GRADUATION THESIS

The Research of the Control Technology of Mobile Robots Based on Kinect(基于 Kinect 移动机器人控制技术研究)

College	Mechanical Engineering College
Major	Mechatronics Engineering
Class	150607
Student No.	15060715
Name	Egbo Munachi Francis(爱格)
Supervisor	MR. JI LI(纪俐)

Shenyang Aerospace University

December 2018

Declaration

To the best of my knowledge and belief, I certify that this thesis does not:

I. Incorporate any material previously submitted for a degree or diploma in any institution of higher education without acknowledgement.

II. Contain any material previously published or written by other person except where due reference is made in the text.

III. Contain any defamatory material.

I also grant permission to Shenyang Aerospace University to make duplicate copies of my thesis as required.

Signature: …………………….

Date: ……December 21, 2018………

Assignment Paper

College:Mechanical Engineering College Major: Mechatronics

Class:150607 Stu.No.:15060715 Name 爱格

Graduation design topic: The research of the control technology of mobile robots based on Kinect

Graduation design time: 2018-09-20 to 2018-12-21

1.Contents:

1） Refer to information, grasp based construction and function of the Kinect control system.

2）Learn the control technology of the mobile robot.

3）According to the requirements of the mobile robot, development of access control system.

4）Compile and debug system, Kinect control mobile robots to perform tasks.

5）Achieve control of mobile robots to perform tasks Kinect.

6）write a graduate design thesis (twenty thousand words).

2.Requirements:

1）Collect the information of Kinect and understand the ；

2）Learn the control technology of mobile robots, assemble the mobile robot in the lab.

3）Design the project of the thesis ahead.

4） Choose the programmer language of the Kinect. Compile the programmer and test the code to Kinect.

5) Connect the kinect to mobile robot with the wireless net. Run the program

6）Let the mobile do some task followed the nature interface control technology from human body posture.

7）Bind thesis. Prepare for disuse.

3. On the basis of design

1）NI LabVIEW of Electromechanical Engineering.

2）The nature interface control of Kinect.

3）Get the nature interface between robot and Kinect.

4. Schedule

1）The first week: Search the information of the Kinect and mobile robot.

2）The second to forth week: Assemble the hardware of the Kinect and mobile robot.

3）The fifth week to seventh week: Design the control method for the nature interface between Kinect and mobile robots.

4）The eighth week to eleventh week: Download the data to the mobile robot and run it. Try to do some task with the Kinect control.

5）The twelfth week to fourteenth week: Write the graduate design thesis according to requirements.

6）The fifteenth week to sixteenth week: Prepare for the report of graduate design。

5. Reference

1) Andrea Sanna, Fabrizio Lamberti. A Kinect-based natural interface for quadrotor control.Entertainment Computing.2013.

2) YU Zhen-zhong, ZHENG Wei-cou, LIU Xin, HUI Jing. Real-time Local Path Planning for Mobile Robot Based on Kinect. Computer Engineering.2013.

3)Antonio Sgorbissa, Damiano Verda. Structure-based object representation and classification in mobilerobotics through a Microsoft Kinect. Robotics and Autonomous Systems.

4) Zhi-Ren Tsai.Robust Kinect-based guidance and positioning of a multidirectional robot by Log-ab recognition. Expert Systems with Applications.2013.

5) L. Susperregi a, A.Arruti b, E.Jauregi. Fusing multiple image transformations and a thermal sensor with kinect to improve person detection ability. Engineering Applications of Artificial Intelligence.

6) ZHENG Li-guo，LUO Jiang-lin，XU Ge. Implementation on mocap system based on Kinect. Journal of Jilin University (Engineering and Technology Edition).

Signature of advisor________________________ ________yy_____mm_____dd

Signature of student________________________ ________yy_____mm_____dd

Abstract

Using Microsoft robotics developer Studio and parallax Eddie robot platform , a mobile robot car(sbrio 9632) was developed using the Microsoft Kinect as the primary computer vision sensor to identify and respond to gesture commands.The project sponsor,Depush Technology of Wuhan, China has requested a commercially viable educational platform. The end user programs the robot using Microsoft Visual Programming Language to implement code written in C# and LabView 2014.

This design not only can be used to control the movements of the robot, modules and design ideas they contain but can also provide the foundation and pave the way for the higher level of development.

Key Words: Kinect sensory detection; natural interaction ; body posture; motion capture; Da-Ni mobile robot: LabVIEW.

Table of Contents

1 Introduction

The field of robotics is rapidly developing; as new technologies are released into the market, robot design is evolving accordingly. New computer vision technologies are a large part of this. Our goal was to develop a mobile robot that uses data collected from the Microsoft Kinect sensor to identify and respond to gesture and voice commands. The Kinect sensor is an advanced computer vision component with a variety of useful features unavailable with other sensors. It combines a microphone array, infrared sensor, and color sensor to produce an accurate visual and auditory map of the environment. One of the major advantages of the Kinect is the skeletal recognition capability. The location of certain human joints identified by the Kinect are gathered and continuously processed by the sensor. This is what makes it possible for the robot to recognize gesture commands.

The DANI-K Robot platform and Kinect will be integrated using Microsoft Robotics Developer Studio running on a laptop on the robot.This provides a library of open source code that take advantage of sensor data. A secondary objective is to implement functionality that will allow the robot to avoid obstacles. Since the end user will determine the extent of his or her obstacle avoidance algorithm, the project team implemented a simple algorithm that keeps the robot from colliding with obstacles. The robot was designed to fit the role of an educational platform for emerging robotics. This eliminated the need for the group to create a library of gesture commands, as these will be defined by the end user. Below is a map of what a typical robotic system looks like that takes input from a user.

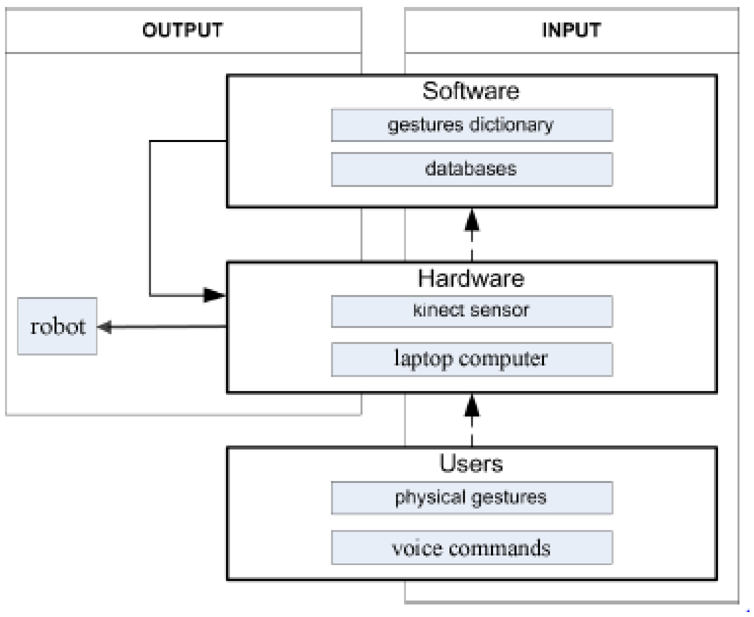


Figure 1-1 Overall Design

The final project would be a robot that would allow the user to write a program in the Microsoft's Visual Programming Language to respond to the voice and gesture commands.

1.1 Mobile Robotics Control And Human-Robotics Interface(HRI)

A brief introduction of human-robot interaction, presenting two general classification, remote and proximate interaction, and detailing the latter, while we detail the social aspect within HRI. Social interaction includes social, emotive, and cognitive facets of interaction, where humans and robots interact as peers or companions, sharing the same workspace and the same goals. Dautenhahn and Billard (1999) propose the following definition to describe the concept of social robot: Social robots are embodied agents that are part of a heterogeneous group: a society of robots or humans. They are able to recognize each other and engage in social interactions, they possess histories (perceive and interpret the world in terms of their own experience), and they explicitly communicate with and learn from each other. According to Fongetal. (2003), the development of such robots requires the use of different techniques to deal with the following aspects: awareness of its interaction counterpart, social learning and imitation, natural language and gesture based interaction. Furthermore, it is worth remembering that HRI research aims to determine friendly social behaviors, thus designing social robots as assistants, peers or companions for humans.

Before there were complex issues regarding human-computer interaction because people needed to think through some of the the hardware input tools such as keyboard, joystick e.t.c … in order to deal with the machine. But the industrial revolution, in the 21st century, human-computer interaction took a large step forward. This trend is more obvious, which somato-sensory interaction with its simple, intuitive , informative and user-friendliness innate superiority to become the new darling of the control mode.

Somato-sensory controller was first revolutionized with the Data Glove input device.Various sensor technologies are used to capture physical data such as bending of the fingers. Often a motion tracker, such as magnetic-tracking device or inertial tracking device, is attached to capture the global position/rotation data of the glove. These movements are then interpreted by the software that accompanies the glove, so any one movement can mean any number of things.Gestures can then be categorized into useful information, such as to recognize sign language or other symbolic functions. Expensive high-end wired gloves can also provide haptic feedback, which is a simulation of the sense of the touch. This allows a wired glove to also be used as an output device. Traditionally, wired gloves have only been available at a huge cost, with the finger bend sensors and the tracking device having to be bought separately. However, the emergence of a new identity is determined by way of an alternative. This approach is shown in the attached appropriate human critical point, then the use of two cameras to capture information and trigonometric functions to calculate the spatial position through these points. These data has a high accuracy for controlling a robot. These somato-sensory sensors are driven largely in the development of somato-sensory measurement technology, and it provides the mathematical algorithms and hardware design, for the development of somato-sensory sensors today, to do the groundwork. With these devices, man-machine communication speed and simplicity was limited, the amount of data that can be transmitted is not large, bt the user experience is poor. To the limitations of these mechanical and electrical equipment, Posture and gestures recognition technology is introduced.

However the traditional somato-sensory system are not found in use for day today robotic design. This is because of its huge in size and also not feasible. But in recent years, as home entertainment controller and sensor devices began to emerge, a new type of human-computer interface design and implementation was introduced. It is the most representative body feeling interactive device, comes from Microsoft’s launch of XBOX consoles of a peripheral. Kinect is Microsoft’s motion sensor add-on for the XBOX-360 gaming console. The device provides a natural user interface(NUI) tat allows user to interact intuitively and without any intermediary device, such as controller.

Figure 1-2 Kinect for Xbox 360

The Kinect system identifies individual players through face recognition and voice recognition. A depth camera, which 'sees' in 3-D, creates a skeleton image of a player and a motion sensor detects their movement. Speech recognition software allows the system to understand spoken commands and gesture recognition enables the tracking of layer movements. Although Kinect was developed for playing games, the technology has been applied to real-world applications as diverse as digital signage, virtual shopping, education.tele-health service delivery and other areas of health IT.

A lot of innovation and change are from the entertainment begun, Microsoft is obviously very aware of this. It is not limited to this advanced gaming technology, but followed the Kinect to the Windows platform, launched the Kinect for Windows SDK, so many developers to design interactive programs based on Kinect, so that it can more play its important applications in many fields, a move that clearly Microsoft profitable way, but also makes the somato-sensory application developers can freely use Kinect sensor in experiments and development.

1.1.1 Kinect Technology

The Microsoft Kinect is a computer vision technology that combines a variety of hardware to capture image, depth and sound. It is also a fairly inexpensive piece of equipment that is readily available to the public. This objective is aimed towards gathering a complete knowledge of what the Kinect is capable of and how this is applicable to this robot design. The goal is to take full advantage of the Kinect and testing was conducted on the actual limitations. The follower service is dependent on the Kinect's field of vision. The group tested the function of the Kinect in a variety of environments. The limitation of the kinect in regards to the lighting in the room was also tested. The group tested with different users with different clothing to ensure that gestures could be recognized regardless of body type.The kinect tracks 20 different joints on the skeleton, so the team tested the gesture recognition service varying the relevant joints to determine how the different combinations affect the effectiveness of the gesture.

1.1.2 Gesture Control

The final prototype is to used as an educational platform; therefore the result of this objective is to produce a service allowing the user to train and recognize gesture and voice commands using the kinect sensor, all through VPL. Microsoft’s open source voice recognition served as a starting point for the starting point for the team’s service, but there is no available Microsoft open source service for gesture recognition, so this service had to completely developed by the team. The services are implemented using Microsoft’s Robotics Developer Studio. Unlike speech recognition, there is not a library for gestures. This requires a program that sets up a library of user inputted gesturesthat the robot can respond to.The gesture trainer service is responsible for storing and recording a new gesture in a library. Finally the gesture recognizer determines which gesture the user performs and matches it to the one stored in the library. It is important that all services are documented properly.

1.1.3 Self Preservation

Obstacle avoidance was originally considered as a secondary objective. The robot purchased from Parallax came equipped with three infrared sensors and two ultrasonic sensors mounted to the front of the chassis. These were utilized effectively for forward obstacle avoidance. After several instances of the robot backing into walls, it was decided that we needed to add five rear facing sensors to the robot to avoid backing into obstacles. There are many obstacle avoidance algorithms available, so the team researched what was available and chose one. It would not have been feasible to write our own algorithm in the time given. The obstacle avoidance service incorporates all ten sensors, ultrasonic and infrared.

1.1.4 Voice and Gesture Control

The final prototype is to be used as an educational platform; therefore the result of this objective is to produce a service allowing the user to train and recognize gesture and voice commands using the Kinect sensor, all through VPL.Microsoft’s open source voice recognition served as a starting point for the team’s service, but there is no available Microsoft open source service for gesture recognition, so this service had to be completely developed by the team. The services are implemented using Microsoft's Robotics Developer Studio. Unlike speech recognition, there is not a library for gestures. This requires a program that sets up a library of user inputted gestures that the robot can respond to. The gesture trainer service is responsible for recording a new gesture. The gesture manager is responsible for storing recorded gestures in a library. Finally, the gesture recognizer determines which gesture the user performs and matches it to one stored in the library. It is important that all services are documented properly.

1.2 Final Project

The final outcome of this project is to integrate these programs, the Microsoft Kinect sensor, a laptop computer, and robot. It should be able to successfully respond to commands and avoid obstacles with a low error rate. The user should have a good personal experience with the robot. This was accomplished by integrating all of the services written and ensuring their proper functionality together.

2 Natural Interactive Motion Capture(Kinect)

Kinect was originally known by the code name "Project Natal". It is a motion-sensing device which was originally developed for the Xbox 360 gaming console. One of the distinguishing factors that makes this device stand out among others in this genre is that it is not a hand-controlled device, but rather detects your body position, motion, and voice. Kinect provides a **Natural User Interface** (**NUI**) for interaction using body motion and gesture as well as spoken commands. Although this concept seems straight out of a fairy tale, it is very much a reality now. The controller that was once the heart of a gaming device finds itself redundant in this Kinect age. You must be wondering where its replacement is. The answer, my friend, is YOU. It's you who is the replacement for the controller, and from now on, you are the controller for your Xbox. Kinect has ushered a new revolution in the gaming world, and it has completely changed the perception of a gaming device. Since its inception it has gone on to shatter several records in the gaming hardware domain. No wonder Kinect holds the Guinness World Record for being the "fastest selling consumer electronics device". One of the key selling points of the Kinect was the idea of "hands-free control", which caught the attention of gamers and tech enthusiasts alike and catapulted the device into instant stardom. This tremendous success has caused the Kinect to shatter all boundaries and venture out as an independent and standalone, gesture-controlled device.

It has now outgrown its Xbox roots and the Kinect sensor is no longer limited to only gaming. Kinect for Windows is a specially designed PC-centric sensor that helps developers to write their own code and develop real-life applications with human gestures and body motions. With the launch of the PC-centric Kinect for Windows devices, interest in motion-sensing software development has scaled a new peak.

Kinect is a horizontal device with depth sensors, color camera, and a set of microphones with everything secured inside a small, flat box. The flat box is attached to a small motor working as the base that enables the device to be tilted in a horizontal direction. The Kinect sensor includes the following key components:

- Color Camera.
- Infrared (IR) Emitter
- IR Depth Senor
- Tilt Motor
- Microphone Array
- LED

Apart from the previously mentioned components, the Kinect device also has a power adapter for external power supply and a USB adapter to connect with a computer. The following figure shows the different components of a Kinect sensor:

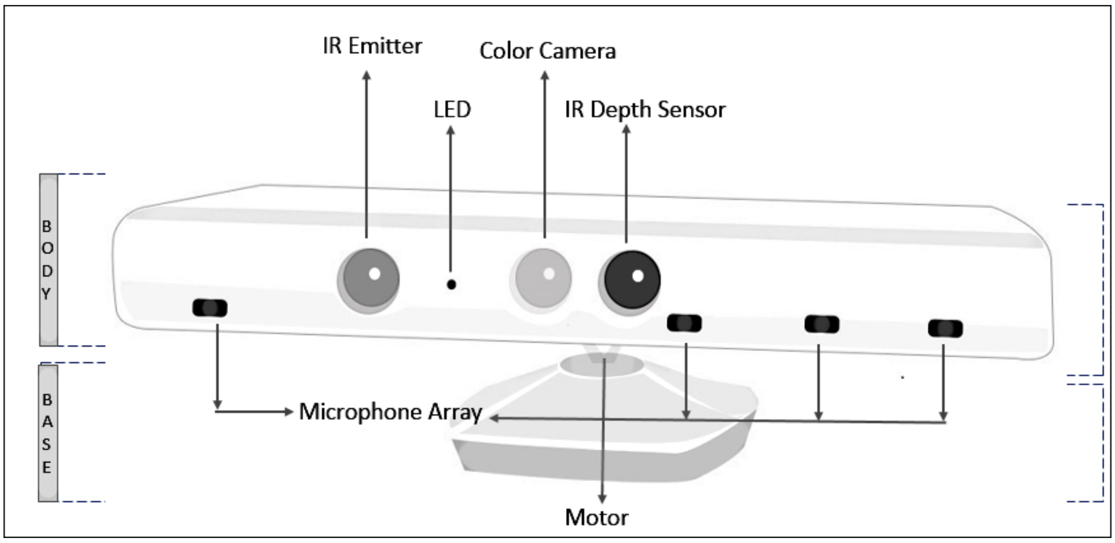


Figure 2-1 Kinect Sensor

From the outside, the Kinect sensor appears to be a plastic case with three cameras visible, but it has very sophisticated components, circuits, and algorithms embedded. If you remove the black plastic cover from the Kinect device, what will you see? The hardware components that make the Kinect sensor work.

The following image shows a front view of a Kinect sensor that's been unwrapped from its black case. Take a look (from left to right) at its IR emitter, color camera, and IR depth sensor:

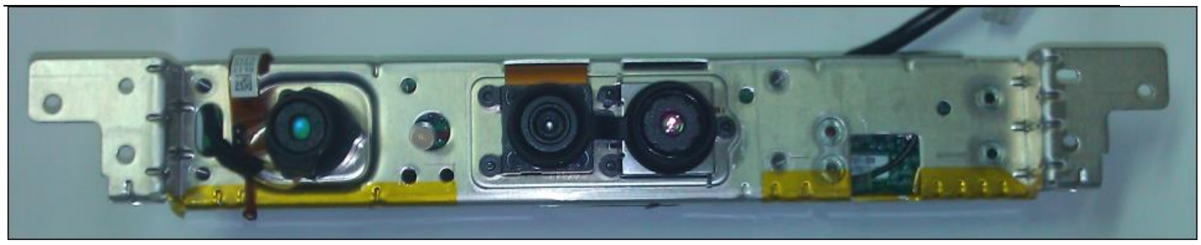


***Figure 2-2** Kinect Camera Sensor*

2.1 Components of Kinect

2.1.1 The Color Camera

This color camera is responsible for capturing and streaming the color video data. Its function is to detect the red, blue, and green colors from the source. The stream of data returned by the camera is a succession of still image frames. The Kinect color stream supports a speed of 30 **frames per second**(**FPS**) at a resolution of 640 x 480 pixels, and a maximum resolution of 1280 x 960 pixels at up to 12 FPS. The value of frames per second can vary depending on the resolution used for the image frame.The viewable range for the Kinect cameras is 43 degrees vertical by 57 degrees horizontal.

2.1.2 IR Emitter and IR Depth Sensor

Kinect depth sensors consist of an IR emitter and an IR depth sensor. Both of them work together to make things happen. The IR emitter may look like a camera from the outside, but it's an IR projector that constantly emits infrared light in a "pseudo-random dot" pattern over everything in front of it. These dots are normally invisible to us, but it is possible to capture their depth information using an IR depth sensor. The dotted light reflects off different objects, and the IR depth sensor reads them from the objects and converts them into depth information by measuring the distance between the sensor and the object from where the IR dot was read. The following figure shows how the overall depth sensing looks:

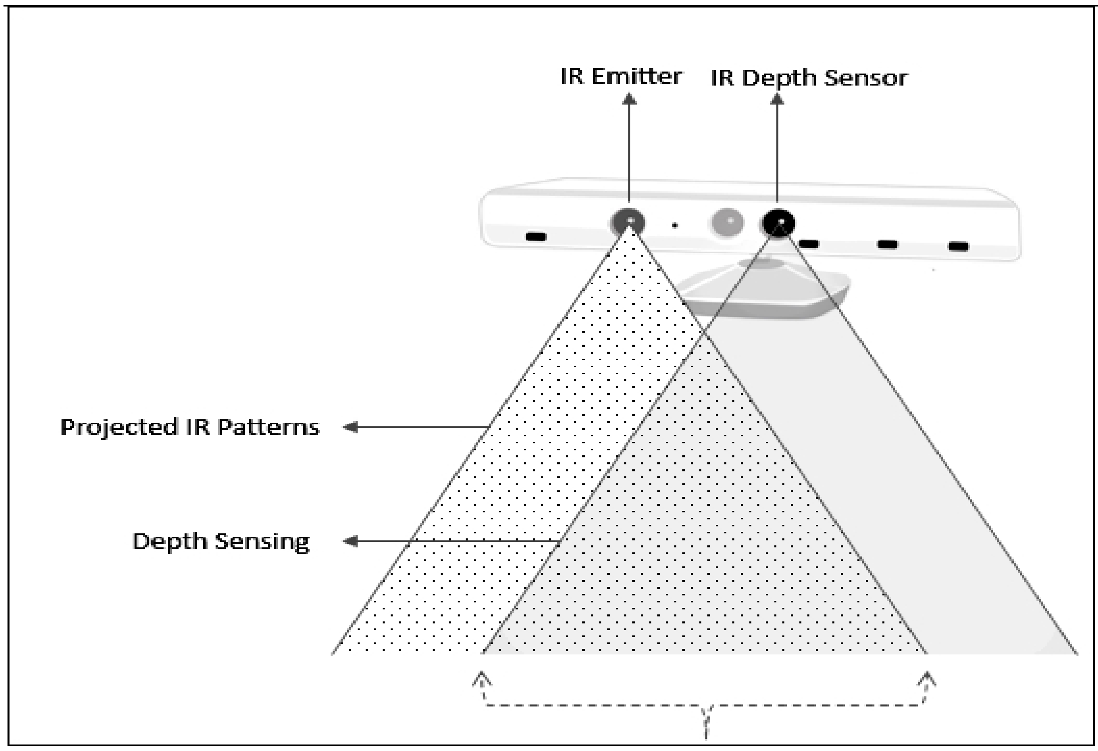


Fig2.3 Kinect Depth Sensing

The Kinect sensor has the ability to capture a raw, 3D view of the objects in front of it, regardless of the lighting conditions of the room. It uses an infrared (IR) emitter and an IR depth sensor that is a monochrome CMOS (Complimentary Metal-Oxide-Semiconductor) sensor.

2.1.3 Tilt Motor

The base and body part of the sensor are connected by a tiny motor. It is used to change the camera and sensor's angles, to get the correct position of the human skeleton within the room. The following image shows the motor along with three gears that enable the sensor to tilt at a specified range of angles,The motor can be tilted vertically up to 27 degrees, which means that the Kinect sensor's angles can be shifted upwards or downwards by 27 degrees.

2.1.4 Microphone Array

The Kinect device exhibits great support for audio with the help of a microphone array. The microphone array consists of four different microphones that are placed in a linear order (three of them are spread on the right side and the other one is placed on the left side, as shown in the following image) at the bottom of the Kinect sensor.The purpose of the microphone array is not just to let the Kinect device capture the sound but to also locate the direction of the audio wave. The main advantages of having an array of microphones over a single microphone are that capturing and recognizing the voice is done more effectively with enhanced noise suppression, echo cancellation, and beam-forming technology.

2.1.5 LED

An LED is placed in between the camera and the IR projector. It is used for indicating the status of the Kinect device. The green color of the LED indicates that the Kinect device drivers have loaded properly. If you are plugging Kinect into a computer, the LED will start with a green light once your system detects the device; however for full functionality of your device, you need to plug the device into an external power source.

Body Posture Detection System of Kinect

The use of the visual Kinect body position detecting method can be achieved by non-contact measurement. The user does not interfere with the purpose of the operation. Re-use Kinect's functions designed as the perfect body posture detection algorithms, has reached the best interactivity. After the design position - instruction table by C# programming for posture and convey instructions to build a bridge.

The depth values produced by the Kinect sensor are sometimes inaccurate because the calibration between the IR projector and the IR camera becomes invalid. This could be caused by heat or vibration during transportation or a drift in the IR laser. To address this problem, together with the Kinect team, I developed a re-calibration technique using the card in Figure 4 that is shipped with the Kinect sensor. If users find that the Kinect is not responding accurately to their actions, they can re-calibrate the Kinect sensor by showing it the card. The idea is an adaptation of my earlier camera calibration technique.2 The depth value produced by the Kinect sensor is assumed to be an affine transformation of the true depth value—that is, Zmeasured ¼ aZtrue þ b—which we found to be a reasonably good model. The goal of recalibration is to determine a and b. (We could also use a more complex distortion model that applies the same technique.) Using the RGB camera, the recalibration technique determines the 3D coordinates of the feature points on the calibration card in the RGB camera's coordinate system, which are considered to be the true values. At the same time, the Kinect sensor also produces the measured 3D coordinates of those feature points in the IR camera's coordinate system. Minimizing the distances between the two point sets, the Kinect sensor can estimate the values of a and b and the rigid transformation between the RGB camera and the IR camera.

2.1.6 Kinect Skeletal Tracking

The innovation behind Kinect hinges on advances in skeletal tracking. The operational envelope demands for commercially viable skeletal tracking are enormous. Simply put, skeletal tracking must ideally work for every person on the planet, in every household, without any calibration. A dauntingly high number of dimensions describe this envelope, such as the distance from the Kinect sensor and the sensor tilt angle. Entire sets of dimensions are necessary to describe unique individuals, including size, shape, hair, clothing, motions, and poses. Household environment dimensions are also necessary for lighting, furniture and other household furnishings, and pets

In skeletal tracking, a human body is represented by a number of joints representing body parts such as head, neck, shoulders, and arms.Each joint is represented by its 3D coordinates. The goal is to determine all the 3D parameters of these joints in real time to allow fluent interactivity and with limited computation resources allocated on the Xbox 360 so as not to impact gaming performance.

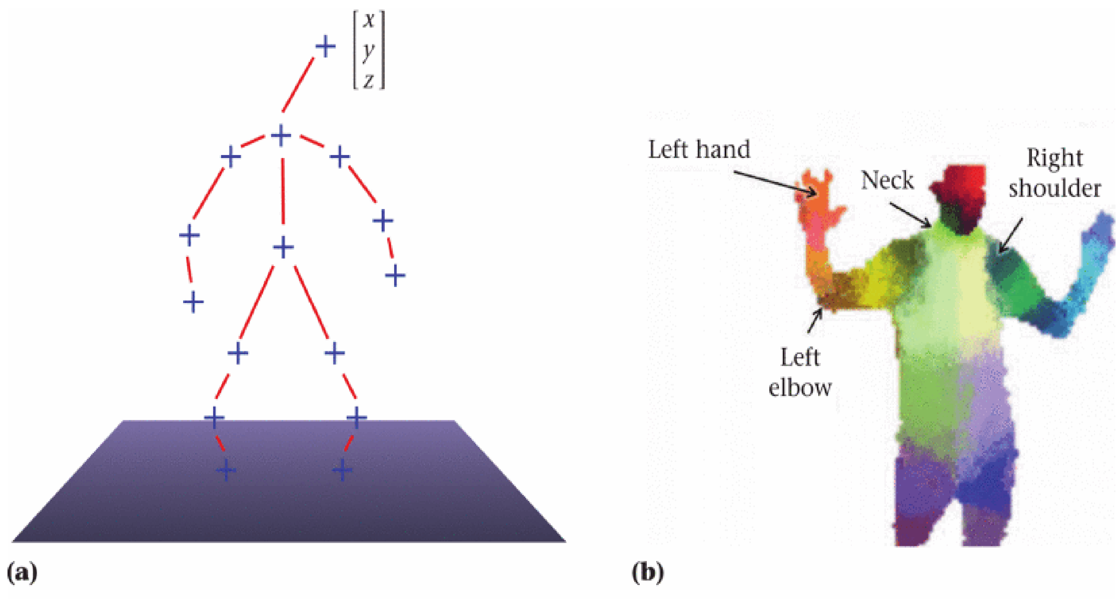


Figure 2-3 Kinect Skeleton Tracking

Player skeleton positions are expressed in x, y, and z coordinates. Unlike the coordinate of depth image space, these three coordinates are expressed in meters. The x, y, and z axes are the body axes of the depth sensor. This is a right-handed coordinate system that places the sensor array at the origin point with the positive z axis extending in the direction in which the sensor array points. The positive y axis extends upward, and the positive x axis extends to the left (with respect to the sensor array), as shown in Figure 5. For discussion purposes, this expression of coordinates is referred to as the skeleton space.

Kinect for Windows can track up to 20 joints in a single skeleton. It can track up to six skeletons, which means it can detect up to six people standing in front of a sensor, but it can return the details of the full skeleton (joint positions) for only one two of the tracking the skeleton of a human body that is seated. The Kinect device can track your joints even if you are seated, but up to 10 joint points only (upper body part).

2.1.7 Floor Determination

Each skeleton frame also contains a floor clip plane vector, which contains the coefficients of an estimated floor plane equation. The skeleton tracking system updates this estimate for each frame and uses it as a clipping plane for removing the background and segmentation of players. The general plane equation is:

$Ax + By + Cz + D = 0$

where:

A = vFloorClipPlane.x
B = vFloorClipPlane.y
C = vFloorClipPlane.z
D = vFloorClipPlane.w

The equation is normalized so that the physical interpretation of D is the height of the camera from the floor, in meters. It is worth noting that the floor might not always be visible. In this case, the floor clip plane is a zero vector. The floor clip plane can be found in the vFloorClipPLanemember of the NUI_SKELETON_FRAME structure in the native interface and the SkeletonFrame.FloorClipPlane field in the managed interface.

2.1.8 Skeleton Mirroring

By default, the skeleton system always mirrors the user who is being tracked. For applications that use an avatar to represent the user, such mirroring could be desirable if the avatar is showing on the screen. However, if the avatar faces the user, mirroring would present the avatar as backwards.

Depending on its requirements, an application can create a transformation matrix to mirror the skeleton and then apply the matrix to the points in the array that contains the skeleton positions for that skeleton. The application is responsible for choosing the proper plane for reflection.

Kinect for Windows SDK functions and Briefings

The Kinect SDK provides both managed and unmanaged libraries. If you are developing an application using either C# or VB.NET, you can directly invoke the .NET Kinect Runtime APIs; and for C++ applications, you have to interact with the Native Kinect Runtime APIs. Both the types of APIs can talk to the Kinect drivers that are installed as a part of SDK installation.

For managed code, the Kinect for Windows SDK provides Dynamic Link Library (DLL) as an assembly (Microsoft.Kinect.dll), which can be added to any application that wants to use the Kinect device.You can find this assembly in the SDK installation director.The Kinect driver can control the camera, depth sensor, audio microphone array, and the motor.Well, as of now we have discussed the components of the Kinect SDK, system requirements, installation of SDK, and setting up of devices. Now it's time to have a quick look at the top-level features of Kinect for Windows SDK. The Kinect SDK provides a library to directly interact with the camera sensors, the microphone array, and the motor. We can even extend an application for gesture recognition using our body motion, and also enable an application with the capability of speech recognition. The following is the list of operations that you can perform with Kinect SDK.

The color camera returns 32-bit RGB images at a resolution ranging from 640 x 480 pixels to 1280 x 960 pixels. The Kinect for Windows sensor supports up to 30 FPS in the case of a 640 x 480 resolution, and 10 FPS for a 1280 x 960 resolution. The SDK also supports retrieving of YUV images with a resolution of 640 x 480 at 15 FPS. Using the SDK, you can capture the live image data stream at different resolutions. While we are referring to color data as an image stream, technically it's like a succession of color image frames sent by the sensor. The SDK is also capable of sending an image frame on demand from the sensor.The Kinect sensor returns 16-bit raw depth data. Each of the pixels within the data represents the distance between the object and the sensor. Kinect SDK APIs support depth data streams at resolutions of 640 x 480, 320 x 240, and 80 x 60 pixels.

C# and WRF Introduction

WPF is a completely new presentation framework, integrating the capabilities of many frameworks that have come before it, including User, GDI, GDI+, and HTML, as well as being heavily influenced by toolkits targeted at the Web, such as Adobe Flash, and popular Windows applications like Microsoft Word.

Graphics, including desktop items like windows, are rendered using Direct3D. This allows the display of more complex graphics and custom themes, at the cost of GDI's wider range of support and uniform control theming. It allows Windows to offload some graphics tasks to the GPU. This reduces the workload on the computer's CPU. GPUs are optimized for parallel pixel computations. This tends to speed up screen refreshes at the cost of decreased compatibility in markets where GPUs are not necessarily as powerful, such as the netbook market.Windows Presentation Foundation (WPF) is a graphical subsystem by Microsoft for rendering user interfaces in Windows-based applications. WPF, previously known as "Avalon", was initially released as part of .NET Framework 3.0 in 2006. WPF uses DirectX and attempts to provide a consistent programming model for building applications. It separates the user interface from business logic, and resembles similar XML-oriented object models, such as those implemented in XUL and SVG.WPF employs XAML, an XML-based language, to define and link various interface elements. WPF applications can be deployed as standalone desktop programs or hosted as an embedded object in a website. WPF aims to unify a number of common user interface elements, such as 2D/3D rendering, fixed and adaptive documents, typography, vector graphics, runtime animation, and pre-rendered media. These elements can then be linked and manipulated based on various events, user interactions, and data bindings.WPF runtime libraries are included with all versions of Microsoft Windows since Windows Vista and Windows Server 2008. Users of Windows XP SP2/SP3 and Windows Server 2003 can optionally install the necessary libraries.

Microsoft Silverlight provided functionality that is mostly a subset of WPF to provide embedded web controls comparable to Adobe Flash. 3D runtime rendering had been supported in Silverlight since Silverlight.

The Windows Presentation Foundation (WPF) is Microsoft's UI framework to create applications with a rich user experience. It is part of the .NET framework 3.0 and higher. WPF's emphasis on vector graphics allows most controls and elements to be scaled without loss in quality or pixelization, thus increasing accessibility. With the exception of Silverlight, Direct3D integration allows for streamlined 3D rendering. In addition, interactive 2D content can be overlaid on 3D surfaces natively.

C# is pronounced “see sharp”. C# is an object-oriented programming language and part of the .NET family from Microsoft. C# is very similar to C++ and Java. C# is developed by Microsoft and works only on the Windows platform. Namespaces are C# program elements designed to help you organize your programs. They also provide assistance in avoiding name clashes between two sets of code. Implementing Namespaces in your own code is a good habit because it is likely to save you from problems later when you want to reuse some of your code.

You specify the Namespaces you want to use in the top of your code. It is easy to add more if you need them. For example, when you create a new Windows Forms application, the following default namespaces will be included.Everything in C# is based on Classes. Classes are declared by using the keyword class followed by the class name and a set of class members surrounded by braces (curly brackets). A class normally consists of Methods, Fields and Properties. Every class has a constructor, which is called automatically any time an instance of a class is created. The purpose of constructors is to initialize class members when an instance of the class is created. Constructors do not have return values and always have the same name as the class.

Display Kinect Data

Because there is no physical interaction between the user and the Kinect sensor, we must be sure that the sensor is set up correctly. The most efficient way to accomplish this is to provide a visual feedback of what the sensor receives. We should not forget to add an option in your applications that lets users see this feedback because will not yet be familiar with the Kinect interface.Even to allow users to monitor the audio, we must provide a visual control of the audio source and the audio level.

All the code you produce will target Windows Presentation Foundation(WPF) 4.0 as the default developing environment. The tools will then use all the drawing features of the framework to concentrate only on Kinect-related code.

2.1.9 The Color Display Manager

Kinect is able to produce a 32-bit RGB color stream. The resolution ranges from 640 x 480 pixels to 1280 x 960 pixels. The Kinect for Windows sensor supports up to 30FPS in the case of a 640 x 480 resolution and 10 FPS for a 1280 x 960 resolution. The SDK also supports retrieving of YUV images with a resolution of 640 x 480 at 15FPS.

3 LABVIEW AND DA-NI ROBOT CONTROL SYSTEM

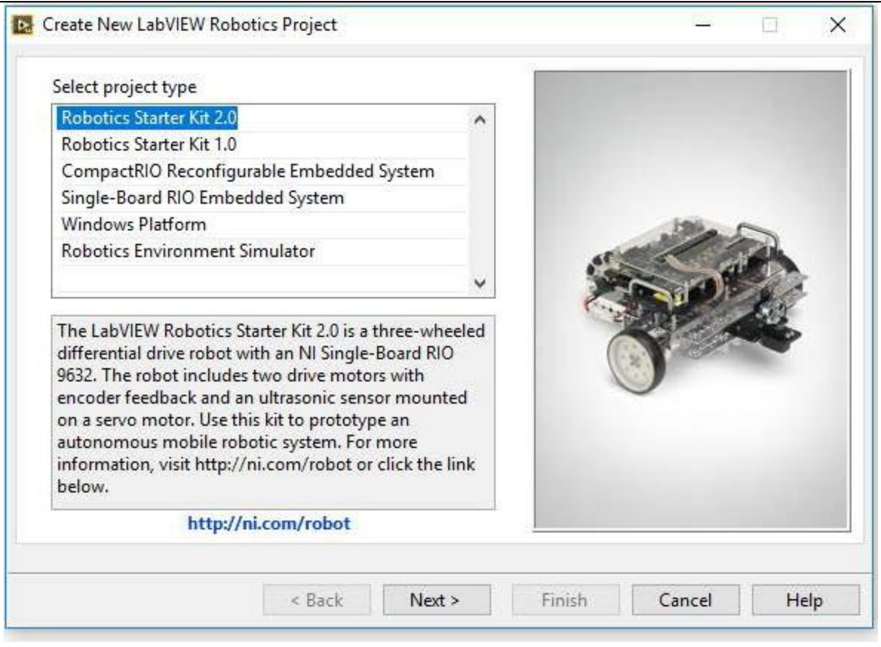
3.1 Introduction to LabView

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a development environment based on the graphical programming language G. LabVIEW is integrated fully for communication with hardware such as GPIB, VXI, PXI, RS-232, RS-485, and plug-in data acquisition boards. LabVIEW also has built-in libraries for using software standards such as TCP/IP Networking and ActiveX.

Using LabVIEW, you can create 32-bit compiled programs that give you the fast execution speeds needed for custom data acquisition, test, and measurement solutions. You also can create stand-alone executable because LabVIEW is a true 32-bit compiler.

You can use LabVIEW with little programming experience. LabVIEW uses terminology, icons, and ideas familiar to technicians, scientists, and engineers, and relies on graphical symbols rather than textual language to describe programming actions.

Programs in LabVIEW for real-world applications can vary from the simple to the powerful, as illustrated in the following graphic.LabVIEW contains comprehensive libraries for data collection, analysis, presentation, and storage. LabVIEW also includes traditional program development tools. You can set breakpoints, animate program execution to see how the program executes, and single-step through the program to make debugging and program development easier. LabVIEW also provides numerous mechanisms for connecting to external code or software through DLLs, shared libraries, ActiveX, and more. In addition, numerous add-on toolkits are available for a variety of application needs.


Create New LabVIEW Robotics Project
Select project type
Robotics Starter Kit 2.0
Robotics Starter Kit 1.0
CompactRIO Reconfigurable Embedded System
Single-Board RIO Embedded System
Windows Platform
Robotics Environment Simulator
http://ni.com/robot
< Back
Next >
Finish
Cancel
Help

The LabVIEW Robotics Starter Kit 2.0 is a three-wheeled differential drive robot with an NI Single-Board RIO 9632. The robot includes two drive motors with encoder feedback and an ultrasonic sensor mounted on a servo motor. Use this kit to prototype an autonomous mobile robotic system. For more information, visit http://ni.com/robot or click the link below.

Figure 3-1 LabView software Interface

3.1.1 Virtual Instruments

- LabVIEW programs are called virtual instruments, or VIs, because their appearance and operation imitate physical instruments, such as oscilloscopes and multimeters.

  -A LabVIEW program has the file ending *.vi, e.g. test.vi

- LabVIEW contains a comprehensive set of tools for acquiring, analyzing, displaying, and storing data, as well as tools to help you troubleshoot code you write.

- In LabVIEW a VI is:

  -A LabVIEW program when it is the top-file.

  -A SubVI when a VI is used in another VI

➢ A SubVI is similar to a function in other programming languages

3.1.2 Data Acquisition

Data acquisition is the process of gathering or generating information in an automated fashion from analog and digital measurement sources such as sensors and devices under test. Data acquisition is supported on Windows and Macintosh only

3.1.3 Data Flow Programming

The programming language used in LabVIEW, also referred to as G, is a dataflow programming language. Execution is determined by the structure of a graphical block diagram (the LV-source code) on which the programmer connects different function-nodes by drawing wires. These wires propagate variables and any node can execute as soon as all its input data become available. Since this might be the case for multiple nodes simultaneously, G is inherently capable of parallel execution. Multi-processing and multithreading hardware is automatically exploited by the built-in scheduler, which multiplexes multiple OS threads over the nodes ready for execution.

Dataflow Programming is a programming paradigm whose execution model can be represented by a directed graph, representing the flow of data between nodes, similarly to a dataflow diagram. Considering this comparison, each node is an executable block that has data inputs, performs transformations over it and then forwards it to the next block. A dataflow application is then a composition of processing blocks, with one or more initial source blocks and one or more ending blocks, linked by a directed edge.

3.1.4 Graphically Programming

The graphical approach also allows non-programmers to build programs simply by dragging and dropping virtual representations of lab equipment with which they are already familiar. The LabVIEW programming environment, with the included examples and the documentation, makes it simple to create small applications. This is a benefit on one side, but there is also a certain danger of underestimating the expertise needed for good quality "G" programming. For complex algorithms or large-scale code, it is important that the programmer possess an extensive knowledge of the special LabVIEW syntax and the topology of its memory management. The most advanced LabVIEW development systems offer the possibility of building stand-alone applications. Furthermore, it is possible to create distributed applications, which communicate by a client/server scheme, and are therefore easier to implement due to the inherently parallel nature of G-code.

3.1.5 DA-NI ROBOTICS

Robotics is built on fundamentals like transducer characterization, motor control, data acquisition, mechanics of drive trains, network communication, computer vision, pattern recognition, kinematics, path planning, and others that are also fundamental to other fields, manufacturing, for instance. The National Instruments (NI) LabVIEW Robotics Kit and LabVIEW provide an active-learning supplement to traditional robotics textbooks and curriculum by providing multiple capabilities in a compact and expandable kit.

The NI LabVIEW Robotics Starter Kit is an out-of-the-box mobile robot platform that features sensors, motors, and NI Single-Board RIO hardware for embedded control. The LabVIEW Robotics software included with the platform includes features for beginners and for those who are more experienced. If you are new to LabVIEW, you can use the high-level LabVIEW Robotics Starter Kit API to quickly get started and control the robot in real time. If you are a more advanced user, you can access the FPGA and perform lower-level customization. The simplicity of this starter kit makes it ideal for teaching robotics and mechatronics concepts or for developing a robot prototype with LabVIEW Robotics. The LabVIEW Robotics Starter Kit includes a prebuilt program that executes a vector fieldhistogram (VFH) obstacle avoidance algorithm based on feedback from the included ultrasonic sensors. With the LabVIEW Robotics Module, you can easily change the behavior of the robot by developing your own algorithms from the ground up or by using algorithms built into LabVIEW Robotics software such as A* path planning.

The NI LabVIEW robotics kit includes DaNI: an assembled robot with frame, wheels, drive train, motors, transducers, computer, and wiring. The hardware can be studied, reverse engineered, and modified. DaNI 2.0, the hardware portion of the LabVIEW Robotics Starter Kit that is used in these experiments, is an out-of-the-box mobile robot platform with sensors, motors, and an NI 9632 Single-Board Reconfigurable I/O (sbRIO) computer mounted on top of a Pitsco TETRIX erector robot base as shown below

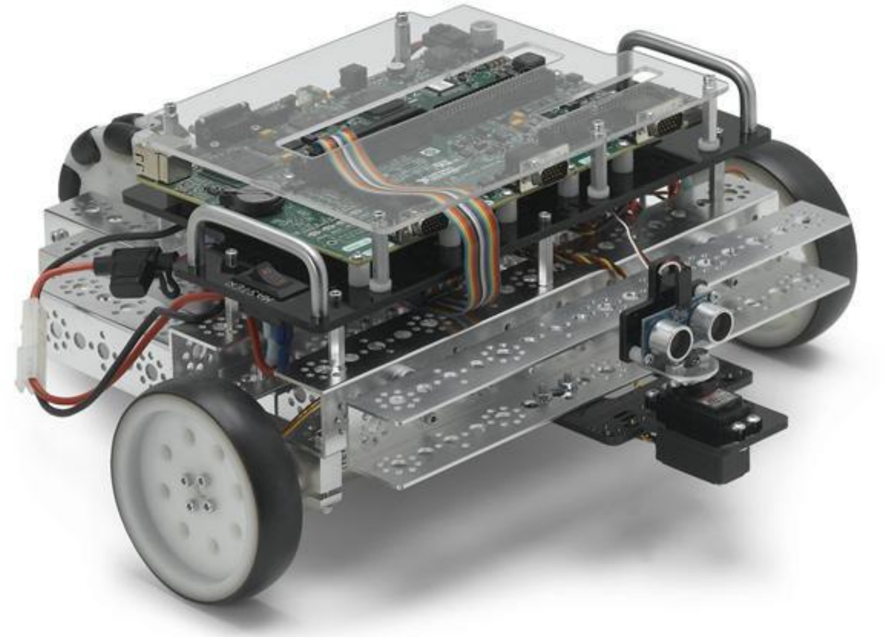


Figure 3-2 DA-NI Robot

3.1.6 DA-NI Hardware Components

Figure 3.2.2 represents a block diagram of the DaNI and Kinect hardware experimental setup (DaNI-K). A miniature PC (fit-PC215) runs LabVIEW 2011, LabVIEW FPGA Module, LabVIEW Real-Time Module, LabVIEW Robotics Module, and the Kinect SDK on Windows 7. The PC is connected to DaNI via an Ethernet cable (RJ-45). Also, the PC is connected to the Kinect sensor via a USB cable. DaNI consists of an NI Single-Board Re-configurable I/O (sbRIO-9632) embedded controller, a TETRIX robotic hardware platform with a dual DC motor controller, two DC motors with encoders, and a PING))) ultrasonic sensor mounted on an R/C type servo motor. The sbRIO is comprised of a 2M gate Xilinx Spartan FPGA, 400 MHz.

Freescale real-time processor, 128 MB DRAM, 256 MB nonvolatile storage, RS232 serial port for peripheral devices, 110 3.3 V (5 V tolerant/TTL compatible) digital I/O lines, 32 singleended/16 differential 16-bit analog input channels at 250 kS/s, four 16-bit analog output channels at 100 kS/s, and a 10/100BASE-T Ethernet port. The sbRIO can be programmed in a combination of programming languages like LabVIEW, LabVIEW MathScript, VHDL, and ANSI C. The TETRIX Building System consists of aluminum parts, DC motors, gears, and wheels for building robotic hardware platforms.

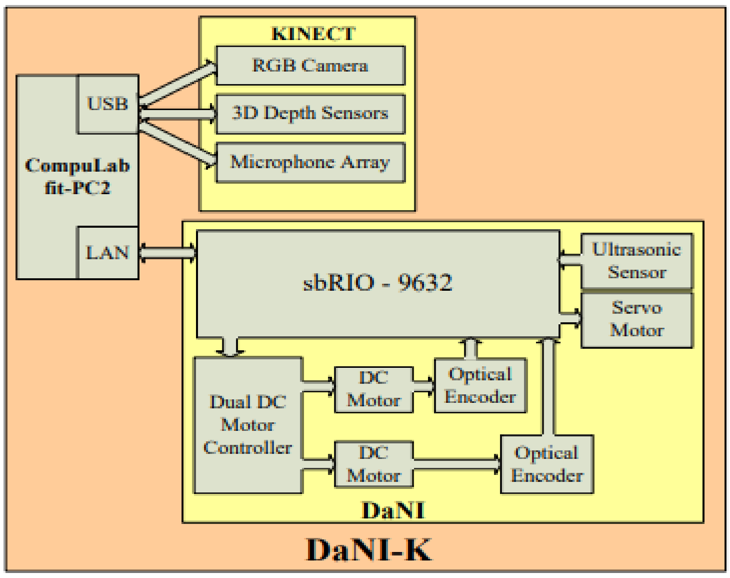


Figure 3-3 Block Diagram of DA-NI

3.1.7 Features of DA-NI

- Integrated real-time controller, reconfigurable FPGA, and I/O on a single board.
- 2M gate Xilinx Spartan FPGA.
- 400 MHz Free scale real-time processor.
- 128 MB DRAM, 256 MB nonvolatile storage.
- RS232 serial port for peripheral devices.
- 110 3.3 V (5 V tolerant/TTL compatible) digital I/O lines.
- 32 single-ended/16 differential 16-bit analog input channels at 250 kS/s.
- Four 16-bit analog output channels at 100 kS/s.
- 10/100BASE-T Ethernet port.
- Low power consumption with single 19 to 30 VDC power supply input.

3.2 D.C MOTOR CONTROL

DC motor can convert Direct Current into Mechanical energy. Because of the good speed performance, it is widely used in Electric Drives. DC motor by exciting way into the permanent, he excited and self-motivation 3 categories, which are divided into self-motivation shunt, series-wound and compound excitation.

Da-NI mobile robot uses a method of changing the voltage of the governor for speed regulation. The mobile robot wheels drive DC motor rotor is mounted on the encoder, the encoder (Encoder) is the signal (such as bit-stream), or data compiled, converted to be used to communicate signals in the form of transmission and storage of equipment. The angular displacement encoder or linear displacement into electrical signals, the former is called a code wheel, the latter known yardstick. The encoder according to readout mode can be divided into contact and non-contact two; according to the working principle can be divided into incremental encoder and absolute categories. Incremental encoder is for the displacement of periodic electrical signal,then changes the electrical signals into counting pulses,represents the magnitude of displacement by the number of pulses. Absolute encoders determine each position corresponds to a digital code, so it shows just the start and end values of the measured position, but not to the middle of the process measurement.

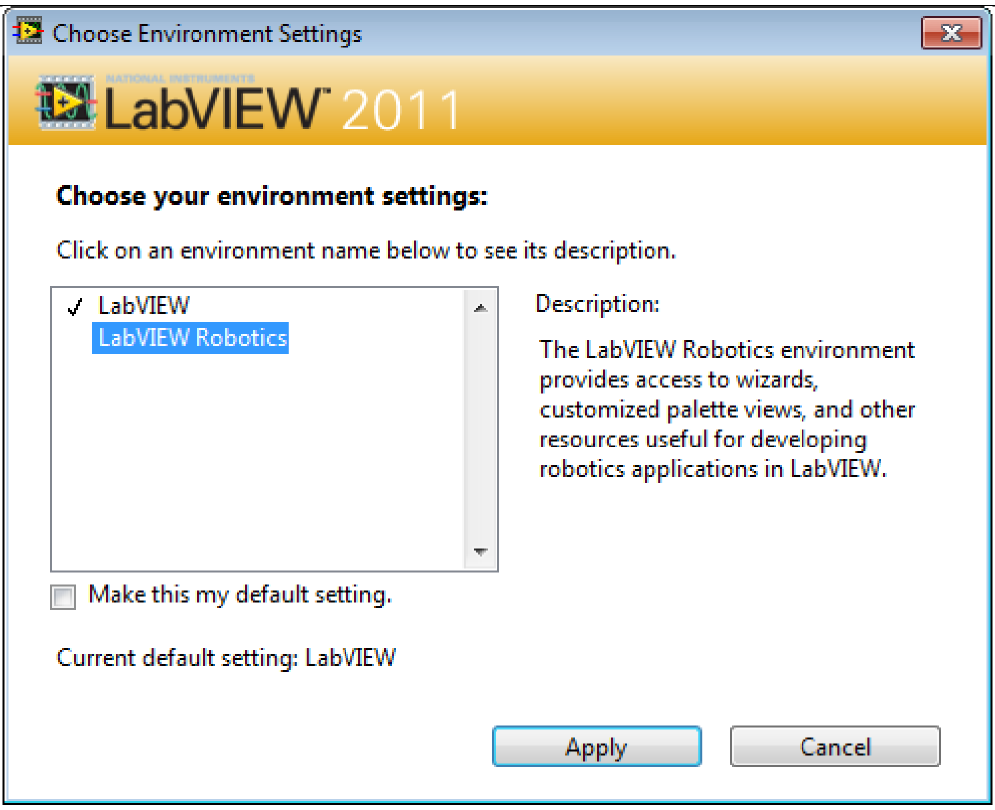
Encoder can be Da-NI have two angular velocity information, angular displacement and angular information. And these are applied to the corresponding module, motor and some control.

3.3 Drive Wheel And Support Wheel

The drive wheels on both sides of the robot can be used as the source of Da-NI power, drive the car forward after about movement, and support wheels mainly play a supportive role as a stable triangle sex. Although the support wheels do not have too many demands, but considering the reliability of friction and movement, still need support wheels make certain requirements that support wheel in X, Y plane (horizontal) movement unfettered as possible, that is, It requires universal support wheels can rotate. So, Da-NI designed a mobile robot were adapted from a caster wheels on the basis of the ordinary.White main wheels may be moving into rolling friction in the Y-axis, while the black deputy wheel can make it in the X-axis moving into rolling friction, so that the freedom of movement in the XY plane, friction is minimal.

3.4 Creating a Project without the Hardware Wizard

The Hardware Wizard automatically loads the necessary software for DaNI. If you didn’t use the wizard, you can add, remove, or update software from MAX. If software has been loaded previously, you can view it by expanding the Software item in the MAX tree as shown in Figure 0-24. If it hasn’t, right click the sbRIO item in the MAX tree and choose add software, or click on the sbRIO item and choose the Add/Remove Software button in MAX. This opens the Real Time Software Wizard window and you can choose which software components to download from the host to DaNI. All items were selected for the configuration shown in Figure below



Choose your environment settings:

Click on an environment name below to see its description.

✓ LabVIEW
LabVIEW Robotics

Description:

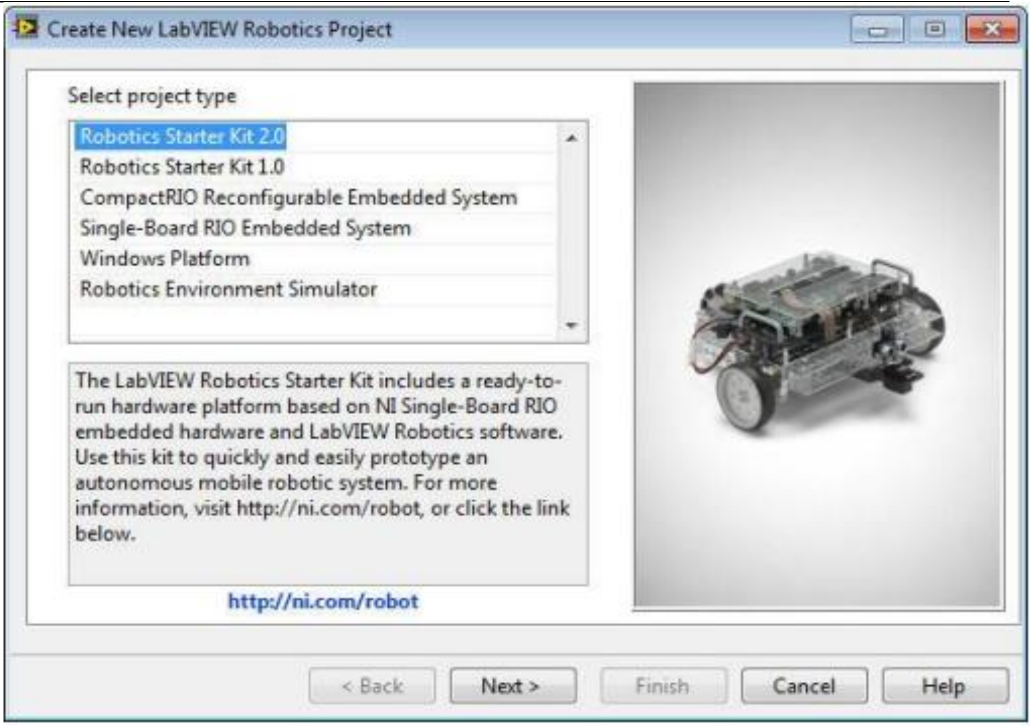
The LabVIEW Robotics environment provides access to wizards, customized palette views, and other resources useful for developing robotics applications in LabVIEW.

☐ Make this my default setting.

Current default setting: LabVIEW

Figure 3-4 LabView Robotics

Click the Start LabVIEW button at the bottom of the window and the Getting Started window shown in Figure 0-26 opens. Select Create New Robotics Project. Select the DaNI 2.0 project from the select project type window as shown in Figure 0-27 and click Next. If the host is connected to DaNI with the crossover cable and DaNI is powered on, the project wizard should inherit the IP address correctly from MAX as shown in Figure 0-28 and you can click Next. If not, enter the IP address and click Next. Enter a location on the host computer hard drive where you have write access as shown in Figure 0-29, enter a project name, and click Finish.


Create New LabVIEW Robotics Project
Select project type
Robotics Starter Kit 2.0
Robotics Starter Kit 1.0
CompactRIO Reconfigurable Embedded System
Single-Board RIO Embedded System
Windows Platform
Robotics Environment Simulator
http://ni.com/robot
< Back
Next >
Finish
Cancel
Help

The LabVIEW Robotics Starter Kit includes a ready-to-run hardware platform based on NI Single-Board RIO embedded hardware and LabVIEW Robotics software. Use this kit to quickly and easily prototype an autonomous mobile robotic system. For more information, visit http://ni.com/robot, or click the link below.

Figure 3-5 Creating A LabView Project

3.5 DA-NI Test

If you used the Hardware Wizard, it will automatically build a project like the one shown in Figure 0-30 for you. If you didn't use the Hardware Wizard, you can use the Project Wizard (explained above) to build the project. To open a project that was closed when you exited LabVIEW, launch LabVIEW and click the project, as shown in Figure 0-31, if it is listed in the getting started window. If it isn't listed, use the Browse button to locate and open it.The project shown in the figure includes the host computer, the DaNI sbRIO target, sensor and actuator drivers, and software programs. Once you become more familiar with LabVIEW you will be able to develop projects without the wizard, and consequently will have more control over what is included in a project. National Instruments uses the LabVIEW Project Explorer to facilitate communication between a PC and a remote target (the sbRIO on DaNI). The Project Explorer window includes two pages, the Items page and the Files page. The Items page shown in the figure displays the project items as they exist in the project tree. The Files page displays the project items that have a corresponding file on disk. The Project Explorer window includes the following items by default: Project root—Contains all the items for the project and displays the file name. My Computer—Represents the local or host computer as a target in the project. Dependencies—Includes items that software programs (VIs) under a target require. Build Specifications—Includes build configurations for source distributions and other types of builds available in LabVIEW toolkits and modules. You can use Build Specifications to configure stand-alone applications, shared libraries, installers, and zip files.

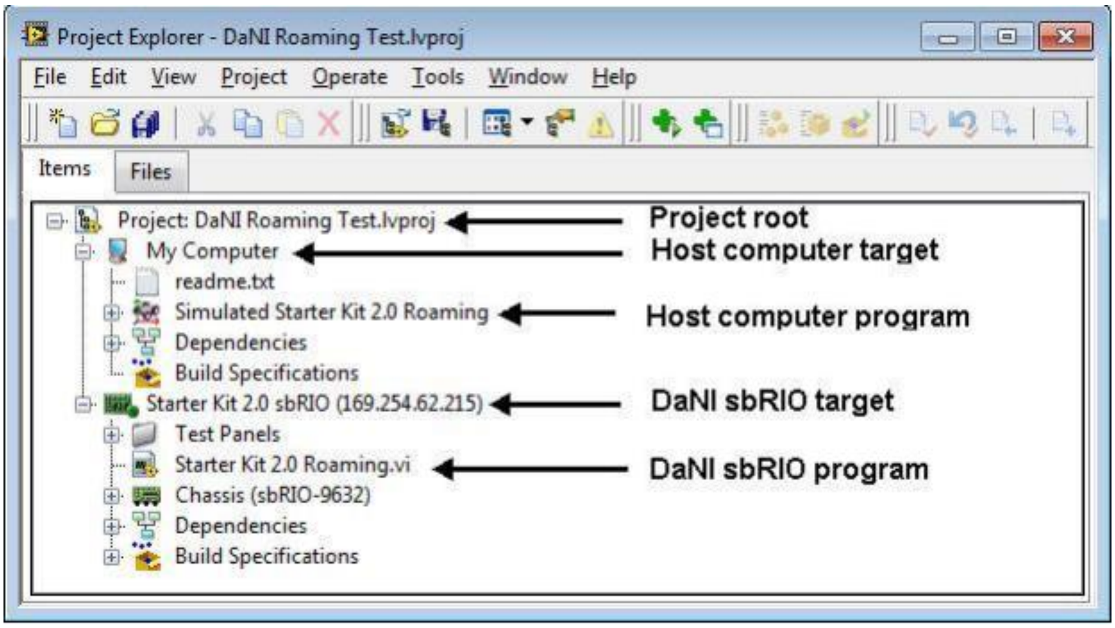
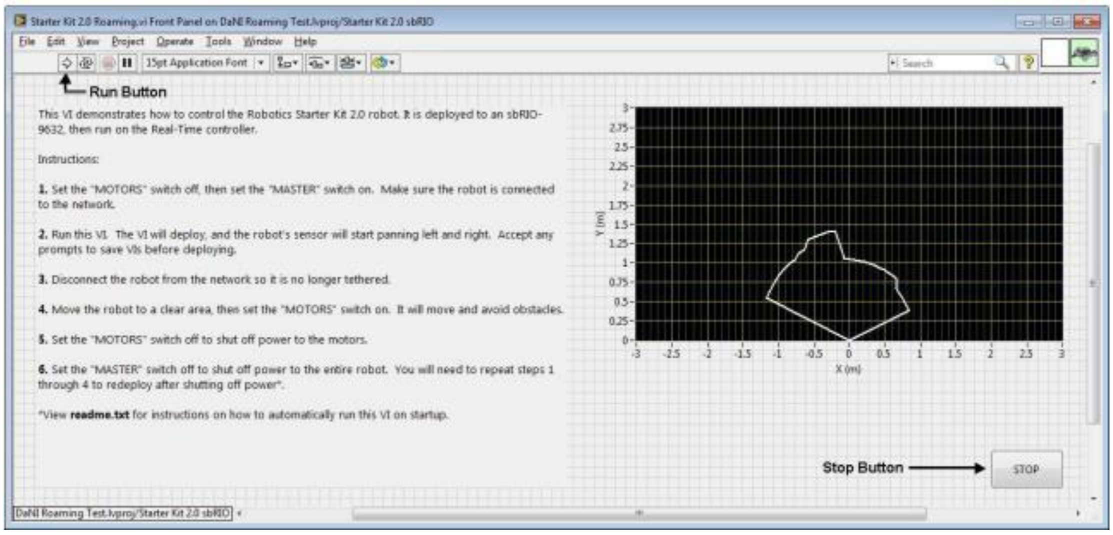

Project Explorer - DaNI Roaming Test.lvproj
File Edit View Project Operate Tools Window Help
Items
Files
Project: DaNI Roaming Test.lvproj
My Computer
readme.txt
Simulated Starter Kit 2.0 Roaming
Dependencies
Build Specifications
Starter Kit 2.0 sbRIO (169.254.62.215)
Test Panels
Starter Kit 2.0 Roaming.vi
Chassis (sbRIO-9632)
Dependencies
Build Specifications
Project root
Host computer target
Host computer program
DaNI sbRIO target
DaNI sbRIO program

Figure 3-6 LabView Project Explorer

The items in the project are arranged in a tree or hierarchical structure. The first item, —Project:...— is the root. This item shows the name of the file saved on disk with the file extension lvproj. The second item, My Computer, is indented to show it is lower in the hierarchy. It represents the host computer where programs are developed. The third and fourth items, Dependencies and Build Specifications, are indented below My Computer indicating that they are lower in the hierarchy and belong to My Computer. The next item moves up in the heirarchy so its level is equivalent to My Computer. It represents another computer in the project, the sbRIO on DaNI. In addition tothe name of the computer, the IP address is displayed. The sbRIO item has dependencies and build specification items like My Computer and some additional items. The Chassis item is part of the sbRIO that connects to and communicates with transducers and actuators. The NI Robotics Starter Kit Utilities.lvlib item are some utility programs that have been written for DaNI to speed the development of programs by allowing users to focus on high-level robotics concepts. To add a software program to a LabVIEW Project, right-click the hardware target which the program should run on, and select New » VI. If the program is placed under My Computer, it will execute on the host. If the program is placed under the sbRIO target, it will deploy to and execute on the sbRIO. When you complete the Robotics Project Wizard, LabVIEW opens the Roaming.vi software program on the host computer as shown in Figure next page.


Starter Kit 2.0 Roaming.vi Front Panel on DaNI Roaming Test.lvproj/Starter Kit 2.0 sbRIO
File Edit View Project Operate Tools Window Help
15pt Application Font
Run Button
Y [m]
X (m)
Stop Button
STOP
DaNI Roaming Test.lvproj/Starter Kit 2.0 sbRIO

This VI demonstrates how to control the Robotics Starter Kit 2.0 robot. It is deployed to an sbRIO-9632, then run on the Real-Time controller.

Instructions:

1. Set the "MOTORS" switch off, then set the "MASTER" switch on. Make sure the robot is connected to the network.

2. Run this VI. The VI will deploy, and the robot's sensor will start panning left and right. Accept any prompts to save VIs before deploying.

3. Disconnect the robot from the network so it is no longer tethered.

4. Move the robot to a clear area, then set the "MOTORS" switch on. It will move and avoid obstacles.

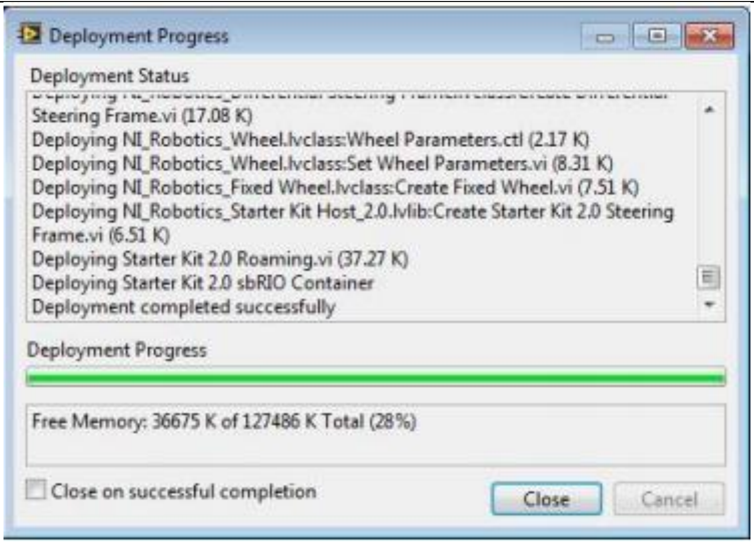
5. Set the "MOTORS" switch off to shut off power to the motors.

6. Set the "MASTER" switch off to shut off power to the entire robot. You will need to repeat steps 1 through 4 to redeploy after shutting off power*.

*View **readme.txt** for instructions on how to automatically run this VI on startup.

Figure 3-7 Roaming program user interface

This program was written for you in LabVIEW. LabVIEW programs are called virtual instruments, or VIs, because their appearance and operation imitate physical instruments, such as oscilloscopes and multimeters. A VI has two windows, a front panel window and a block diagram window. The front panel shown in above is the user interface for the VI. The block diagram will be discussed later. The front panel has a graph of distance to obstacles and a stop button. The distance to obstacles graph is from the PING))) ultrasonic transducer that pans +/- 65 while DaNI drives. The point direction angle is relative to the direction of travel, so the graph orientation or reference is as if you were riding on DaNI. Before clicking the Run button, review the LabVIEW Robotics Starter Kit Safety Guide by navigating to the LabVIEW\readme directory on the DVD that is packaged with the kit and opening StarterKit_Safety_Guide.pdf. Remember that DaNI is expensive so be very careful not to damage it. When you click the Run button, the software that was automatically developed by the Wizards will be deployed to the DaNI sbRIO and the Deployment Progress Window shown in below will be displayed.


Deployment Progress

Deployment Status

Steering Frame.vi (17.08 K)
Deploying NI_Robotics_Wheel.lvclass:Wheel Parameters.ctl (2.17 K)
Deploying NI_Robotics_Wheel.lvclass:Set Wheel Parameters.vi (8.31 K)
Deploying NI_Robotics_Fixed Wheel.lvclass:Create Fixed Wheel.vi (7.51 K)
Deploying NI_Robotics_Starter Kit Host_2.0.lvlib:Create Starter Kit 2.0 Steering Frame.vi (6.51 K)
Deploying Starter Kit 2.0 Roaming.vi (37.27 K)
Deploying Starter Kit 2.0 sbRIO Container
Deployment completed successfully

Deployment Progress

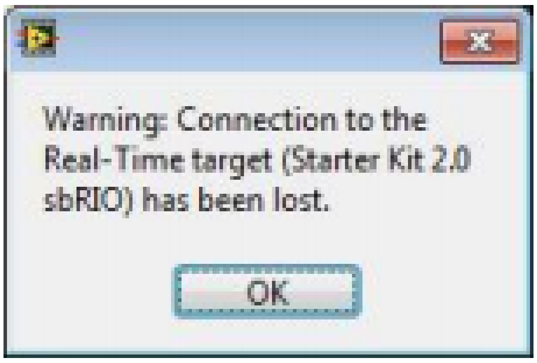
Free Memory: 36675 K of 127486 K Total (28%)

Close on successful completion

Close Cancel

Figure 3-8 Deployment progress window

When you disconnect the network cable to allow DaNI to roam untethered, the program running on the host computer will display a series of messages like the one shown in Figure 0-35. When you click the OK button, the application on the host will terminate. That is okay since DaNI no longer needs this program. Whether the program on the host runs or not doesn’t affect the operation of DaNI after the network cable has been disconnected


Warning: Connection to the Real-Time target (Starter Kit 2.0 sbRIO) has been lost.

OK

Figure 3-9 Lost connection message

If the software deploys and DaNI functions with and without the network tether, it has successfully tested and you can proceed to the next section of the experiment. Save the project in a folder on your computer where you have write access and you can open it in the future without using the robotics project wizard.

3.6 LabView and Wifi Connection

3.6.1 Configuring the DA-NI chasis network settings.

➢ Plug one end of your Ethernet cable into the NI CompactDAQ chassis and the other end into your computer’s Ethernet port.

➢ Open Measurement & Automation Explorer (MAX).

- Expand Devices and Interfaces to confirm that your device is detected. If the chassis is not listed under Devices and Interfaces » Network Devices, right-click the chassis and select Add Device. If the device is not listed, right-click Network Devices, and select Find Network NI-DAQmx Devices.
- Click on the DA-NI chassis and navigate to the Network Settings tab on the bottom of the MAX window
- On the Configure IPv4 Address pull-down tab, select Static. Change the IPv4 Address to 192.168.127.xxx where xxx is a number less than 255 and not 253 (253 is the default IP Address of the Moxa AWK-3121 wireless device. Change the subnet mask to 255.255.255.0. Click the save option at the top of the MAX window. This configures your NI CompactDAQ chassis to communicate with the Moxa AWK-3121 wireless device.

However, you will not be able to access the device from your computer until after configuring the PC network settings.

3.6.2 Configure the router network settings.

- Remove the Ethernet cable from your DA-NI chassis and plug it into the LAN port of your wireless router. This setup uses a TP-Link wireless router, but any wireless router with the capability to maintain a static IP address should be acceptable.
- In your browser, enter the Default IP Address number. This should be listed on the bottom of your wireless router. If not, search "<Manufacturer's Name> Default IP Address" on Google or any other search engine, where <Manufacturer's Name> is the name of the router manufacturer. For TP-Link, this number is 192.168.0.1 and the login is Admin with no password.
- Log in to the router using the default user name and password.
- In the Router IP Address box, enter a new IP Address. For the Sb-9628 device, valid IP Addresses will look like 192.168.127.xxx where xxx is a number less than 255 and not 253 or the value assigned to the DA-NI chassis. Change the subnet mask to 255.255.255.0. An image of how this should look when using a TP-Link wireless router.

3.6.3 Configure the PC.

At this point, all of your devices are configured so that they can communicate with each other. The only thing left to do is to configure your computer's IP Address so that it can close the communication circuit.

- Navigate to your Control Panel. Click on Network and Sharing Center. In the Network and Sharing Center, select Change Adapter Settings option.
- Right-click the Local Area Connection network and select Properties

- Double-click on Internet Protocol Version 4 (TCP/IPv4) option and its property box should pop up.
- Select Use the following IP Address and change the IP Address to 192.168.127.xxx where xxx is a number below 255 but not a number previously used or 253 and click OK.
- Click OK on the Local Area Connection Properties pop-up to enable the new IP Address.

4 Kinect Based Mobile Robotics Control System Design

4.1 Overall Framework

The system should take the attitude detection and command transmission as the core, to complete the main objective programming. In order to use we add color video data and information presentation and other auxiliary user-friendly commissioning functions. The method of the control of robot to be more natural and vivid as possible, we use gesture and command control method. This will work in benefit of somatosensory control method. To appoint basic motion control of the robot we can take people's daily habits and simple and intuitive and so on in to account. The basic controls of the mobile robot would be; moving forward, moving backward, turn left, turn right and stop moving. To accomplish the full range of function of the mobile robot on the ground we use these simple postures listed below in table 4.1. Human hand gestures will be used with the use of the right hand.

Table 1 *Postural Robot Instructions Accordance with Human Hand Gestures*

Postural	Right	Right	Right	Right	Right
	hand raised	hand down	hand to the		hand on the

Robot	Moving	Moving	Turn	Turn	Stop
Instruction	Forward	backwards	left	right	movin

4.1.1 Three-Dimensional Coordinate Algorithm of Skeletal Points

The Kinect sensor consists of an infrared laser emitter, an infrared camera and an RGB camera. The inventors describe the measurement of depth as a triangulation process. The laser source emits a single beam which is split into multiple beams by a diffraction grating to create a constant pattern of speckles projected onto the scene. This pattern is captured by the infrared camera and is correlated against a reference pattern. The reference pattern is obtained by capturing a plane at a known distance from the sensor, and is stored in the memory of the sensor. When a speckle is projected on an object whose distance to the sensor is smaller or larger than that of the reference plane the position of the speckle in the infrared image will be shifted in the direction of the baseline between the laser projector and the perspective center of the infrared camera.

These shifts are measured for all speckles by a simple image correlation procedure, which yields a disparity image.

To get Kinect detected skeletal information, polling mode constantly gives the latest skeletal point information. This method is simple, accurate, and the event model is when a particular event occurs, go get the skeletal point information in certain circumstances recall skeletal point information. This method saves system resources, and totally efficient. However, this procedure requires real-time updates on the latest bones information, so the use of polling method. In this case, we can define the human skeleton information ready to complete the event, when the Kinect ready intact skeleton information, event triggers. Then the program will receive information about the new human skeleton type of variable load, so that new bone was to obtain information to complete the update skeletal information.

The three-dimensional coordinates of the points of the human body is not capable of direct access, since the Kinect detects Position property to 20 skeletal points in the X, Y coordinates of -1 to +1. This is only displayed in the screen position relative to the screen center it coordinates, so we can directly be used to determine the body posture has some limitations, and Z-axis coordinate is the distance from the point to the plane of the camera (unit: m). It is possible to use it to calculate the real value of X, Y coordinates, and use it to detect the body posture.

As I mentioned before, the viewable range of Kinect camera is 43 degrees vertical by 57 degrees horizontal. When the Kinect camera has been set as the origin, the establishment of the spatial Cartesian coordinate system is required. Using the trigonometric knowledge the X, Y and Z coordinates can be determined.

Wherein, X, Y, Z coordinates of the bones are returned directly to Kinect, and realX, realY, realZ skeletal point of real-world coordinates in meters (m). The figure below shows the establishment of Cartesian coordinate system.

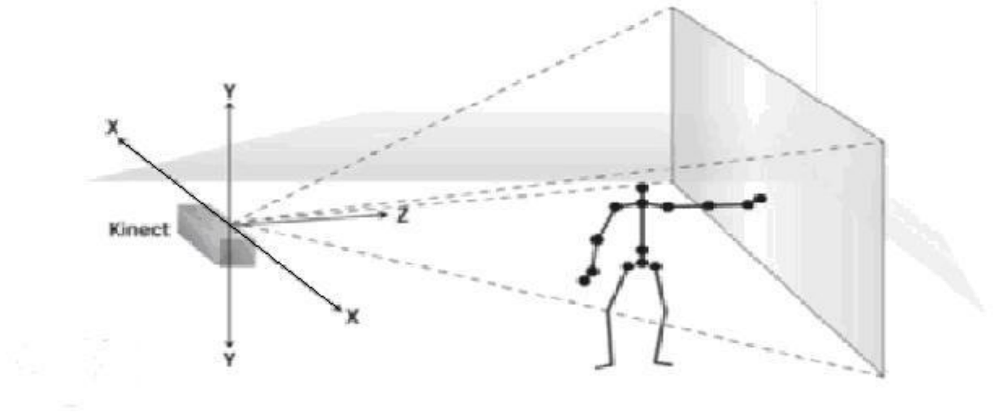


Figure 4-1 Three dimensional plot of Kinect

Using the above formula you can get a few real three-dimensional skeletal point coordinates. In C#, classes can be used very easily to achieve this function. Create Realposition class that in the right realX, realY, realZ three properties which are used to store three real coordinate values of the above three formulas and the use of which, you can get these three values.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Kinect;

namespace kinectPPTControl
{
    class RealPosition
    {
        public double realX;
        public double realY;
        public double realZ;
        public RealPosition(Joint a)
        {
            this.realZ = a.Position.Z;
            this.realX = System.Math.Tan(Math.PI * 28.5 / 180) * a.Position.Z * 
a.Position.X;
            this.realY = System.Math.Tan(Math.PI * 21.5 / 180) * a.Position.Z * 
a.Position.Y;
        }

    }
 }
```

So that only the definition of a RealPosition class object (for example RealPositionrealPositionHead;), And its input variables corresponding Joint type (egrealPositionHead = newRealPosition(head);), then the real three-dimensional coordinate values realX， realY,realZ three properties to attach its corresponding.

4.1.2 Human Height Detection Algorithm

We must identify user's height used in the program. In order to achieve this we define a global variable height (double type) to store the value of height. The calculation to get the height is relatively easy. Simply subtract actual Y axis coordinates foot

(`realPositionHead.realY`) with the head point actual Y axis coordinates

((`realPositionRightFoot.realY`)) to get the user height. According to the implementation body as follows: Height define global variables (publicdoubleheight;) in Mainwindow. When the skeletal point real three-dimensional coordinate conversion is

complete, add the following code:

```
    height = realPositionHead.realZ – realPositionRightFoot.realZ;
```

So that we can successfully get the user's height, due after every bones get information, real human skeleton point coordinate value will be recalculated, the body height is no exception, so the height data is constantly updated. When the course the user changes the posture detection system is still able to make judgments based on the new user's height. After a certain number of experiments we were able to get the following results.

Table 2 *Height Measurement*

Real Height(m)	1.81	1.75	1.65	1.73	1.77
Measured Height(m)	1.84	1.79	1.69	1.74	1.77
Error(%)	1.66	2.29	2.42	0.58	0.00

There is a slight error in our results, but the error is less than 3%. Therefore it can be said to meet the actual demand. So the algorithm we used is reasonable. The algorithm can be used to measure it is before Kinect users using height, thus providing the body height parameters for the program after the content.

4.1.3 Arm Pose Threshold

The purpose of features is to ease the fast and accurate detection of the hand region in a depth image. According to the body structure of the data, in order control the Da-NI mobile robot more naturally and reliably we need to set a zero-reference point, after testing and research. The human shoulder intermediate point (shoulderCenter) 0.1 right heights (height), to 0.1 heights at the point set reference point, and set the following thresholds.

Table 3 Human Hand Posture Criteria

Postural	Right	Right	Right	Right	Right
	hand to the	hand to the	hand up	hand down	hand to
	Right	left			chest
Criteria	Right X	Right X	Y	Y	None
	Coordinate	coordinate	coordinate of	coordinate of	of the
	greater than	greater than	the origin of	the origin of	four cases
	or origin 0.2	or origin 0.2	the right hand	the right hand	and set
	Height	Height	is greater	less than 0.2	
			than 0.2	Height	
			Height		

According threshold setting, in order to reflect more directly, the value of different regions, drawing diagrams of the various regions of the judgment results are shown in below.

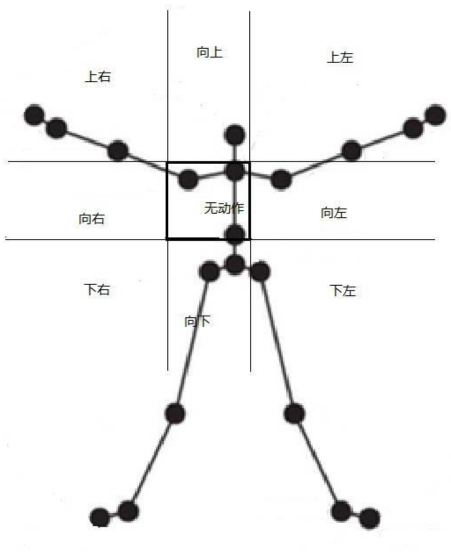


Figure 4-2 Posture in different regions (Center Line Determination)

4.1.4 Attitude Detection of Programming

First the program should obtain a color image and skeletal information. Color image helps the user as a reference to the functions. The skeletal information is the key to determine the body posture. And as in for support the program some additional tips and modules are needed to be added accordingly.

When the program is running, firstly it obtains a color image while waiting for the skeletal information to be detected. The information is ready when the bones (skeletal) make a posture. Then the development of text-based prompt box, and then call the attitude detection function detects the posture in turn determine when there is a corresponding position is made, and if not, will not issue commands; and if so, to issue the corresponding instructions Ni procedures, after completion of this step, return to the "Waiting skeleton information is detected," this step on. And continue to cycle until the end of the program.

While we install all the hardware devices we also need to install all the required software. Firstly we install Kinect Driver to our computer, which is Kinect for Windows SDK. I installed Kinect for Windows SDK version 1.7. And at the beginning of the preparation process, in order to successfully accomplish these functions described above, the first to do some preparation work such as the production of the Kinect sensor interface and initialized. To make the interface of the program we need to use the Windows Presentation Foundation 4.0. In this we create a WPF Project, and then add an Image form image sensor acquired for display. Kinect is initialized after this step, since to use Kinect hardware, we should call it the SDK API, mentioned above, it’s an API for developers experiment is free, so we can easily obtain the API and call them After install SDK, we add a reference to Microsoft Kinect. After making reference to the namespace (MicrosoftKinect), through our program we can directly call the Kinect for Windows SDK API.

For Kinect Sensor to define an object, a KinectSensor class should be introduced. The class provides for managing the data provided, do not switch and gets a lot of information. MainWindow statement follows two variables, which KinectSensor object represents a separate Kinect device, while byte array is used to store image data.

```
KinectSensorkinectSensor;
    private byte [] pixelData;
```

When the form is loaded no doubt want to Kinect device is initialized, it is added to the Kinect device initialization code in the handler Loaded time, as follows:

```
private void Window_Loaded(object sender, RoutedEventArgs e)
       {
           kinectSensor = (from sensor in KinectSensor.KinectSensors where
sensor.Status == KinectStatus.Connected select sensor).FirstOrDefault();

kinectSensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30)
;
           kinectSensor.Start();
          kinectSensor.ColorFrameReady += kinectSensor_ColorFrameReady
       kinectSensor.SkeletonStream.Enable();
           kinectSensor.SkeletonFrameReady+=new
EventHandler<SkeletonFrameReadyEventArgs>(kinectSensor_SkeletonFrameReady);

}
```

Since Kinect SDK supports multiple sensors simultaneously, we use KinectSensor as an array. LINQ statement using the first or default KinectSensor extracted, can be operational. After the collar it can acquire the image information and bones information, you can start running.

Followed by a color image acquisition, adding the code in kinectSensor_ColorFrameReady event to obtain a color image, as follows:

```
       private void kinectSensor_ColorFrameReady(object sender,
ColorImageFrameReadyEventArgs e)
       {
           using (ColorImageFrame imageFrame = e.OpenColorImageFrame())
           {
               if (imageFrame != null)
               {
                   this.pixelData = new byte[imageFrame.PixelDataLength];
                   imageFrame.CopyPixelDataTo(this.pixelData);
                   this.ColorImage.Source =
BitmapSource.Create(imageFrame.Width, imageFrame.Height, 96, 96,
PixelFormats.Bgr32, null, pixelData, imageFrame.Width *
imageFrame.BytesPerPixel
               }
           }
       }
       private void kinectSensor_SkeletonFrameReady(object sender,
SkeletonFrameReadyEventArgs e)
       {
           using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())
           {
               if (skeletonFrame != null)
               {
                   skeletonData = new
Skeleton[kinectSensor.SkeletonStream.FrameSkeletonArrayLength];
                   skeletonFrame.CopySkeletonDataTo(this.skeletonData);
                   Skeleton skeleton = (from s in skeletonData where
s.TrackingState == SkeletonTrackingState.Tracked select
s).FirstOrDefault();
                   if (skeleton != null)
                   {
                       SkeletonCanvas.Visibility = Visibility.Visible;
                       ProcessGesture(skeleton);
                       textBox1.Text = " Get Skeleton Data!!
                   }
               }
           }
       }
```

When a frame color data stream preparation is completed, it will trigger ColorFrameReady event, this handler will pass a ColorImageFrameReadyEventArgs parameter e, () method to obtain the next frame of data returned by the Kinect device OpenColorImageFrame. If you get this data, call CopyPixelDataTo () method to copy the data to the byte array, and the size of the array by ColorImageFrame of PixelDataLength OK. Finally, a BitmapSource, assign its value to ColorImage Source property, even if its source into the acquired image data. It can be displayed in a color image in Image1 Kinect returned, and thanks to the polling method, a color image obtained will be constantly updated, so you get the video stream data. Data traffic will occur because of the two ways, polling mode and event mode. In the polling mode it gets the latest skeletal point information constantly. This method is simple and accurate. In event model it will go get the skeletal point information only when a particular event occurs. We use this in specific cases, which saves system resources and totally efficient. In this we use the polling mode, in order to facilitate continued access to the latest data.Skeletal data stream and color data stream are similarly obtained when the corresponding sensor data processors the function to pass the appropriate parameters. And then it calls the method to get the data, copy it to a predetermined corresponding variable of the type. After the re-use them as needed, then it is no longer Ao said similar parts, different parts of the specific code as follows:

```
        RealPositionrealPositionHead;


        RealPositionHead=new RealPosition (head);
```

First, define a variable RealPosition type realPositionHead, give RealPosition() method to pass the corresponding original sketal point information, namely joint type of head data. This allows the realPositionHead of realX, realY,realZ the three properties obtained in the corresponding coordinate the information.In order to get the bone position when the data stream after that skeleton is not empty, then the posture detection method passes the ProcessGesture() to the current skeletal information. This alternative method is to detect the body posture at the same time, in order to show that the information has been acquired bone, set "Get Skeleton Data !!!" in textBox1 the Text property, the string is displayed in textBox1.Next is the position detection method ProcessGesture(), you must first obtain the original skeleton data. Joint definition of a data type is used to store the original skeleton information, in the head. for example, the code is:

```
    Joint head = (from j in s.Joints where j.JointType == JointType.Head
    select j)


    FirstOrDefault ();
```

Use LINQ statement attribute selection JointType Head of skeletal point (first or default which one), and assigns it to the variable head, head of the Unit Head variables there are points of the skeleton information. The same method can be used a few remaining original skeletal bones information point, and is not Ao said.

In the skeletal point information into a real three-dimensional coordinates of the location. This calls RealPosition classes. Code is as follows:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Kinect;

namespace kinectPPTControl
{
    class RealPosition
    {
        public double realX;
        public double realY;
        public double realZ;
        public RealPosition(Joint a)
        {
            this.realZ = a.Position.Z;
            this.realX = System.Math.Tan(Math.PI * 28.5 / 180) *
a.Position.Z *     a.Position.X;
            this.realY = System.Math.Tan(Math.PI * 21.5 / 180) *
a.Position.Z * a.Position.Y;
        }

    }
}
```

RealPosition class definitions, containing realX, realY, realZ three properties represent the three-dimensional real skeleton point coordinate information. At the same time the definition of public RealPosition (Joint a) function, the function of the right to information on a skeleton of the original information is processed to obtain a true three-dimensional skeleton information, and assign it to realX, realY, realZ three properties. So it is very easy call.

In the head, for example, get real coordinate information code is as follows:

```
RealPositionrealPositionHead;

realPositionHead = new RealPosition (head);
```

Define a variable RealPosition type realPositionHead, give RealPosition() method to pass the corresponding original skeletal point information, namely Joint type of head data. This allows the realPositionHead of realX, realY, realZ three properties obtained in the corresponding coordinate the information.

To calculate the height after height define global variables to hold the height Found (double type) after getting real-world coordinates, the Y coordinate Y coordinate minus the head of his right foot, come to value and assign height.

```
    height = realPositionHead.realZ - realPositionRightFoot.realZ;
```

This step is to set a reference point centerPosition upon completion. CenterPosition define a class, as follows:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace kinectPPTControl
{
    class CenterPosition
    {
        public double X;
        public double Y;

    }
}
```

This type has two properties, X and Y, the two coordinate values are stored for. Then we have to define a variable of type CenterPosition, assignment of its X, Y property. Code is as follows:

```
    centerPosition.X = realPositionCenterShoulder.realX + 0.1 * height;

    centerPosition.Y = realPositionCenterShoulder.realY - 0.1 * height;
```

Thus, the mid-point on the right shoulder height 0.1, 0.1 below the height of the point has been set to the reference point. It can provide a reference for judging whether or exceeds the threshold. After the above preparatory work is completed, it is necessary to detect the human body posture, right here in order to detect whether the left, for example, explain the testing process. First of all, we at the beginning of the program defines four Boolean, used to label the current machine which instruction is executed, the code is as follows:

```
        publicboolisBackGestureActive = false;

        publicboolisForwardGestureActive = false;

        publicboolisLeftGestureActive = false;

        publicboolisRightGestureActive = false;
```

The four Boolean indicates respectively whether backward, whether forward, whether left, right if four Boolean values. Detecting whether the right to the left and send a corresponding signal code as follows:

```
private void ProcessGesture(Skeleton s)
        {
            Joint rightHand = (from j in s.Joints where j.JointType ==
JointType.HandRight select j).FirstOrDefault();
            Joint head = (from j in s.Joints where j.JointType ==
JointType.Head select j).FirstOrDefault();
            Joint centerShoulder = (from j in s.Joints where j.JointType ==
JointType.ShoulderCenter select j).FirstOrDefault();
            Joint rightfoot = (from j in s.Joints where j.JointType ==
JointType.FootRight select j).FirstOrDefault();
            RealPosition realPositionRightHand;
            RealPosition realPositionHead;
            RealPosition realPositionRightFoot;
            RealPosition realPositionCenterShoulder;
            realPositionRightHand = new RealPosition(rightHand);
            realPositionHead = new RealPosition(head);
            realPositionRightFoot = new RealPosition(rightfoot);
            realPositionCenterShoulder = new RealPosition(centerShoulder);
            height = realPositionHead.realY - realPositionRightFoot.realY;
            centerPosition = new CenterPosition();
            centerPosition.X = realPositionCenterShoulder.realX + 0.1 *
height;
            centerPosition.Y = realPositionCenterShoulder.realY - 0.1 *
height;
```

```
            textBox2.Text = Convert.ToString(height);
            if (realPositionRightHand.realX < centerPosition.X - height *
0.2)
            {
                if (isLeftGestureActive)
                { }
                else
                {
                    mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0);
                    mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0);
                    System.Windows.Forms.SendKeys.SendWait("{Backspace}");
                    System.Windows.Forms.SendKeys.SendWait("{Backspace}");
                    System.Windows.Forms.SendKeys.SendWait("{Backspace}");
                    System.Windows.Forms.SendKeys.SendWait("{Backspace}");
                    System.Windows.Forms.SendKeys.SendWait("3");
                    System.Windows.Forms.SendKeys.SendWait("{Enter}");
                    System.Windows.Forms.SendKeys.SendWait("{Enter}");
                    System.Windows.Forms.SendKeys.SendWait("{Enter}");
                    System.Windows.Forms.SendKeys.SendWait("{Enter}");
                    isForwardGestureActive = false;
                    isBackGestureActive = false;
                    isLeftGestureActive = true;
                    isRightGestureActive = false;
                }
                }
```

First of all we judge the real right hand position, which the true value is less than the X coordinate, X coordinate of the reference point of minus 0.2 times the height. If the result is false then there would be no action, and continues to detect the next position. If it is true, it is determined whether the current instruction is the left, i.e. isLeftGestureActiveis true, the current instruction has been left, then the operation is not done to prevent repeatedly send the same instruction, if the current instruction is not left, NI car left the issue of instructions, and then update the four Boolean, if left to set to true, the rest of the set to false. This can be done on the left position to judge. The remaining position determination also is similar, this is no longer Ao said. This will complete the posture of judgment and send the appropriate instructions to the NI program.

Movement Control Scheme

4.1.5 The Design of the DaNI Mobile Robot

As we already know the car has two wheels and a caster. The robot can move freely along the rear axle (defined as X-axis) is free to move forward. The vice wheel can make it along the left and right axle (defined as Y-axis) to move freely. The integrated use of the main wheel and pinion, you can make the car rear wheel movement in the XY plane are into rolling friction, that almost no resistance to the free movement in the XY plane, while Caster is its main supporting role, and two drive wheels so as to form three-point Support, since the triangle with the stability that can guarantee the car will not fall. The mobile car control, we must work hard on two wheels. As the two drive wheels grip ideally performance, so take advantage of the two drive wheels to rotate at a speed corresponding to the front left and right movement to control the car.

4.1.6 Overall Framework of the Movement Control Program

The both sides of the drive wheels rotate with the same speed to achieve straight forward and backward movement. And the difference between the rotational speeds of the two-wheels allows to adjust the sub direction. Therefore, when a LabVIEW program receives the attitude test signal sent, it is different according to the received signal, DA-NI mobile robot two wheels different speeds of data written to follow the appropriate rotational speed, and by encoder error correction errors, achieve closed-loop control. In this framework, the car will be able to move in accordance with our requirements on the ground.

4.2 LabView and Starter Kit 2.0 with NI sbRIO-9632

Once we enter the four rounds of input and the output oscilloscope, we can see the waveform. This process is continuous cycle, constantly updated based on the obtained value of the car rounds command speed input, you can control the car in a different manner. After the program is written, firstly using the initial module allows the car to boot. After the establishment of the process structure, as shown below, the corresponding data will transfer and will be initialized. Then the two-wheeled speed variable stored Left and Right input speed value and the motor car are set to 0.

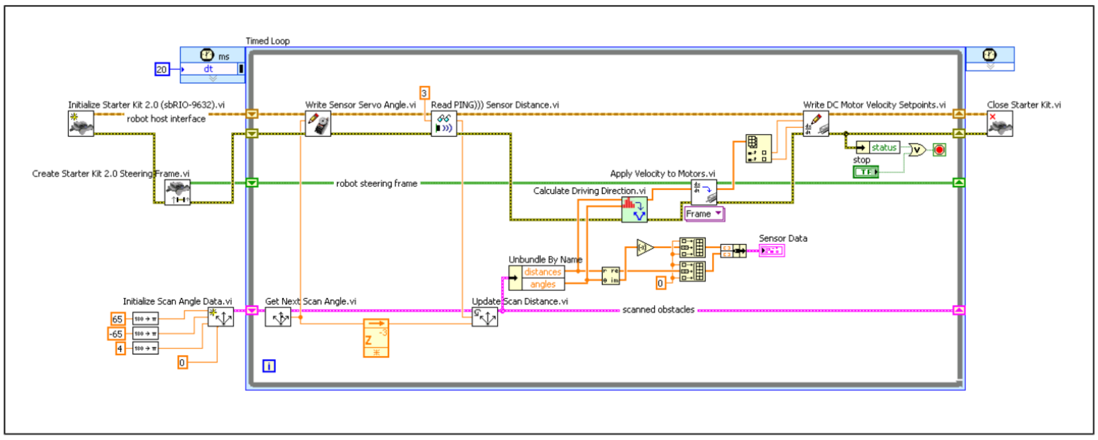

Timed Loop
ms
dt
20
Initialize Starter Kit 2.0 (sbRIO-9632).vi
robot host interface
Write Sensor Servo Angle.vi
3
Read PING))) Sensor Distance.vi
Write DC Motor Velocity Setpoints.vi
Close Starter Kit.vi
status
stop
Create Starter Kit 2.0 Steering Frame.vi
robot steering frame
Apply Velocity to Motors.vi
Calculate Driving Direction.vi
Frame
Sensor Data
Unbundle By Name
distances
angles
0
Initialize Scan Angle Data.vi
65
-65
4
0
Get Next Scan Angle.vi
Update Scan Distance.vi
scanned obstacles

Figure 4-3 Block diagram of Starter Kit 2.0 with NI sbRIO-9632

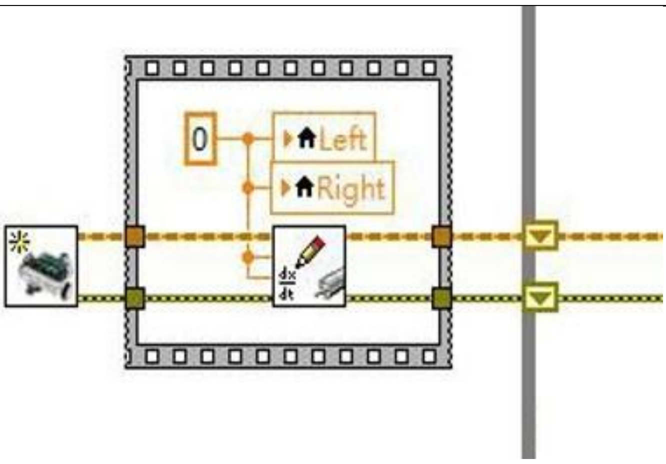


Figure 4-4LabView Program Flow

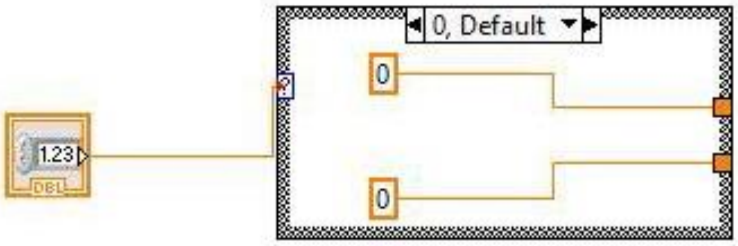


Figure 4-5 Stop Branch

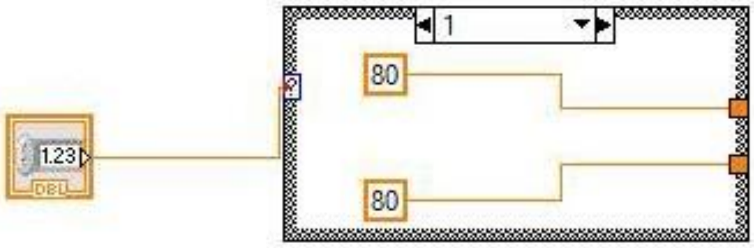


Figure 4-6 Forward Branch

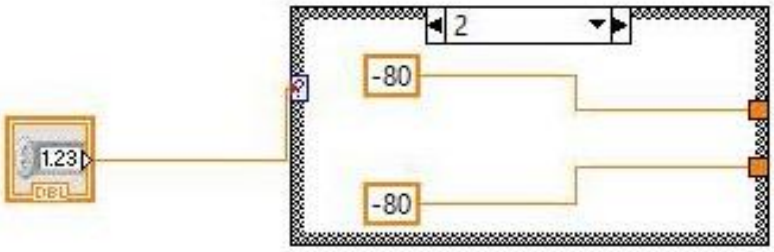


Figure 4-7 Backward Branch

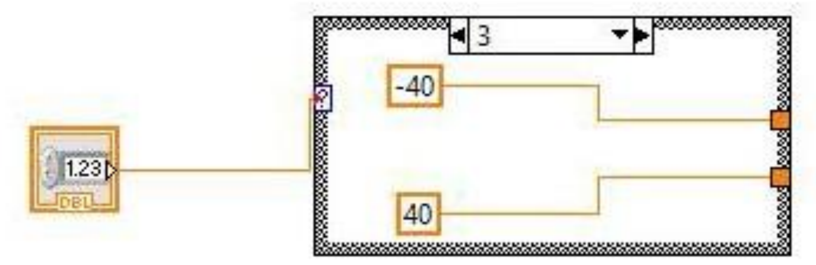


Figure 4-8 The left Branch

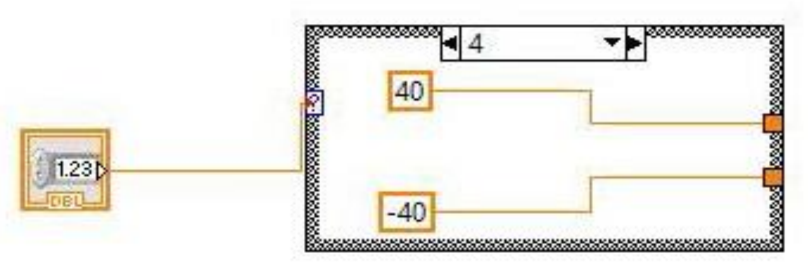


Figure 4-9 The Right Branch

4.3 Command Correspondence to Different Branch Structures

The left and right wheel motor rotation is reversed, so be on the right wheel negate the desired effect has been reached. Afterwards the value of the variable left and right of transformation that changed in to the unit of rad/s, and then multiplied by the gear ratio 2. Then the Left and Right modules writes motor through the motor speed. The two numeric controls on the front panel correspond to the value of the slider motor speed writing module. Then Up and Down on both sides of the two thick lines followed by a start signal and a trolley car wrong signal are carried throughout the program. Da-NI robot drive module requires access to these two signals, so after the drawing of the two signals is no longer Ao said.

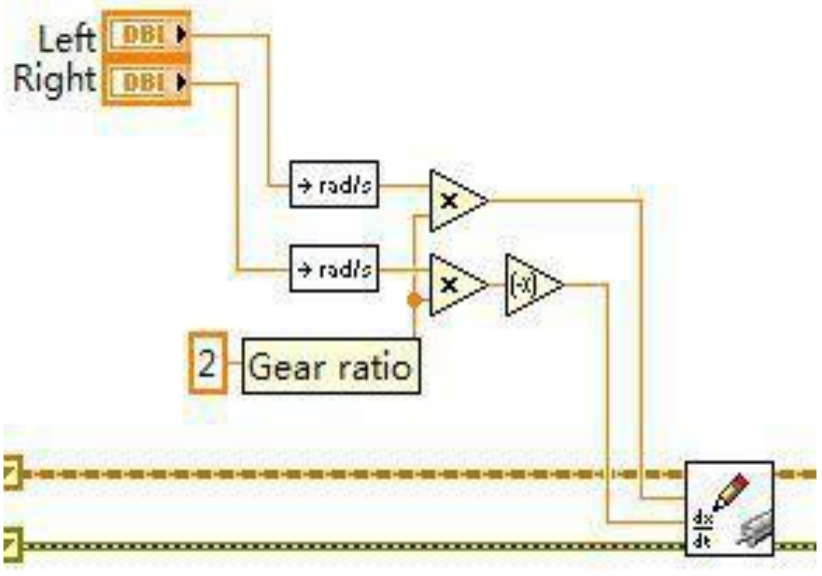


Figure 4-10 Command Correspondence to Different Branch Structutres

4.4 Out of the loop Oscilloscope and Buttons

The front panel layout program design, front panel and block diagram LabVIEW program section, completes the control and programming power to move the car and then stop off in accordance with the command of the whole process.

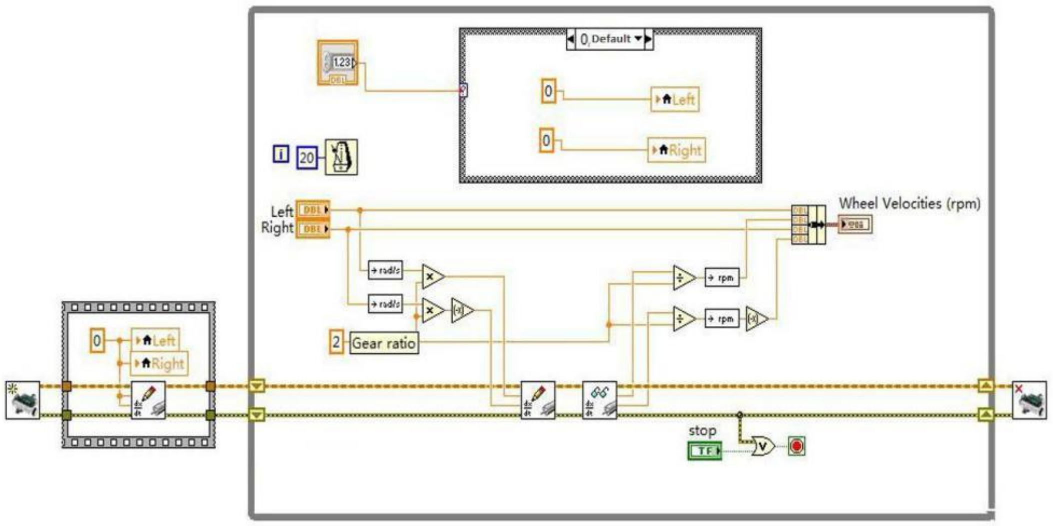


Figure 4-11 Out of the Loop Osilloscope and Buttons

5 Finalizing And Debugging

5.1 Kinect Studio

As we look into this whole gesture control scenario, debugging of the program should be a nightmare. (i.e. which should be difficult) Because, in order for Kinect to capture the images, we should stand around 4-5 meters away from the Kinect device. Also to get better debugging effect, which a distance range will be compressed smaller, that is, act as developers in debugging the time, we had to leave from beside the computer. And debugging should be done in front of the computer at the same time, which sounds impossible and not practical at all.

However this is where Kinect Studio comes into play. This helps the developer to build an extremely useful development tools, and contents are very simple. We can record a video using Kinect for a certain amount of time to get all the data, including images, sound, depth and so on. Once we record the video, simply use the 'play' command to playback the required debugging action. So that we can sit in front of the computer watching the debugging process and results. Not only that, we can use it to a positive image of a frame or a section of video repeatedly play, easy to observe the details.

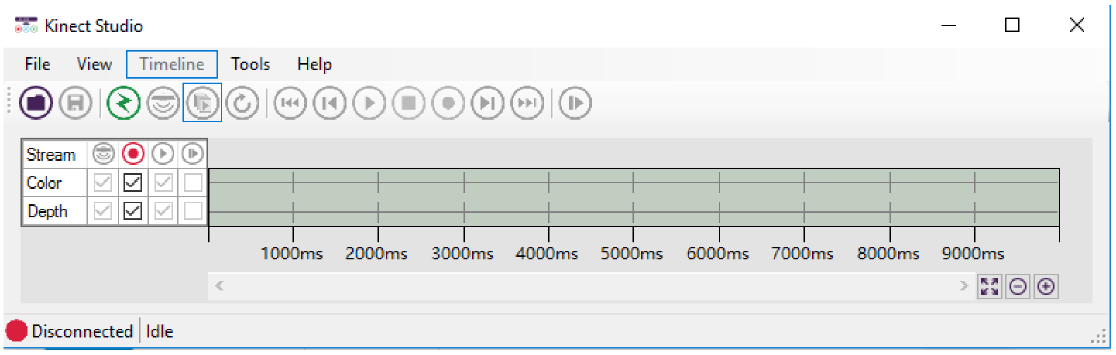
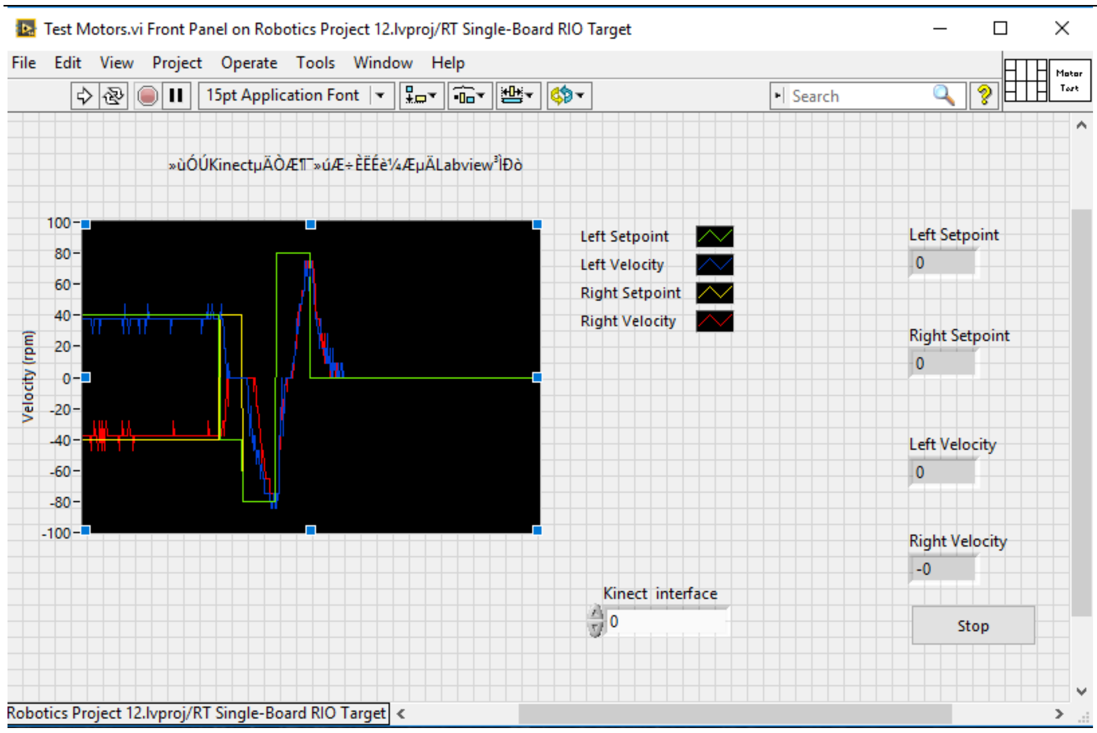


Figure 5-1 Kinect Studio

5.2 Running /Debugging/Finalizing

There are five actions that we need to accomplish in this mobile robot. Those are, Stop, Move Forward, Move Backward, Turn Left and Turn Right. And the right hand is used to make gestures in order to move the robot. Hand on the chest, hand up, hand down, left and right are the gestures accordingly. Next, launch the Kinect Studio, opening the Record button, go to the appropriate location, right in the chest by way of Kinect into the scope of monitoring, etc. after the full access, extended his right hand up, return the chest for a brief time, to complete his right hand upward movement, then do down the right hand, right hand to the left, right hand movements, which are similar to the right hand upward movement process. After exiting the monitoring range of Kinect, stop recording and save. We get a fixed format file.


Test Motors.vi Front Panel on Robotics Project 12.lvproj/RT Single-Board RIO Target
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
Motor Test
»ùÓÚKinectµÄÒÆ¶¯»úÆ÷ÈËÉè¼ÆµÄLabview³ÌÐò
100
80
60
40
20
0
-20
-40
-60
-80
-100
Velocity (rpm)
Left Setpoint
Left Velocity
Right Setpoint
Right Velocity
Stop
Robotics Project 12.lvproj/RT Single-Board RIO Target

Left Setpoint: 0
Right Setpoint: 0
Left Velocity: 0
Right Velocity: -0
Kinect interface: 0

Figure 5-2 A modify Test Motors with Kinect integrated

5.3 Operation and Running

Using Kinect we compile programs written in C# programming language. We also can generate solutions and check for errors. Once we launch the LabVIEW program, we should select motor test VI files (LabVIEW debugging program is based on the motor adaption of LabVIEW program), the two procedures breakdown on the screen for easy observation. After opening the LabVIEW, select the file we saved of '.xed' format selection single play button and then the cursor LabVIEW front panel of the command input box on the "value" to wait for the debugger and Run, relevant documents and video on the CD will be presented.

5.4 Output

After subsequent testing, we were able to accomplish our goal which was to make the care move in five different directions. Therefore we can say the design was quite successful. But there are some flaws, such as from human action to make the car started to have about 0.5 seconds of delay between the instruction, which is mainly due to the wait for the next frame refresh the image as well as the loop twice circulation loop structure between LabVIEW program execution procedures to use when. But overall, Da-NI motion of the robot is still very good completed the tasks

6 Improvements And Conclusions

6.1 Future Improvement to the project

Firstly the Kinect sensor has some disadvantages of making skeletal tracking and depth imaging, by making these sensors more accurate we believe that it would be more useful even than it is now. And also could make the IR emitter grasp data during day light which would be helpful to use the Kinect outdoors. I hope making the microphone array more accurate by implementing background noise cancellation technology, will result in handy on speech recognition service.

Also there were some difficulties on detecting color. I hope by adding a camera with more pixel range could solve this problem. Last but not least, the services written for this project are open source code as mandated by the project sponsor. This allows for a variety of future improvements on the platform as the sponsor sees fit. The robot is easily reconfigurable, making it a quality educational product. You need to stand at least 6 feet away from the sensor for it to see you, and about 8 feet from it if you are playing next to someone else. This can be a struggle in some settings. Nyko released Zoom for Kinect, which promises to reduce the play range required by up to 40%. Instead of having to play 6 to 8 feet away from the sensor, Zoom enables you to play 4 to 6 feet away from it. We hope the Kinect would include a zoom on itself to prevent this struggle.

6.2 Conclusions

Using Kinect somatosensory sensors via a computer program to control the movement of a robot is a bit advanced step of robotics control system. The gesture recognition system made the HRI far more superior. The communication system was bought into a new level through my project. I think the device Kinect will make a huge influence on the robotics field in the next few decades. As the Microsoft has the patent for Kinect, they would be able to make huge profits and manipulate the industry for some extent. The improvement of speed and reliability of the control, the amount of information transmitted human posture is far more than hardware media. Such as the human body to control a humanoid robot, so the original high complex control becomes very simple.

National Instruments provides this LabVIEW Robotics starter kit also known as DaNI Robot, which make the study of robotics more vivid. To allow the robots to operate in different networks, they are configured to get their IP address via DHCP. This is a much easier way to access the robot. We also designed and applied a real human skeleton point conversion module, body height measuring module, pose determining module, wheel speed and wheel speed write control module and read modules. All these modules contain a lot of mathematical algorithms and advanced design concepts.

Overall the research was a success and was able to achieve the goal I desire. I think we will be able to make use of this theory and technology, for far more advanced cases and make the maximum use of robotics.

Acknowledgments

First of all, I would like to thank my family for raising me and making me what I am. Thank you for always being by my side during this long and lonely journey.

I would like to express my deepest gratitude to my supervisor Mr. Ji Li for his kind guidance, advice and wisdom he granted upon throughout the period of this project. I would also like to thank my classmates who supported and encouraged me during this project in various ways.

Thank you.

References

[1] Young Joon Kim and Yong-Ho Seo, “Simple Programming Language for Creating a Simulation Environment with Mobile Robotics”, *International Journal of Smart Home* Vol. 7, No. 4, July, 2013.

[2] Bitter, Rick et al "Introduction to LabVIEW", *LabVIEW Advanced Programming Techinques,* Boca Raton: CRC Press LLC, 2001.

[3] Kyle Johns , Trevor Taylor, “Professional Microsoft® Robotics Developer Studio”, Wiley Publishing, Inc., Indianapolis, Indiana, 2008.

[4] David Catuhe, “Programing with the Kinect for Windows Software Development Kit”, Microsoft Press, 2012.

[5] Abhijit Jana, “Kinect for Windows SDK Programming Guide”, Packt Publishing Ltd, 2012.

[6] Hans-PetterHalvorsen, “Introduction to LabVIEW”, University College of Southeast Norway, 2016-09-07.

[7] Aaron Staranowicz, Gian Luca Mariottini, “Robotic Simulation Guide”, *ASTRA Robotics Lab, Department of Computer Science and Engineering, University of Texas at Arlington*.

[8] Trevor Taylor, “Using Kinect with Robotics Developer Studio”.

[9] AbhishekKar, “Skeletal Tracking using Microsoft Kinect”, *Department of Computer Science and Engineering, IIT Kanpur*.

[10] D. Pagliari, F. Menna, R. Roncella, F. Remondino, L. Pinto, “KINECTFUSION IMPROVEMENT USING DEPTH CAMERA CALIBRATION”, *The International*

Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XL-5, 2014.

[11] Matthew MacDonald, “Pro WPF in C# 2010: Windows Presentation Foundation in .NET 4.0”, *ISBN-13 (electronic): 978-1-4302-7204-5,* 2010.

[12] Ian Griffiths, Chris Sells, “Programming Windows Presentation Foundation”, *ISBN: 0-596-10113-9,* September 2005.

[13] DavideVitelaru, “Visual C# Programming Basics”, *http://davidevitelaru.com/.*

[14] Adam Nathan, “WPF 4 Unleashed”, *800 East 96th Street, Indianapolis, Indiana 46240 USA,* 2010.

[15] DR. ROBERT KING, “Mobile Robotics Experiments with DaNI”,*COLORADO SCHOOL OF MINES.*

[16] SubarnaSinha, Suman Deb, “Depth Sensor Based Skeletal Tracking Evaluation for Fall Detection Systems”, *International Journal of Computer Trends and Technology (IJCTT)* – volume 9 number 7– Mar 2014.

[17] Stefan Reifinger, Frank Wallhoff, Markus Ablassmeier, Tony Poitschke, Gerhard Rigoll, “Static and Dynamic Hand-Gesture Recognition for Augmented Reality Applications”, *TechnischeUniversitätMünchen, Institute for Man Machine Communication, Theresienstraße 90, 80333 Munich, Germany*.

[18] Matthew Fitzpatrick, NikolaosMatthiopoulos, “Real Time Person Tracking and Identification using the Kinect Sensor”, *Major Qualifying Project in Electrical & Computer Engineering,*4/25/2013.

[19] Natural User Interface: the Future is Already Here. *Design float blog.* [Online] http://www.designfloat.com/blog/2013/01/09/natural-user-interface/.

[20] Kinect for Windows Sensor Components and Specifications. *MSDN.* [Online] MICROSOFT. http://msdn.microsoft.com/en-us/library/jj131033.aspx.

[21] Getting Started with DaNI Robot.

[Online]http://faculty.salina.k-state.edu/tim/robotics_sg/Robotics/DaNI.html [22] WenjunZeng, “Multimedia at Work”, *Microsoft Kinect Sensor and its Effects,* Microsoft Research, April-June 2012.

[23] Kinect for Windows SDK Beta, *Microsoft Research.*

[24] Jinfan Chen “AN INTELLIGENT HUMAN-TRACKING ROBOT BASED-ON KINECT SENSOR” December 2015.

Appendix I – Program Codes

MainWindow.xaml

```
<Window x:Class="kinectPPTControl.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="764" Width="716" Loaded="Window_Loaded">
    <Grid>
        <Image Name="ColorImage" Margin="0,-26,27,122" />
        <Canvas Name="skeletonCanvas" Visibility="Visible" >
            <Ellipse Canvas.Left=" 0" Canvas.Top=" 0" Height="10"
Name="righthand" Width="10" Fill="Red" />
        </Canvas>
        <Canvas Background="Transparent" Name="SkeletonCanvas">
            <TextBox Canvas.Left="31" Canvas.Top="609" Height="79"
Name="textBox1" Width="561" />
        </Canvas>
        <TextBox Height="126" HorizontalAlignment="Left" Margin="10,10,0,0"
Name="textBox2" VerticalAlignment="Top" Width="226" FontSize="72" />
    </Grid>
</Window>
```

Class1.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Kinect;

namespace kinectPPTControl
{
    class RealPosition
    {
        public double realX;
        public double realY;
        public double realZ;
        public RealPosition(Joint a)
        {
            this.realZ = a.Position.Z;
            this.realX = System.Math.Tan(Math.PI * 28.5 / 180) * a.Position.Z *
a.Position.X;
            this.realY = System.Math.Tan(Math.PI * 21.5 / 180) * a.Position.Z *
a.Position.Y;
        }

    }
}
```

Class2.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace kinectPPTControl
{
    class CenterPosition
    {
        public double X;
        public double Y;

    }
}
```

MainWindow.xaml.cs

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using Microsoft.Kinect;//Kinect reference

using System.Windows.Forms;

using kinectPPTControl;

using System.Runtime.InteropServices;


namespace kinectPPTControl

{

    /// <summary>

    /// MainWindow.xaml Thw interaction
    /// </summary>

    public partial class MainWindow : Window

    {

        public bool isBackGestureActive = false;

        public bool isForwardGestureActive = false;

        public bool isLeftGestureActive = false;

        public bool isRightGestureActive = false;//Four   Boolean variables
    are defined for the current record is what kind of attitude

        KinectSensor kinectSensor;

        private byte[] pixelData;//Define an array for storing the color
image data

        private Skeleton[] skeletonData;//Define an array for storing data
bones

        public double height;//Defined variable height（double type）

        CenterPosition centerPosition;//definition
CenterPositionVariablecenterPosition

        private readonly int MOUSEEVENTF_LEFTDOWN = 0x0002;//Simulation of
the left mouse button pressed

        private readonly int MOUSEEVENTF_MOVE = 0x0001;//Simulate mouse
movement

        private readonly int MOUSEEVENTF_LEFTUP = 0x0004;//Simulation of the
left mouse button lift


        [DllImport("user32")]

        public static extern void mouse_event(int dwFlags, int dx, int dy,
int dwData, int dwExtraInfo);//Define a mouse event


        public MainWindow()

        {

            InitializeComponent();

        }


        private void Window_Loaded(object sender, RoutedEventArgs e)

        {

            kinectSensor = (from sensor in KinectSensor.KinectSensors where
sensor.Status == KinectStatus.Connected select
```

```
sensor).FirstOrDefault();//By selecting Kinect sensor to obtain the label
first or default, then get Kinect sensor


kinectSensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30)
;//Itallows the sensor to acquire color image information, and attribute
setting

           kinectSensor.Start();//KinectPower

           kinectSensor.ColorFrameReady +=
kinectSensor_ColorFrameReady;//Capturing a color image

           kinectSensor.SkeletonStream.Enable();//Get information allows
the sensor to the bone

           kinectSensor.SkeletonFrameReady += new
EventHandler<SkeletonFrameReadyEventArgs>(kinectSensor_SkeletonFrameReady);
//Get skeletal information

       }

       private void kinectSensor_ColorFrameReady(object sender,
ColorImageFrameReadyEventArgs e)

       {

           using (ColorImageFrame imageFrame = e.OpenColorImageFrame())

           {

               if (imageFrame != null)

               {

                   this.pixelData = new byte[imageFrame.PixelDataLength];

                   imageFrame.CopyPixelDataTo(this.pixelData);

                   this.ColorImage.Source =
BitmapSource.Create(imageFrame.Width, imageFrame.Height, 96, 96,
PixelFormats.Bgr32, null, pixelData, imageFrame.Width *
imageFrame.BytesPerPixel);//When the color image information is ready, it
will be displayed in the Image1

               }

           }

       }

       private void kinectSensor_SkeletonFrameReady(object sender,
SkeletonFrameReadyEventArgs e)

       {

           using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())

           {

               if (skeletonFrame != null)
```

```
                {

                    skeletonData = new
Skeleton[kinectSensor.SkeletonStream.FrameSkeletonArrayLength];

                    skeletonFrame.CopySkeletonDataTo(this.skeletonData);

                    Skeleton skeleton = (from s in skeletonData where
s.TrackingState == SkeletonTrackingState.Tracked select
s).FirstOrDefault();

                    if (skeleton != null)

                    {

                        SkeletonCanvas.Visibility = Visibility.Visible;

                        ProcessGesture(skeleton);

                        textBox1.Text = " Get Skeleton Data!! ";//When ready
skeletal information, call the posture detection function ProcessGesture,
and displays the information ready to complete skeleton of the prompt text
in textBox1


                    }

                }

            }

        }


        private void ProcessGesture(Skeleton s)

        {

            Joint rightHand = (from j in s.Joints where j.JointType ==
JointType.HandRight select j).FirstOrDefault();

            Joint head = (from j in s.Joints where j.JointType ==
JointType.Head select j).FirstOrDefault();

            Joint centerShoulder = (from j in s.Joints where j.JointType ==
JointType.ShoulderCenter select j).FirstOrDefault();

            Joint rightfoot = (from j in s.Joints where j.JointType ==
JointType.FootRight select j).FirstOrDefault();//Being right hand, head,
shoulder and foot bones midpoint of the original information

            RealPosition realPositionRightHand;

            RealPosition realPositionHead;

            RealPosition realPositionRightFoot;

            RealPosition realPositionCenterShoulder;//The definition of the
right hand, head,  shoulder and foot bones midpoint of the true position
variable (realPosition category)
```

```
            realPositionRightHand = new RealPosition(rightHand);

            realPositionHead = new RealPosition(head);

            realPositionRightFoot = new RealPosition(rightfoot);

            realPositionCenterShoulder = new 
RealPosition(centerShoulder);//The right hand, head, shoulder and foot 
bones midpoint of the original information into skeletal real location, and 
given to the appropriate properties (see specific process Class1.cs)

            height = realPositionHead.realY - 
realPositionRightFoot.realY;//Users measuring height

            centerPosition = new CenterPosition();

            centerPosition.X = realPositionCenterShoulder.realX + 0.1 * 
height;

            centerPosition.Y = realPositionCenterShoulder.realY - 0.1 * 
height;//Set reference point X, Y coordinate values

            textBox2.Text = Convert.ToString(height);

            if (realPositionRightHand.realX < centerPosition.X - height * 
0.2)//Detecting whether the left hand

            {

                if (isLeftGestureActive)

                { }//If you have left is not issued a directive (here it is 
mainly to prevent repeat sending the same instruction, wasting CPU memory)

                else

                {

                    mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0);

                    mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0);//Click

                    System.Windows.Forms.SendKeys.SendWait("{Backspace}");

                    System.Windows.Forms.SendKeys.SendWait("{Backspace}");

                    System.Windows.Forms.SendKeys.SendWait("{Backspace}");


System.Windows.Forms.SendKeys.SendWait("{Backspace}");//Delete an existing 
command

                    System.Windows.Forms.SendKeys.SendWait("3");//Enter the 
appropriate command

                    System.Windows.Forms.SendKeys.SendWait("{Enter}");

                    System.Windows.Forms.SendKeys.SendWait("{Enter}");

                    System.Windows.Forms.SendKeys.SendWait("{Enter}");


System.Windows.Forms.SendKeys.SendWait("{Enter}");//Implementation of
```

```
existing directives (repeated here mainly to prevent leakage of the key,
because the situation of several key leak appeared in the previous
debugging)

                    isForwardGestureActive = false;

                    isBackGestureActive = false;

                    isLeftGestureActive = true;

                    isRightGestureActive = false;//Update four Boolean

                }

            }

            if (realPositionRightHand.realX > centerPosition.X + height *
0.2)//Detecting whether the right hand

            {

                if (isRightGestureActive)//If you have the right not to
issue instructions

                { }

                else

                {

                    mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0);

                    mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0);//Click

                    System.Windows.Forms.SendKeys.SendWait("{Backspace}");

                    System.Windows.Forms.SendKeys.SendWait("{Backspace}");

                    System.Windows.Forms.SendKeys.SendWait("{Backspace}");


System.Windows.Forms.SendKeys.SendWait("{Backspace}");//Delete   an
existing command

                    System.Windows.Forms.SendKeys.SendWait("4");//Enter the
appropriate command

                    System.Windows.Forms.SendKeys.SendWait("{Enter}");

                    System.Windows.Forms.SendKeys.SendWait("{Enter}");

                    System.Windows.Forms.SendKeys.SendWait("{Enter}");


System.Windows.Forms.SendKeys.SendWait("{Enter}");//Implementation of
existing directives (repeated here mainly to prevent leakage of the key,
because the situation of several key leak appeared in the previous
debugging)

                    isForwardGestureActive = false;

                    isBackGestureActive = false;
                    isLeftGestureActive = false;

                    isRightGestureActive = true;//Update four Boolean

                }

            }

            if (realPositionRightHand.realX > centerPosition.X - height * 
0.2 && realPositionRightHand.realX < centerPosition.X + height * 0.2 && 
realPositionRightHand.realY < centerPosition.Y + height * 0.2 && 
realPositionRightHand.realY > centerPosition.Y - height * 0.2)

            {

                if (!isForwardGestureActive && !isBackGestureActive 
&&!isLeftGestureActive &&!isRightGestureActive)

                { }

                else

                {



                    mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0);

                    mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0);//Click

                    System.Windows.Forms.SendKeys.SendWait("{Backspace}");

                    System.Windows.Forms.SendKeys.SendWait("{Backspace}");

                    System.Windows.Forms.SendKeys.SendWait("{Backspace}");


System.Windows.Forms.SendKeys.SendWait("{Backspace}");//Delete and existing 
command

                    System.Windows.Forms.SendKeys.SendWait("0");//

                    System.Windows.Forms.SendKeys.SendWait("{Enter}");

                    System.Windows.Forms.SendKeys.SendWait("{Enter}");

                    System.Windows.Forms.SendKeys.SendWait("{Enter}");

                    System.Windows.Forms.SendKeys.SendWait("{Enter}");//

                    isForwardGestureActive = false;

                    isBackGestureActive = false;

                    isLeftGestureActive = false;

                    isRightGestureActive = false;//

                }

            }
```

```
           if (realPositionRightHand.realY > centerPosition.Y + height * 
0.2)
           {
               if (isForwardGestureActive)
               { }
               else
               {
                   mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0);
                   mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0);//
                   System.Windows.Forms.SendKeys.SendWait("{Backspace}");
                   System.Windows.Forms.SendKeys.SendWait("{Backspace}");
                   System.Windows.Forms.SendKeys.SendWait("{Backspace}");
                   System.Windows.Forms.SendKeys.SendWait("{Backspace}");//
                   System.Windows.Forms.SendKeys.SendWait("1");//
                   System.Windows.Forms.SendKeys.SendWait("{Enter}");
                   System.Windows.Forms.SendKeys.SendWait("{Enter}");
                   System.Windows.Forms.SendKeys.SendWait("{Enter}");
                   System.Windows.Forms.SendKeys.SendWait("{Enter}");//
                   isForwardGestureActive = true;
                   isBackGestureActive = false;
                   isLeftGestureActive = false;
                   isRightGestureActive = false;//


               }
           }
            if (realPositionRightHand.realY < centerPosition.Y - height * 
0.2)
            {
                if (isBackGestureActive)
                { }
                else
                {
                    mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0);
```

```
                        mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0);//
                        System.Windows.Forms.SendKeys.SendWait("{Backspace}");
                        System.Windows.Forms.SendKeys.SendWait("{Backspace}");
                        System.Windows.Forms.SendKeys.SendWait("{Backspace}");
                        System.Windows.Forms.SendKeys.SendWait("{Backspace}");//
                        System.Windows.Forms.SendKeys.SendWait("2");//
                        System.Windows.Forms.SendKeys.SendWait("{Enter}");
                        System.Windows.Forms.SendKeys.SendWait("{Enter}");
                        System.Windows.Forms.SendKeys.SendWait("{Enter}");
                        System.Windows.Forms.SendKeys.SendWait("{Enter}");//
                        isForwardGestureActive = false;
                        isBackGestureActive = true;
                        isLeftGestureActive = false;
                        isRightGestureActive = false;//
                    }
                }
            }
        }
}
```

Appendix I – List of Figures

Appendix II – List of Tables