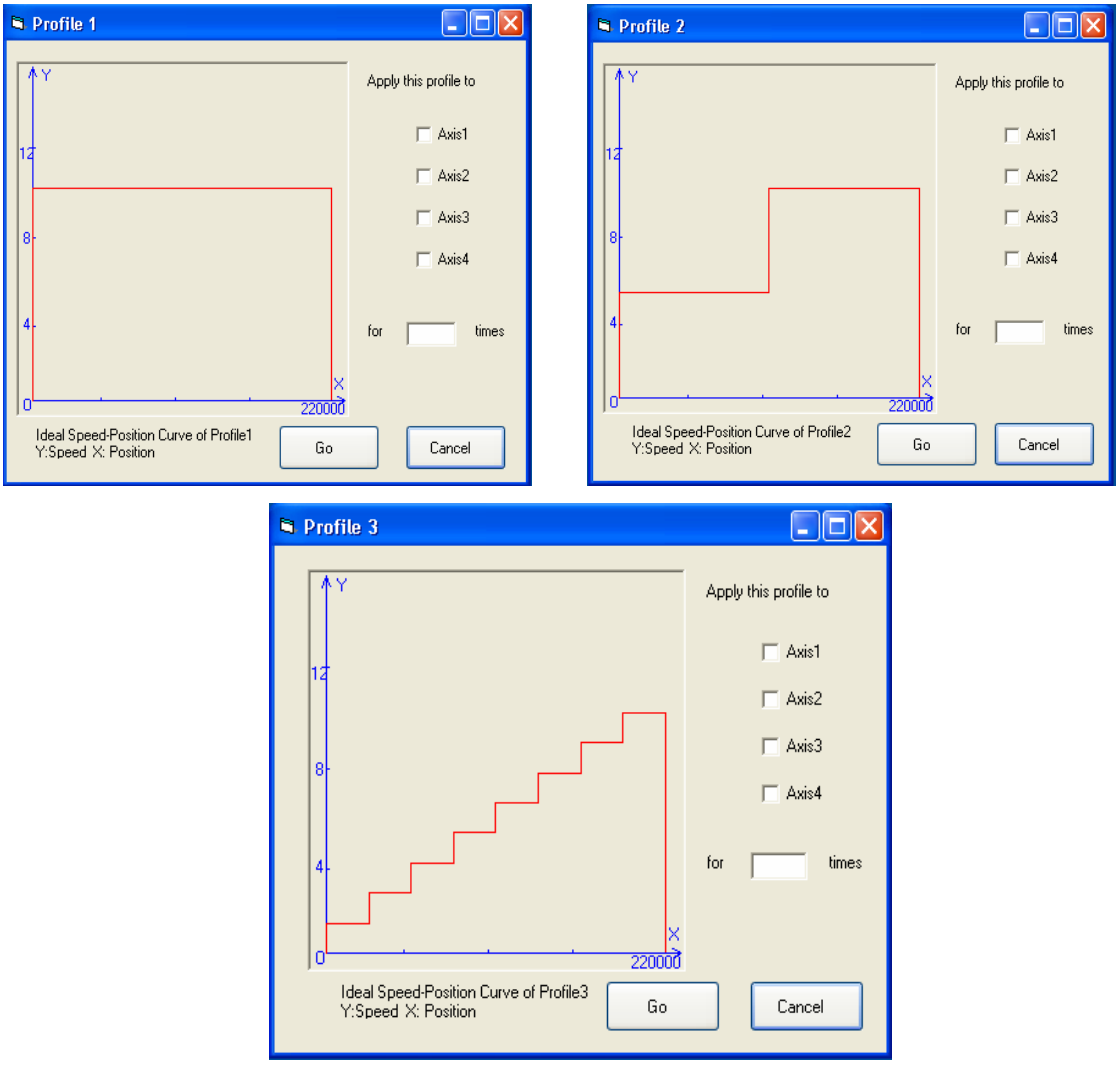


Figure 4.7 Simple speed-position profiles integrated in GUI

For simple speed requirements cases as shown above, the user needs only to click on the axis on which he intends to apply the profile and input a loop time. The motion will be conducted on the corresponding axis.

The window shown in Figure 4.8 is the customized loop control window. Among the inputs, “times” is the key value that decides if motion is going to happen in an axis. The first time check on “copy” will set the parameters for that axis including starting and ending position, loop times and speed curves into a sample buffer. The second time check on “copy” of other axises will copy every parameter in that buffer into corresponding axises so that movements in both axises will happen simultaneously and with the same speed and distance. If no speed curve is specified, the value in that velocity slider bar will be used as default. The user can also set the time interval between every time the load reaches its end and starts motion in the contrary direction.

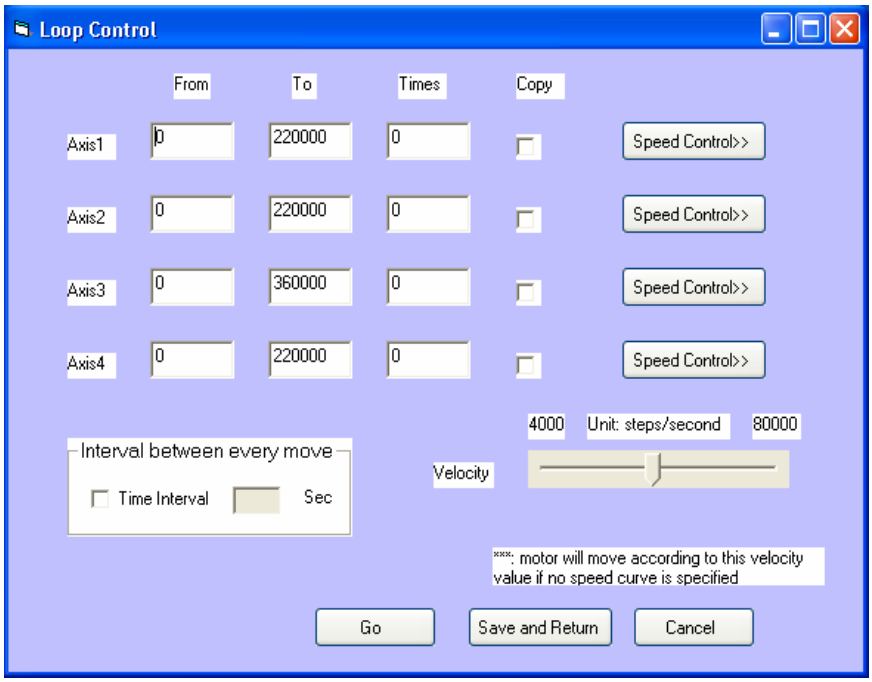


Figure 4.8 Loop motion control window

Step 6

In case the user needs to specify a different moving speed for different parts of the motion, a speed control window for each individual axis was designed. First, a work mode needs to be specified to decide which action among adding, deleting and dragging a point is about to happen. The picture box allows the user to set a speed-position curve.

The x-axis is the motor position and y is the moving speed. In add mode, the user can click and add a new point anywhere in the picture box as shown in Figure 4.9. Once a mouse click is detected, the program will line up this point with its nearest neighbors (if this is the first point, the program will connect it with the original and the maximum point). At the same time, a new record with the coordinates of the new point will be added to the list box on the right. p means position and v means velocity. The preset maximum motion range is 220,000. Accordingly, the position shown in the list box is in that scale. Practically, when the user needs to change the whole moving distance to other values like 100,000, a scale factor will be automatically multiplied to generate the acceleration rate and time.

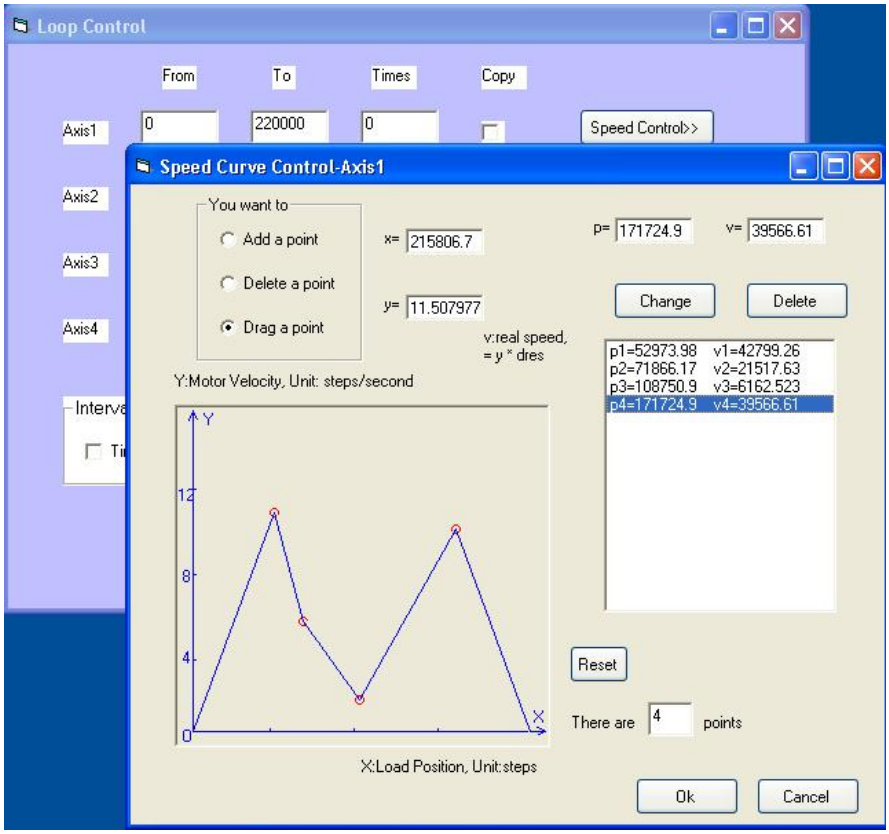


Figure 4.9 Speed control window

The other two modes are “delete” and “drag”. In the delete mode, the user needs to click on the points within the range of a red circle. The previous lines that connect the point to its nearest neighbors are replaced by a new line between those neighbors. At the

same time, the corresponding record in the list box is deleted. A similar operation happens in drag mode. The user selects a point by clicking on the red circle and moving the mouse. The coordinates will be changed. In order to avoid program confusion, the moving range is limited within the x coordinate of its neighbors. The button "Change" at the upper right corner allows the user to input a value in the textbox above to specify an exact desired position.

Step 7

Once the speed curves are specified, the motors will start rotating after the user clicks the "Go" button in the loop window. The program will first check the current load position and the initial position from which the loop will start. An initializing movement will be made if the values are not equal. Then the program will generate a series of codes including the distance, the acceleration rate and the moving time.

5 Data Acquisition Experiment

In order to testify to the usability of the GUI, a data acquisition experiment followed the design process. The main idea was to utilize the rotary motor to perform 180 degrees scanning of human faces so that different angle views of face profiles could be acquired and used in a face recognition project. Figure 5.1 shows the hardware platform sketch and an actual photograph of this setup is shown in Figure 5.2

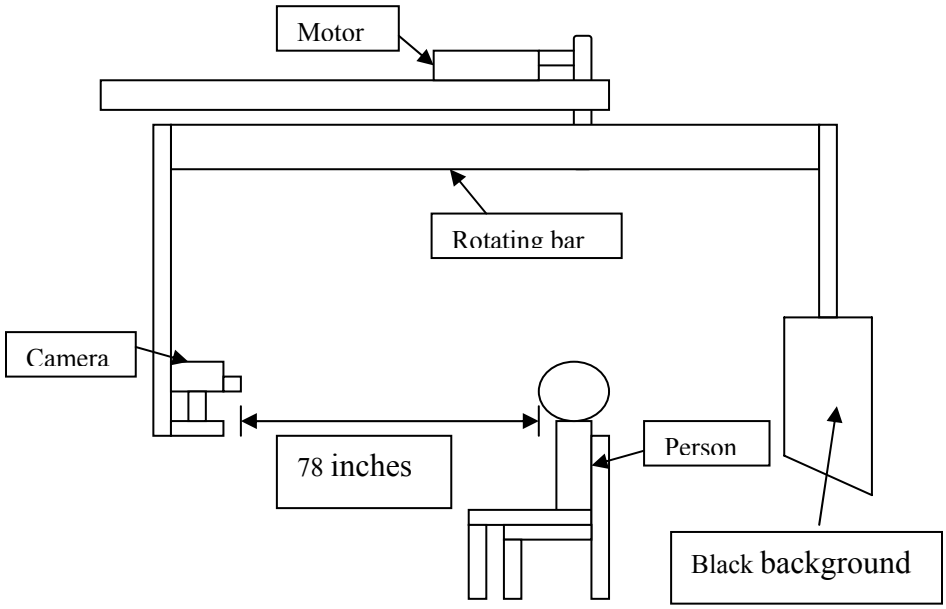


Figure 5.1 Sketch of data acquisition platform

Figure 5.2 Sample picture of data acquisition platform

As can be seen from the Figure 5.2, the whole setup is connected to a metal frame that is made of 80/20 aluminum sticks. In order to maximally reduce moving vibration, heavy-duty materials such as 2”x 8”x 8’ woods and 1”x3’ metal shafts were used. The motor was placed on top of the setup and a timing belt used to convey motion. The Figure 5.3 shows the inside of the setup.

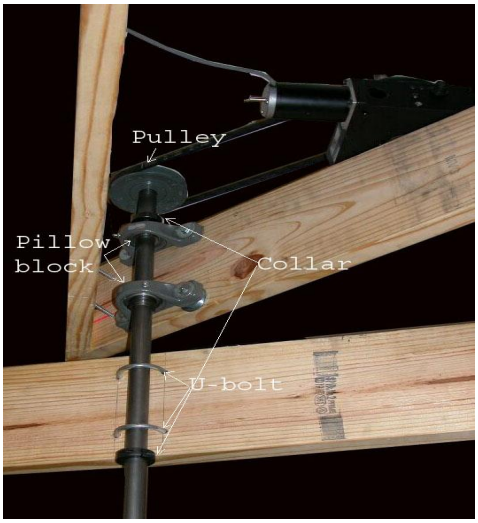


Figure 5.3 Inner structure of platform

During scanning, the motor is controlled in rotation from the host computer. With a person sitting beneath the shaft, the cameras start rotating and real time video data are sent to a video capturing computer through the RCA ports. Two frame grabbers, which are connected to two individual cameras, transfer the video to image frames. Henceforward, after 180-degree scanning (side to side), multiple views of human face data are acquired.

In order to provide a comparatively large image gallery for the face recognition project, we considered changing the illumination conditions and having people show different expressions. Besides the regular room lights, two extra spot lights were placed in the front of the person so that the lights could be turned on and off in sequence enabling us to capture able to get face images under different illumination conditions. A list of all conditions under which we acquired images from includes:

- Different camera modalities
  - Thermal camera
  - Color camera

- Difference illumination conditions
  - Both spot lights on, room light on

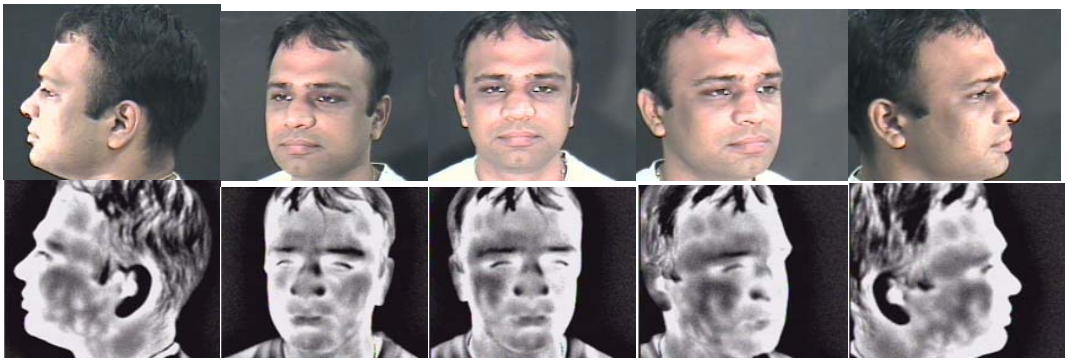
- Left spot light on, right off, room light on
- Left spot light off, right on, room light on
- Both spot lights off, room light on
- All off

- Different face expressions
  - Anxiety/ Astonished
  - Smiling
  - Angry/ Worried

Sample results from this process are shown following: (video and thermal images in the same column indicate+ both are taken at the same moment)

Illumination part:

1.Both lights on



2. Left light on

3. Right light on

4. Both lights off

5. Complete dark

Expression:

1. Anxiety/ Astonished

2. Smiling

3. Angry/ Worried

6. Conclusion

The design of a graphical user interface for a 3D stepper motor system is proposed in this paper. The hardware system architecture and software implementation process is outlined. The complete system consists of four PM type stepper motors, with three mounted on a linear motion table, one mounted on a rotary motion table, four motion controllers, one hardware indexer and one PC card. The hardware specifications are included. All system movements are controlled by a host computer using the GUI described in this paper. Besides simple movement and basic parameter setting functions, the GUI provides windows for easy loop motion control with the preset speed-position curve. A data acquisition experiment followed to testify to the usability of the GUI. The results show the ability for easy control and accurate positioning. A hardware setup was designed for acquisition purposes. However, a vibration problem remains. Stability of the setup affects the quality of the result images.

In the future, the GUI, together with the hardware system, can be used in more applications to fulfill the tasks of data acquisition, robot navigation and camera surveillance, and so on. More GUI functions such as the polynomial speed-positioning curve and more convenient input methods can also be considered as an extension. Additionally, the textbox in main window which lets user type MCL sentences allows the user to realize more functionalities inherited from the manufacturer's control software package.

Reference

[1] "Integrating PC-based logic and motion control" White paper, Entivity Incorporation

[2] Benjamin C, Kuo. "Incremental motion control-step motors and control systems", 1979

[3] Technical paper "Introduction to stepper motor system". Anaheim Automation Corporation

[4] Douglas W. Jones "Control of stepping motor. Tutorial" http://www.cs.uiowa.edu/~jones/step/types.html

[5] http://www.sdp-si.com/D220/PDF/D220T155.pdf

[6] Susan Weinschenk, Pamela Jamar, Sarah C. Yeo. "GUI design essentials". 1997

[7] Mark Minasi. "Secret of effective GUI design", 1994

[8] Tan Kok Kiong, Lee Tong Heng, Dou Huifang and Huang Sunan. "Precision motion control: design and implementation", 2001

[9] Igor Aleksander, Henri Farreny, Malik Ghallab Technology. "Decision and Intelligence (volume 6)", 1987

[10] J. Randolph Andrews. "Motion control terminal blocks: the next step in distributed motion control", Incremental Motion Control Systems and Devices Symposium, 1999

[11] Parker Hannifin Corporation technical paper. "Linear motor basics"

[12] Hayer control system Ltd technical essay. “The advantages of PC-based control systems”. 2000

[13] Lorenzo Sciavicco, Bruno Siciliano. “A survey on robot control technology”, 2001

[14] www.Compumotor.com

[15] www.motorola.com

Appendix ----Comments of Visual Basic Source code

In order to make this GUI more understandable, a short explanation of the algorithm is provided. Focus is on the use of motor control language (referred to as MCL) in Visual Basic and how to realize real time control to the system so that it is easier for other people to understand this program and make changes.

As mentioned earlier, the information exchange between the computer and the controller is achieved by downloading sequences of code sentences in the format of MCL. The function, SendAT6400Block (Device_Address%, cmd$, 0) is called every time when the downloading process takes place. “cmd” is a string that stores the sentences written in MCL. The main function of the GUI is to generate several “cmd” strings under different circumstances. The figure below shows a sample cmd string that is going to be downloaded to chip.

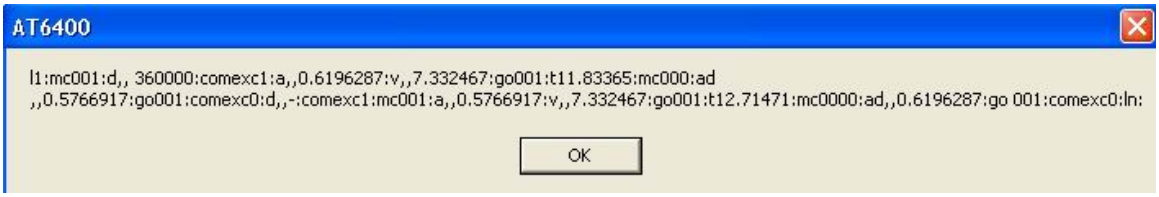


Figure 1 Sample of control code sequences

At design time, the MLC source codes is always output to a message box before downloading for clarity and ease in checking. In the Figure 1, colons are used to separate two sentences. Generally, there are two types of syntax in MCL. The first one is similar to “d 20000, 30000, 40000, 50000”, where “d” is the command to set the moving distance on individual axis. “20000, 30000, 40000, 50000” means the motor will move 20000, 30000, 40000, 50000 steps in axis 1, 2, 3 and 4, respectively. An example of the second syntax would be “go0010”. Each binary bit after “go” stands for the command settings for one axis. In this example “go0010” means to initiate motion in axis 3, while at the mean time, other axises remain still. Another frequently used command is “mc” and its usage is of the second type of syntax. It sets two motion modes of the motor. When “mc” is set to 1, the continuous mode is enabled and all movements in that axis

will be at a specific velocity with distance command establishing the direction. On the other hand, all movements will be for a specific distance in the preset mode (mc0).

One of the areas that contains a number of commands is the loop control window. When the "Go" button in the loop control window is pressed, the program will first check how many "time" text boxes are not zero. At the beginning of each time loop, all axises will go to the initial positions, which are the values in the "From" text box. Those that are not involved in the loop will be commanded to return to their home position. After the number of moving axises is detected, the program will determine if there is specified speed control curve for each axis. If not, the default speed and acceleration rate will be applied. Also if the speed profiles between axises are detected to be identical, movements in those axises will happen at the same time. The whole string is stored in the variable "cmd". When dealing with complex situations, such as desired motion in several axises with different speed requirements for each, the string can be very long. In order to prevent buffer overloads, the string is cut into several parts to download.

All the MCL commands used in this GUI are indicated below for quick reference.

Commands	Description	Usage
!	Immediate command identifier	N/A
@	Global command identifier (applied on all axises)	N/A
A	Represents acceleration rate	A2,2,2,2
AD	Represents deceleration rate	ad2,2,2,2
CMDDIR	Defines directions	cmddir0101
COMEXC	Enables continuous command mode (1 means next command is executed before motion is finished)	comexc1
D	Distance	D2000,2000,2000,2000
DRIVE	Drive enable (1: energize, 0: de-energize)	drive1111
DRES	Drive Resolution	dres 4000,4000,4000,4000
ERRLVL	Error detection level	errlvl0
GO	Initializes motion	go1101
K	Kill motion	N/A
MC	Preset/Continuous mode enable	mc0000
L	Starts a loop	L1
LH	Enables hardware limits (0: disable both, 1: disable cw, 2: disable ccw, 3: enable both)	lh3,3,0,3
LN	Loop ends	N/A
PULSE	Pulse width	pulse 1,1,1,1
RESET	Reset parameters to factory default	N/A
T	Waiting time	T 3

(Note: To download the string, all commands must be followed by a ":", or they will not be executed)