HUMAN MOTION TRACKING USING MOBILE ROBOT WITH KINECT 3D SENSOR

A Project Report

by

RAMANAN D (117012032)

Bachelor of Technology in Robotics & Mechatronics

SASTRA UNIVERSITY

Tirumalaisamudram-613 401

ABSTRACT

KEYWORDS: Human tracking robot, Kinect, Motion Detection, tracking, RGB and Depth Sensors

The development of robotics has become more vital because of their extensive applications in industries, the military, hospitals, and homes. Most of these are highly smart machines that are generally not in common use yet. Human tracking and the following robot are some of the finest applications to be brought into use. This can serve various purposes like spying on someone of high interest, monitoring baby/old people/patients, carrying households, and other numerous activities. This has to be of affordable cost to become familiar among common people. Most human tracking robots use markers on the human body which is not advisable for infants, the elderly, and the sick.

Kinect is a motion-sensing input device by Microsoft. Kinect is a very cheap and efficient device that observes a person's motion in 3D and helps us to develop open-source drivers. Kinect can capture human motion by a skeleton model, which presents us position and distance data of human motion in 3D. This project illustrates tracking of human motion with Kinect mounted on a mobile robot. Kinect aids us by contributing depth information and RGB information. Microsoft Kinect's depth and RGB image data of humans obtained in real-time enables us to control the velocity and angular position of the mobile robot directly.

The mobile robot is controlled to track the moving human in the view of Kinect. This can be achieved by not allowing the human to escape out of the robot's vision. The robot moves either forward or backward based on the human's motion to maintain a constant distance between them. It also turns right or left to retain the skeleton at the center of its vision.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Human tracking is the process of monitoring human beings by means of electronic sensors and observing characteristic behavior. The advancement in the fabrication of these sensors and technology in processing their crude information or data has also risen the requirement for obtaining the correct and efficient device for the same. The existing tracking mechanism offer two-dimensional scanning of a given subject to be tracked, which proves to be disadvantageous during the positional estimation of that subject, with the 3D monitoring machines requiring multiple points of installation and monitoring/observing. Introducing a 3D platform for human tracking purposes from a single machine observation provides significant ease in obtaining the position of the subject. A simple IR sensor array for area scanning and object tracking can provide three-dimensional information of the subject with the help of a digital camera. For the processing of the sensor information and obtaining the characteristic behavior existing computer algebra system software such as MATLAB would suffice.

Without the need to design or fabricate an ideal sensor system for the purpose of object tracking comes in the form of a powerful device from Microsoft Corporation, the KINECT. Microsoft Kinect provides the ideal system whose functions can be implemented into human tracking scenarios. Kinect's architecture is designed for easy and spontaneous human body recognition and positional estimation. This framework evidently proves to have huge potential in the fields of image processing and object tracking. COROBOT, a robot DIY architecture, provides a proper connection with the Kinect using third-party software like PHIDGETS.

1.1.1 Microsoft KINECT Sensor:

Kinect, introduced as Project Natal by Microsoft Corporation, is a device that comes under the motion-sensing input device category. Kinect, primarily developed for motion recognition for gaming purposes slowly entering into the field of motion tracking and monitoring for its ease in procurement and processing, is a single body system housing an RGB camera, an IR depth sensor, and a multi-array microphone whose proprietary software and APIs can be used to processing its sensor information. It can easily be integrated with computer algebra system software such as MATLAB providing the build of a useful system for human tracking.

Advantages of using Kinect for sensing purposes:

i. Premium quality and accurate sensors

ii. Strong sensor housing

iii. Efficient proprietary software availability

iv. Ease of procurement

v. ease in software integration, by use of existing APIs

1.1.2 COROBOT:

A four-wheeled development platform centered on an onboard mini ITX computer. Mini ITX is an Intel-produced microcomputer that provides numerous robot building services with the help of its socket build modular system architecture. Socket system provides direct product integration with the computer for a faster and seamless connection, which can also be used for performance enhancement purposes. CoroWare produced bots come in various configurations such as four-wheel drive, two-wheel drive, etc. The two-wheel drive provides sufficient machine movement with limited resources for the required project. A 2 wheel body with castor at the rear end provides standard stability and ease of control and movement. One socket on the mini ITX processor is dedicated for Phidget integration and standardized connection between the computer hosting the processing software and the Corobot. Phidget application can be burned onto the mini ITx computer thereby giving ease in administrating the bot, also providing wide testing capabilities. The Corobot hosts additional resources such as integrated batteries, laptop stand, etc. Which thus becomes an ideal machine for human tracking and processing requirements.

Advantages of using Corobots:

i. High-performance mini ITX computer

ii. Accurate and agile bot movement

iii. Easy for code burning and testing

iv. Overwhelming battery charge availability

1.1.3 PHIDGET:

Phidget is a physical GUI widget focused on graphical representation and actuation of physical entities and models. With the help of a personal computer, these PHIDGETS can be used for resource control. Phidgets prove to be very cheap considering the work that can be extracted from them. They provide a simple and effective way to configure and test the machine components. Accurate test input in terms of power and other constraints can be provided to the motors and other actuators using these phidgets.

Advantages of PHIDGETS:

i. easiest way for resource testing

ii. easy to learn and use

iii. low system requirement and consumption

1.1.4 Hosting computer:

The hosting computer facilitates a platform for the processing of sensor data by using the available APIs for the different systems used and implemented. It is required to provide high performance since it has to get values and processes in quick time for human tracking purposes. Providing the required storage and booting facility for the computer algebra system which is the brain of the machine. The ideal computer algebra software for such bots is MATLAB. Matlab is an online powerful software that can help process the information that it can procure with the system-specific API calls. The higher version of this software provides better performances.

Disadvantages of the hosting computer:

i. Additional weight on the bot, decreases performance

ii. Supreme quality computer required to extract perfect work

CHAPTER 2

LITERATURE SURVEY

A journal by **Eiji Machida, Meifen Cao, and Toshiyuki Murao** on the Tracking control system of human motion is taken as the base paper where the design of a care robot that watches over and supports old people's daily life is discussed. It uses Kinect to reduce the cost to become widely available in the ordinary household. Kinect is very cheap and efficient. Kinect is a motion-sensing input device by Microsoft for the Xbox 360 video game console and Windows PCs. With Kinect for Windows SDK, the features of Kinect devices, such as skeletal tracking, can be developed and applied. This paper also proposes methods that make estimation and human tracking effective. The experimental results prove that Kinect 3D sensor is very useful because of its low price and the ability to extract bone models in 3D. It is also found that its resolution is sufficient to track the human walking with a velocity be less than one meter per second. Furthermore, image processing is relatively fast because of the special-purpose processor in Kinect 3D. On the other hand, the ordinary problems in tracking humans with vision systems are unresolved. That is the image recognition problems, including light condition, spatial resolution, sampling rate, and so on, sometimes cause the disturbance within tracking human motion, so parameter tuning and certain filtering are required for stable tracking.

2.1 TRACKING CONTROL:

The robot's objective is to learn human motion and provide motor actuation through computer processing within a single computer working cycle. Basic movement is evaluated with the error obtained by the three-dimensional bone model data structure. Particular error values provide the proportional feed current to the two-wheel motors. This feed current is regulated and chiefly focused on maintaining the subject(Human Being) in a given dead zone in front of the robot. The control however is noticed to be linear, which results in irregular motion and unwanted jerks.

Motion capture, facial recognition, voice recognition, etc. These are just a few of the many features Microsoft made available to everyone with the release of the Kinect and the Kinect SDKs. For this project, the use of devices other than Kinect could have been considered, being that probably most of the research will be done based on its RBG camera. But the fact is that Kinect is able to integrate all these different technologies in one small device, making it a very powerful tool.

Therefore, and taking into consideration all the features and possibilities Kinect offers, we think that this device will be suitable for the successful accomplishment of the proposed goals. We also think that if the integration of Kinect with the “Human Tracking robot” project is successful, other than a suitable device for our experiment, it will be a device that, due to its characteristics, will allow researchers to go much further in their investigations, allowing them to explore new and different areas

CHAPTER 3

SENSORS AND CONTROLLERS

3.1 MICROSOFT KINECT SENSOR DEVICE

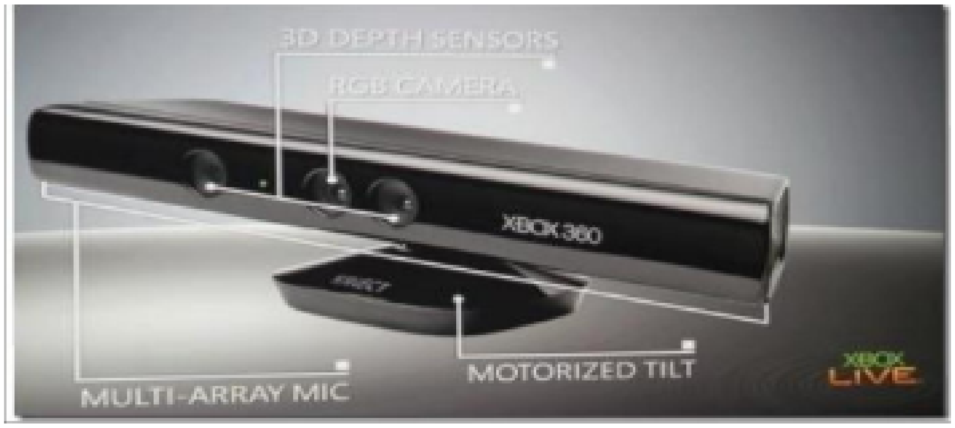


Fig 3.1: Microsoft Kinect

Microsoft revolutionized the world of video games and video games consoles when they introduced, in November 2010, Kinect for Xbox 360. Kinect is an interactive, motion-sensing gaming device for the Microsoft Xbox 360. In its body, Kinect includes a multiple array microphone, a regular RGB camera, and a pair of 3-D depth-sensing range cameras (see Figure 9). All these technologies allow Kinect to track objects and up to six people in three dimensions, as well as to recognize faces and voices and provide full-body 3D motion capture.

Both RGB and depth sensors work at a frame rate of 30Hz. The depth sensor combines an infrared laser projector with a monochrome sensor, allowing it to capture video data in 3D under any ambient light condition. This sensor uses VGA resolution (640x480 pixels) with 11- bits depth, while the RGB sensor uses 8-bit VGA resolution (640x480 pixels). Kinect's working space (playing area) has been identified as the main problem of this device. To work properly, a user has to be within a range of 1.2 to 3.5 meters from the device and within an area of approximately 6m^2. Despite these limitations, it has been seen that the sensor can still track a user within a range of approximately 0.7 to 6 meters.

Fig 3.2: Motion Sensor

Fig 3.3: Skeletal Tracking

Fig 3.4: Voice Recognition

With Kinect, players no longer need to memorize the different functions of conventional hand controllers, because now the player is the controller. Figures 10, 11, 12, and 13 show how Kinect technology works. After presenting itself as one of the most spectacular console controllers ever, Microsoft rapidly realized that Kinect had a huge potential for other things than just games and released, in mid-2011 and late-2011, non-commercial and commercial versions of its official SDK, respectively. With these releases, Kinect rapidly expanded to PCs and became

a platform for programmers and companies to develop new and innovative products.

Kinect is a relatively recent technology, no more than one and a half years old. But being such an advanced and novel piece of hardware, a lot of research has already been performed on it. That research includes studies in fields that vary from physical rehabilitation to robotics. A good example of its use in robotics is NASA's integration of Kinect with their prototype robots [24]. The LIREC project5 (Living with Robots and Interactive Characters) is also another good example of Kinect's integration in robotics. This project is a collaboration between several entities (universities, research institutes, and companies) from several different countries, Heriot-Watt is one of the partner universities. Heriot-Watt researchers have been integrating Kinect with their prototype robot and studying how can it be used to facilitate human-robot interaction.

Efficiently tracking and recognizing hand gestures with Kinect is one of the fields that is getting more attention from researchers. This is a complex problem but it has a number of diverse applications, being "one of the most natural and intuitive ways to communicate between people and machines". Regarding full-body poses recognition, E. Suma et al. [29] developed a toolkit that allows customizable full-body movements and actions to control, among others, games, PC applications, virtual avatars, and the onscreen mouse cursor. This is achieved by binding specific movements to virtual keyboard and mouse commands that are sent to the currently active window.

In a different perspective, F. Kistler et al. [32] adopted a gamebook and implemented an interactive storytelling scenario, using full-body tracking with Kinect to provide different types of interaction. In their scenario, two users listen to a virtual avatar narrating parts of a gamebook. At specific points, the users need to perform certain gestures to influence the story. Almost none of the Kinect games developed so far concentrate on a story, and this may be an interesting approach for the creation of new games. Another interesting idea is the one presented by M. Caon et al. [33] for the creation of smart environments. Using multiple Kinects, these researchers developed a system capable of recognizing with great precision the direction to where a user is pointing in a room. Their "smart living room" is composed of several "smart objects", like a media center, several lamps, a fan, etc., and it's able to identify several users' postures and pointing gestures. If a user points to any of those objects, they will automatically change their state (On/Off). A lot of work can be done to improve home automation based on this idea (and using Kinect).

In almost all of the papers presented before, researchers have used both the RGB and depth sensors to track the human body or objects. At the time this literature review was conducted, we weren't able to find any relevant paper where researchers have studied how well Kinect's RGB camera can recognize and track colors and/or objects changes in size under different lighting conditions, especially on smart textiles. Searching online, we can find some videos of developers/users demonstrating Kinect applications where color recognition and tracking seem to be the main objective. However, this is clearly not enough to report on or to formulate an informed opinion. This information would be valuable, as these are very important features for the successful achievement of the purposed goals.

We believe that given all the possibilities Kinect offers, much more research will be done based on it in the next years. One fact that supports this idea was the release of the Kinect for Windows in February 2012. Kinect for Windows consists of an improved Kinect device and a new SDK version specially designed for Windows PCs. This will allow the development of new kinds of applications, and, consequently, new tools will become available to researchers to perform new studies

To develop the proposed software application, several choices in terms of SDKs, frameworks, libraries, etc., will have to be made, having into account the fact that the application will have to make use of the Kinect Sensor. In the next subsection, we will present and discuss some of the possible choices available.

3.1.1 KINECT SOFTWARE DEVELOPMENT TOOLS

There are several tools available that can be used to develop software using Kinect. Next, we will briefly describe three of the most important software development tools currently used.

3.1.1.1 Microsoft Kinect SDK

The Microsoft Kinect SDK is the official Kinect SDK developed and released by Microsoft. Both the non-commercial and commercial versions were released in 2011. This SDK allows programmers to use the Kinect sensor on computers running Windows 7. It also allows programmers to develop applications for Kinect using Visual Studio 2010 IDE and C++, C# or VB.NET languages.

Microsoft Kinect SDK provides, among others, three important features. The first one is the access to raw data streams from the depth and color camera sensors, and the microphone array. The second enables skeleton tracking, which is able to track up to two people moving in the Kinects sensor's field of view. Finally, the advanced audio processing capabilities, using the microphone array.

In February 2012 Kinect for Windows SDK was released. According to the Microsoft Kinect website [34], this new SDK "offers improved skeleton tracking, enhanced speech recognition, modified API, and the ability to support up to four Kinect for Windows sensors plugged into one computer".

3.1.1.2 Open NI Framework

OpenNI is a non-profit organization that developed a multi-language, cross-platform and open-source framework, which provides APIs for creating applications using natural interfaces (NI).

OpenNI framework provides the interfaces for both physical and middleware components. The physical components currently supported are: 3D sensor, RGB camera, IR camera, and audio device. The middleware components currently supported are: full body analysis (component that generates body related information), hand point analysis (component that generates the location of a hand point), gesture detection (component that identifies predefined gestures and alerts the application) and scene analyzer (component that analysis the image of the scene to produce the separation between the foreground and background of the scene's data, the coordinates of the floor plane data and the individual identification of figures in the scene data).

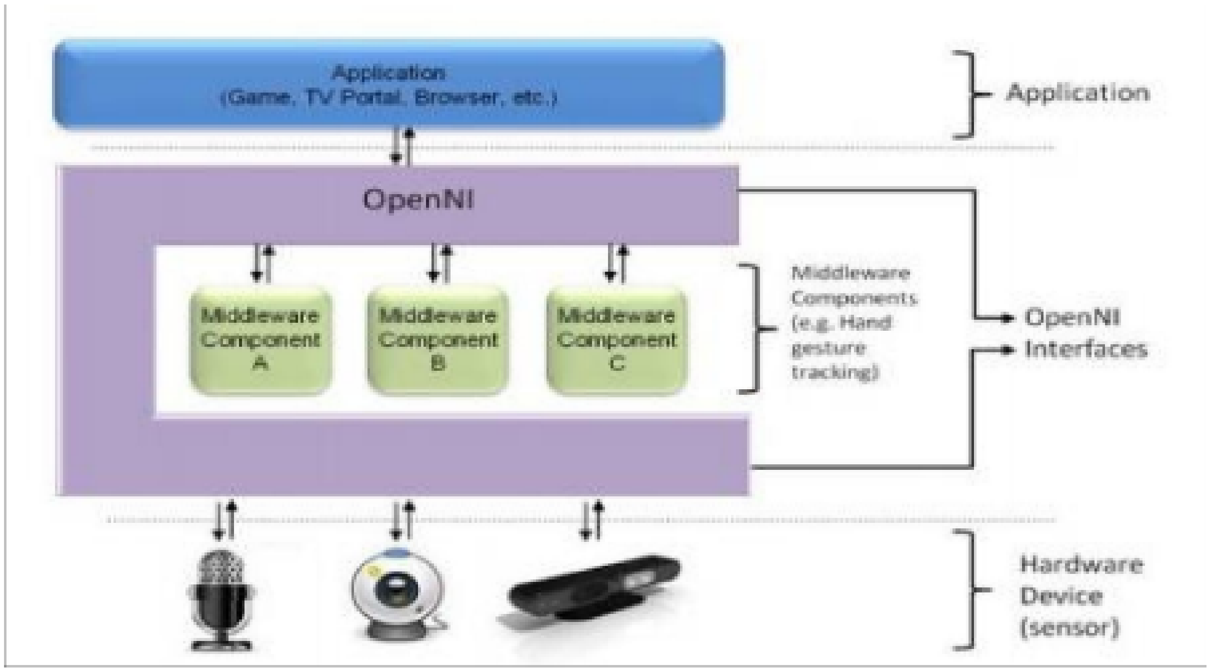


Fig 3.5 Abstract Layered View of the OpenNI concept

Figure 3.5 shows the three-layered view of the OpenNI concept. The top layer represents the software that implements NI applications. The middle layer represents the OpenNI API providing interfaces that interact with both sensors and the middleware components. The bottom layer represents the hardware devices that capture the scene's audio and visual elements.

Production Nodes – "set of components that have a productive role in the data creation process required for Natural Interaction based applications" – are the fundamental elements of the OpenNI interface. There are two types of production nodes, sensor-related and middleware-related production nodes. OpenNI currently supports the following sensorrelated production nodes: device (node that represents a physical device and enables the device configuration); depth generator (node that generates a depth-map); image generator (node that generates colored image-maps); IR generator (node that generates IR imagemaps); and audio generator (node that generates an audio stream). OpenNI also currently supports the following middleware-related production nodes: gestures alert generator (node that generates callbacks to the application when specific gestures are identified); scene analyzer (node that analysis a scene, identifying figures and detecting the floor plane); hand point generator (node that supports hand detection and tracking and generates a callback that provides alerts when a hand point is detected); and user generator (node that generates a representation of a body in the 3D scene). Recorder (node that implements data recordings), Player (node that reads data from a recording and plays it) and Codec (node used to compress and decompress data in recordings) are also supported by OpenNI for recording purposes.

3.1.2. MATLAB

The MATLAB platform is optimized for solving engineering and scientific problems. The matrix-based MATLAB language is the world's most natural way to express computational mathematics. Built-in graphics make it easy to visualize and gain insights from data. A vast library of prebuilt toolboxes lets you get started right away with algorithms essential to your domain. The desktop environment invites experimentation, exploration, and discovery. These MATLAB tools and capabilities are all rigorously tested and designed to work together.

Image Processing Toolbox: Image segmentation using active contours, C code generation for 10 functions, and GPU acceleration for 11 functions.

Image Acquisition Toolbox: Kinect® for Windows® sensor support for acquiring images, depth maps, and skeleton data

System Requirements for MATLAB 2013a

OS: Windows 10

PROCESSOR: Any Intel or AMD x86-64 processor

DISK SPACE: 2 GB for MATLAB only, 4–6 GB for a typical installation

RAM: With Simulink, 4 GB is required

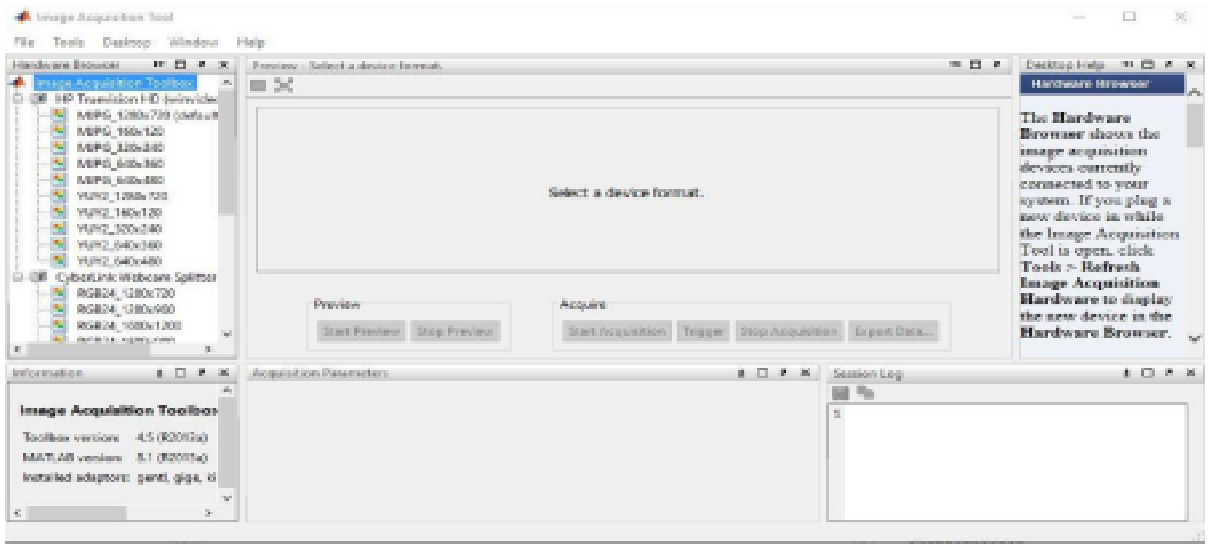
GRAPHICS CARD: No specific graphics card is required, Hardware accelerated graphics card supporting OpenGL 3.3 with 1GB GPU memory is recommended.

3.1.3 IMAGE PROCESSING AND ACQUISITION TOOLBOX

Image Processing Toolbox in MATLAB provides a comprehensive set of reference-standard algorithms and workflow apps for image processing, analysis, visualization, and algorithm development. You can perform image segmentation, image enhancement, noise reduction, geometric transformations, image registration, and 3D image processing.
Image Processing Toolbox apps let you automate common image processing workflows. You can interactively segment image data, compare image registration techniques, and batch-process large datasets. Visualization functions and apps let you explore images, 3D volumes, and videos; adjust contrast; create histograms; and manipulate regions of interest (ROIs).

Image Acquisition Toolbox provides functions and blocks that enable you to connect industrial and scientific cameras to MATLAB® and Simulink®. It includes a MATLAB app that lets you interactively detect and configure hardware properties. The toolbox enables acquisition modes such as processing in-the-loop, hardware triggering, background acquisition, and synchronizing acquisition across multiple devices.

Image Acquisition Toolbox supports all major standards and hardware vendors, including USB3 Vision, GigE Vision, and GenICam, GenTL. You can connect to 3D depth cameras, machine vision cameras, and frame grabbers, as well as high-end scientific and industrial devices.



Select a device format.

The Hardware Browser shows the image acquisition devices currently connected to your system. If you plug a new device in while the Image Acquisition Tool is open, click **Tools > Refresh Image Acquisition Hardware** to display the new device in the **Hardware Browser.**

Image Acquisition Toolbox

Toolbox version: 4.5 (R2013a)
MATLAB version: 8.1 (R2013a)
Installed adaptors: gentl, gige, ...

Fig 3.6 Image Acquisition Tool

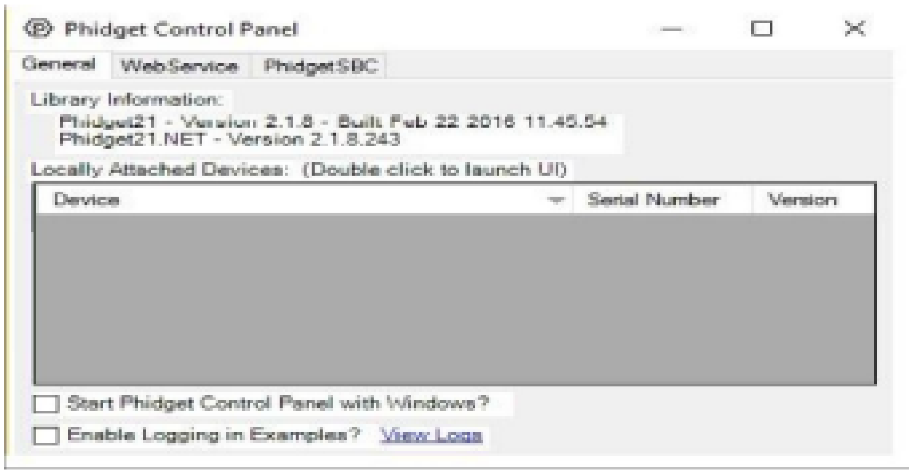
3.2 PHIDGET MOTOR CONTROL (1064_1)

The PhidgetMotorControl HC allows you to control the angular velocity and acceleration of up to two high-current DC motors.

The 1064:

- ∑ Provides bi-directional control to 2 DC motors
- ∑ Requires external 6 to 15VDC Power Supply
- ∑ The USB port is isolated from the motor control outputs
- ∑ Over voltage, over-temperature, over current conditions are fed back to the API on the PC

Many variations of brushed DC motors exist: permanent magnet motors, electromagnet motors, coreless motors, linear motors... the PhidgetMotorController can be used with any of these, as well as other devices like small solenoids, incandescent light bulbs, and hydraulic or pneumatic devices like small pumps and valves.


Phidget Control Panel

General | WebService | PhidgetSBC

Library Information:
Phidget21 - Version 2.1.8 - Built Feb 22 2016 11:45:54
Phidget21.NET - Version 2.1.8.243

Locally Attached Devices: (Double click to launch UI)

Device	Serial Number	Version

Start Phidget Control Panel with Windows?

Enable Logging in Examples? View Logs

Fig 3.7 Phidget API

3.2.1 Controller Properties

API Object Name	MotorControl
Motor Type	DC Motor
Number of Motor Ports	2
Acceleration Time Min	52 ms
Acceleration Time Max	105 s
PWM Frequency Max	20 kHz

3.2.2 Electrical Properties

Supply Voltage Min	6 V DC
Supply Voltage Max	15 V DC
Continuous Motor Current Max	(per motor) 14 A
Current Consumption Max	100 mA
Current Consumption Min	15 mA

USB Speed	Full Speed

CHAPTER 4

ASSEMBLY AND PROCESSING PROCEDURES

Given below is the workflow of the human tracking using Kinect and Matlab. Initially the Kinect has to be connected with the Corobot and the hosting computer using the Kinect drivers. After connecting the Kinect, image acquisition, skeletal tracking, and error calculation is done with the help of Matlab 2013a. Now based on the position of the human the speed of the robot's wheels has to be controlled. This is done using the Phidget motor controller. The phidget

motor controller is continuously fed with values calculated by the program.

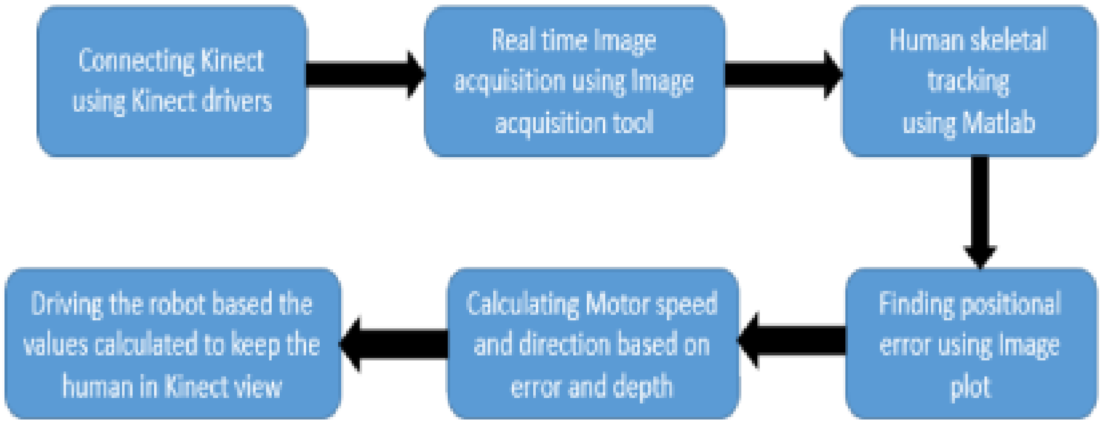

Connecting Kinect using Kinect drivers
Real time Image acquisition using Image acquisition tool
Human skeletal tracking using Matlab
Finding positional error using Image plot
Calculating Motor speed and direction based on error and depth
Driving the robot based the values calculated to keep the human in Kinect view

Fig 4.1 project workflow

4.1 CONNECTING KINECT USING KINECT DRIVERS:

Installing Kinect drivers is very vital for using Kinect in windows OS. Kinect driver serves as interface between the operating system and the robot. Given below are the necessary drivers required to be installed for connecting Xbox 360 Kinect to windows 10

- Kinect for Windows drivers v1.6
- Kinect for Windows SDK v1.6
- Kinect for Windows Runtime v1.6
- Kinect for Windows developer toolkit v1.6

Any of the three Kinect software development tools described before have the necessary tools needed for the development of our application. However, we think that Microsoft Kinect SDK has some advantages when compared to the other two. First of all, it is the official SDK, which means that all features were appropriately tested and should work properly. One other big advantage is that the documentation of Kinect SDK appears to be much better than the documentation of the other two. Being a Microsoft product, it is fairly easy to start developing applications that make use of the Kinect and Kinect SDK on Visual Studio.

In the perspective of the application that we will develop, the main disadvantage of the Microsoft Kinect SDK, when compared to the other two, is that applications developed using that Kinect SDK only runs on Windows operating systems, unlike the other two.

Despite the fact that the use of any of the tools would be possible for the development of our application, we can then conclude that Microsoft Kinect SDK will probably be our best choice in terms of Kinect-based applications development tool.

Knowing that a Kinect sensor is going to be used in this project and that the tool that most likely will be used to develop the Kinect application will be the Microsoft Kinect SDK, we now need to consider which IDE will be more suitable for our development.

4.2 REAL TIME IMAGE ACQUISITION AND SENSOR DATA EXTRACTION:

The IR sensor and RGB camera are the only sensors whose input is required for the process of human tracking. The sensor trigger count and shutter time for the video are given by the user these features can be modified according to the precision required for the bot. The Kinect is then fed these trigger times and other parameters for it to perform the image acquisition. The information that is obtained is a video of the specified frames per second from the RGB sensor and a grey shaded video image depicting the change in the depth of the object with the contract in hue and color saturation of the video from the IR sensor. The in-house microphone multi-array is not used for this application.

4.3 SKELETON RECOGNITION AND IMAGE PROCESSING

The Microsoft APIs provide efficient processing of the sensor data and provide crucial skeletal information. The image obtained is fed as a parameter with an API cal which spots the skeletal structure of the human subject in the image. This data is processed by spotting a 20 pointed skeleton on the image and mapping it to a human being. The 20 points are indexed and given as a dedicated data structure. This data structure can be used to study the skeleton and find the position of a subject in the three-dimensional space in front of the kinect sensor. These indices are particular for a particular body location in the skeleton, eg: the first index of the data structure represents the hip location of the human skeleton which provides the distinctive information of the subject position, the second index is that of the head of the human skeleton and so on. The image is displayed on the hosting computer for understanding the robot's vision.

4.4 ERROR CALCULATION

The displayed image is of a known and user set aspect ratio, as a result, the center point can be obtained and hard coded in the program. This point is the position where the subject is expected to be kept for monitoring purposes. The deviation of the hip point that is the first index of the skeleton data structure from the center indicates the movement of the human. This is calculated as an error, which should be maintained as an error in order to keep the subject in front of the bot. This error is maintained between a particular range so that there is provided a minimum allowance of movement of the human which would provide smooth and seamless movement of the bot. The head position is also noted for understanding the magnification of the subject. The closer the head position is to the top of the image in the screen implies the closer is the subject to the Kinect, the further the head position from the top of the image implies the further the subject is from the Kinect. This error thus provides the information of how far the subject is from the bot and can be made to rest between a particular position from the bot.

4.5 COROBOT SETUP AND DESIGN:

The Corobot is assembled as per the instruction manual excluding the additional components such as the LIDAR, etc in order to keep the system as lightweight as possible. The Microsoft XBOX Kinect sensor is loaded on the Corobot on the second level of the system, at the front in order to obtain the best performance. The Kinect is wired to the socket extension of the ITX computer motherboard. The topmost level of the Corobot provides the ideal real estate for placing the hosting computer whose USB port delivers the connection to the USB socket in the ITX computer of the Corobot. This established connection provides the crucial Phidget bonding of the two systems. All additional components are removed so as to reduce the weight and keep it as low as possible.

4.6 COROBOT MOVEMENT:

The Corobot is made to move according to the error obtained from the sensor. The chief requirement for the bot movement is powering the two motors connected to the two wheels. If the hip positional error obtained is negative indicates the subject position in the left of the machine center, implying the left movement of the bot in order to position the bot in line with the subject. The positive value implies the movement of the bot has to be in the right direction. This rotational movement is obtained by the Skid-Steer drive mechanism, by feeding current of opposite polarity to the different wheels thereby producing torque and giving a rotational motion. This motion signal is fed until the subject is centered with the bot. Once the lateral error is corrected the magnification error is obtained. If this error is closer to two zero implies the subject is closer to the bot and requires the bot to move back to position the subject at an optimum distance. The higher the error implies further the subject position. For the forward and backward movement wheels are fed current of the same polarity to the wheels. The current that is fed is regulated to the wheel motor by the motor drivers which are managed by the mini Itx computer.

4.7 TESTING

The motors and the sensors are to be tested initially to ensure ideal output from the bot. The motor testing is carried out by the Phidget system that integrates the hosting computer and the Corobot. The Phidget Utility interface provides information of all the physical entities of the bot and the required values of inputs that can be provided thereby making the testing

process easier. The input values can be fed from this utility interface using slider bars associated with the names of each motor it has scanned. Changing these values provides movement of the wheels and accuracy can also be found in this procedure.

For the Kinect testing, a test program is run in order to verify if the images obtained are at the specified trigger rates and if the quality is as required. The skeletal structure formed are also verified

CHAPTER 5

MATLAB FUNCTIONS FOR HUMAN TRACKING

5.1 FUNCTIONS

Human tracking here is done using MATLAB programs (APPENDIX A). Below are some of the functions used for image acquisition, skeletal tracking and error calculation explained in detail. The program used to achieve human tracking is provided in APPENDIX A for reference.

Imaqreset

Disconnect and delete all image acquisition objects

Syntax

Imaqreset

Description

imaqreset deletes any image acquisition objects that exist in memory and unloads all adaptors loaded by the toolbox. As a result, the image acquisition hardware is reset. imaqreset is the image acquisition command that returns MATLAB® to the known state of having no image acquisition objects and no loaded image acquisition adaptors.

You can use imaqreset to force the toolbox to search for new hardware that might have been installed while MATLAB was running.

Note that imaqreset should not be called from any of the callbacks of a video input object, such as the StartFcn or FramesAcquiredFcn.

Acquire Color and Depth Data

In order to acquire synchronized color and depth data, we must use manual triggering instead of immediate triggering. The default immediate triggering suffers from a lag between streams while performing synchronized acquisition. This is due to the overhead in starting streams sequentially.

```
% Create the VIDEOINPUT objects for the two streams
colorVid = videoinput('kinect',1)
Summary of Video Input Object Using 'Kinect Color Sensor'.
   Acquisition Source(s): Color Source is available.
  Acquisition Parameters: 'Color Source' is the current selected source.
                          10 frames per trigger using the selected source.
                          'RGB_640x480' video data to be logged upon START.
                          Grabbing first of every 1 frame(s).
                          Log data to 'memory' on trigger.

      Trigger Parameters: 1 'immediate' trigger(s) on START.
                  Status: Waiting for START.
                          0 frames acquired since starting.
                          0 frames available for GETDATA.
depthVid = videoinput('kinect',2)
Summary of Video Input Object Using 'Kinect Depth Sensor'.
   Acquisition Source(s): Depth Source is available.
  Acquisition Parameters: 'Depth Source' is the current selected source.
                          10 frames per trigger using the selected source.
                          'Depth_640x480' video data to be logged upon START.
                          Grabbing first of every 1 frame(s).
                          Log data to 'memory' on trigger.

      Trigger Parameters: 1 'immediate' trigger(s) on START.
                  Status: Waiting for START.
                          0 frames acquired since starting.
                          0 frames available for GETDATA.
```

```
% Set the triggering mode to 'manual'
triggerconfig([colorVid depthVid],'manual');
```

Set the FramesPerTrigger property of the VIDEOINPUT objects to '100' to acquire 100 frames per trigger. In this example, 100 frames are acquired to give the Kinect for Windows sensor sufficient time to start tracking a skeleton.

```
colorVid.FramesPerTrigger = 100;
depthVid.FramesPerTrigger = 100;
% Start the color and depth device. This begins acquisition, but does
not % start logging of acquired data.
start([colorVid depthVid]);
% Trigger the devices to start logging data.
```

```
trigger([colorVid depthVid]);
% Retrieve the acquired data
[colorFrameData,colorTimeData,colorMetaData] = getdata(colorVid);
[depthFrameData,depthTimeData,depthMetaData] =
getdata(depthVid); % Stop the devices
stop([colorVid depthVid]);
```

Configure Skeletal Tracking

The Kinect for Windows sensor provides different modes to track skeletons. These modes can be accessed and configured from the VIDEOSOURCE object of the depth device. Let's see how to enable skeleton tracking.

```
% Get the VIDEOSOURCE object from the depth device's VIDEOINPUT
object. depthSrc = getselectedsource(depthVid)
```

```
Display Summary for Video Source Object:
   General Settings:
    Parent = [1x1 videoinput]
    Selected = on
    SourceName = Depth Source
    Tag =
    Type = videosource

Device Specific Properties:
   ∑ Accelerometer = [-0.008547 -0.98046 -0.11966]
```

∑ BodyPosture = Standing

∑ CameraElevationAngle = 9

∑ DepthMode = Default

∑ FrameRate = 30

∑ IREmitter = on

∑ SkeletonsToTrack = [1x0 double]

∑ TrackingMode = Off

The properties on the depth source object that control the skeletal tracking features are TrackingMode, SkeletonToTrack and BodyPosture properties on the VIDEOSOURCE. TrackingMode controls whether or not skeletal tracking is enabled and, if it is enabled, whether all joints are tracked, 'Skeleton', or if just the hip position is tracked, 'Position'. Setting TrackingMode to 'off' (default) disables all tracking and reduces the CPU load.

The 'BodyPosture' property determines how many joints are tracked. If 'BodyPosture' is set to 'Standing', twenty joints are tracked. If it is set to 'Seated', then ten joints are tracked.

The SkeletonToTrack property can be used to selectively track one or two skeletons using the 'SkeletonTrackingID'. The currently valid values for 'SkeletonTrackingID' are returned as a part of the metadata of the depth device.

```
% Turn on skeletal tracking.
depthSrc.TrackingMode = 'Skeleton';
```

Access Skeletal Data

The skeleton data that the Kinect for Windows produces is accessible from the depth device as a part of the metadata returned by GETDATA. The Kinect for Windows can track the position of up to six people in view and can actively track the joint locations of two of the six skeletons. It also supports two modes of tracking people based on whether they are standing or seated. In standing mode, the full 20 joint locations are tracked and returned; in seated mode, the 10 upper body joints are returned. For more details on skeletal data, see the MATLAB documentation on Kinect for Windows adaptor.

```
% Acquire 100 frames with tracking turned on.
% Remember to have a person in person in front of the
% Kinect for Windows to see valid tracking data.
colorVid.FramesPerTrigger = 100;
depthVid.FramesPerTrigger = 100;
start([colorVid depthVid]);
trigger([colorVid depthVid]);
% Retrieve the frames and check if any Skeletons are tracked
[frameDataColor] = getdata(colorVid);
[frameDataDepth, timeDataDepth, metaDataDepth] =
getdata(depthVid); % View skeletal data from depth metadata
```

metaDataDepth

```
metaDataDepth =
100x1 struct array with fields:
```

- ∑ AbsTime
- ∑ FrameNumber
- ∑ IsPositionTracked
- ∑ IsSkeletonTracked
- ∑ JointDepthIndices
- ∑ JointImageIndices
- ∑ JointTrackingState
- ∑ JointWorldCoordinates
- ∑ PositionDepthIndices
- ∑ PositionImageIndices
- ∑ PositionWorldCoordinates
- ∑ RelativeFrame
- ∑ SegmentationData
- ∑ SkeletonTrackingID
- ∑ TriggerIndex

We randomly choose the 95th frame to visualize the image and skeleton data.

```
% Check for tracked skeletons from depth metadata

anyPositionsTracked = any(metaDataDepth(95).IsPositionTracked ~= 0)

anySkeletonsTracked = any(metaDataDepth(95).IsSkeletonTracked ~= 0)

anyPositionsTracked =

    1

anySkeletonsTracked =

    1
```

The results above show that at least one skeleton is being tracked. If tracking is enabled but no IDs are specified with the TrackingID property, the Kinect for Windows software automatically chooses up to two skeletons to track. Use the IsSkeletonTracked metadata to determine which skeletons are being tracked.

```
% See which skeletons were tracked.
trackedSkeletons = find(metaDataDepth(95).IsSkeletonTracked)
trackedSkeletons =
    1
```

Display skeleton's joint coordinates. Note that if the 'BodyPosture' property is set to 'Seated', the **'JointCoordinates'** and **'JointIndices'** will still have a length of 20, but indices 2-11(up per-body joints) alone will be populated.

```
jointCoordinates = metaDataDepth(95).JointWorldCoordinates(:, :, trackedSkeletons)
% Skeleton's joint indices with respect to the color image
jointIndices = metaDataDepth(95).JointImageIndices(:, :, trackedSkeletons)
jointIndices =
  318 256
  317 240
  318 143
  319 92
  271 177
  239 243
  219 303
```

```
216 323
```

```
363 177
399 243
421 303
424 322
296 277
286 387
288 492
286 520
340 277
342 384
347 493
350 522
```

Draw the Skeleton Over the Corresponding Color

Image % Pull out the 95th color frame

```
image = frameDataColor(:, :, :, 95);
% Find number of Skeletons tracked
nSkeleton = length(trackedSkeletons);
% Plot the skeleton
util_skeletonViewer(jointIndices, image, nSkeleton);
```

5.2 PHIDGET CODE

5.2.1 Open Motor

```
loadphidget21;
mhandle = libpointer('int32Ptr');
calllib('phidget21', 'CPhidgetMotorControl_create', mhandle);
calllib('phidget21', 'CPhidget_open', mhandle, -1);
```

You can write your MATLAB code the same way you usually do, whether within a text editor or within MATLAB itself. However, to use Phidget calls within your code, you must include a reference to the Phidget library in your main body of code, before any Phidget calls:

```
function phidgettest(n)
    loadphidget21;
    % More code goes here
  end
```

loadphidget21 is a .m file we have included that will check your OS and reference the proper phidget21Matlab.h file (there is a different one for each OS). Also, before running your code, remember to copy Phidget header files and loadphidget21.m to your project directory

The mhandle is created as a handle for the Phidget. This example is *specific to the Interface*

Kit because the call CPhidgetMotorControl_create is used. For another device, use the corre spondingly named called in the C/C++ API.

The handle **mhandle** is then used for all the C function calls where CPhidgetHandle mhandle is used in the C/C++ API. Every type of Phidget also inherits functionality from the Phidget base class.

Note that open() opens the software object, but not hardware. So, it is not a guarantee you can use the Phidget immediately. The different types of open can be used with parameters to try and get the first device it can find, open based on its serial number, or even open across the network. The API manual lists all of the available modes that open provides.

5.2.2 RUN MOTOR

```
calllib('phidget21', 'CPhidgetMotorControl_setVelocity',mhandle,0,lv);
calllib('phidget21', 'CPhidgetMotorControl_setVelocity',mhandle,1,rv);
```

MATLAB does not support event handling, so all data must be read and sent directly.The most common thing you might want to do is read data from sensors. For example, in the code above, we might want to read data from kinect on the Phidget Interface Kit.

5.2.3 CLOSE MOTOR

```
calllib('phidget21', 'CPhidget_close', mhandle);
calllib('phidget21', 'CPhidget_delete', mhandle);
```

At the end of your program, don’t forget to call close to free any locks on the Phidget that the open() call put in place.

CHAPTER 6

RESULTS AND DISCUSSION

6.1 IMAGE ACQUISITION AND SKELETAL TRACKING:

The image of the human being tracked is acquired by the depth and RGB camera available in the Kinect. The Kinect senses whether the object before it is a human or not. When a human enters the vision of the Kinect, it tracks the skeletal structure of the human. The skeletal structure consists of twenty points which helps us to track the human. Shown below are two images acquired by the Kinect for the purpose of tracking.

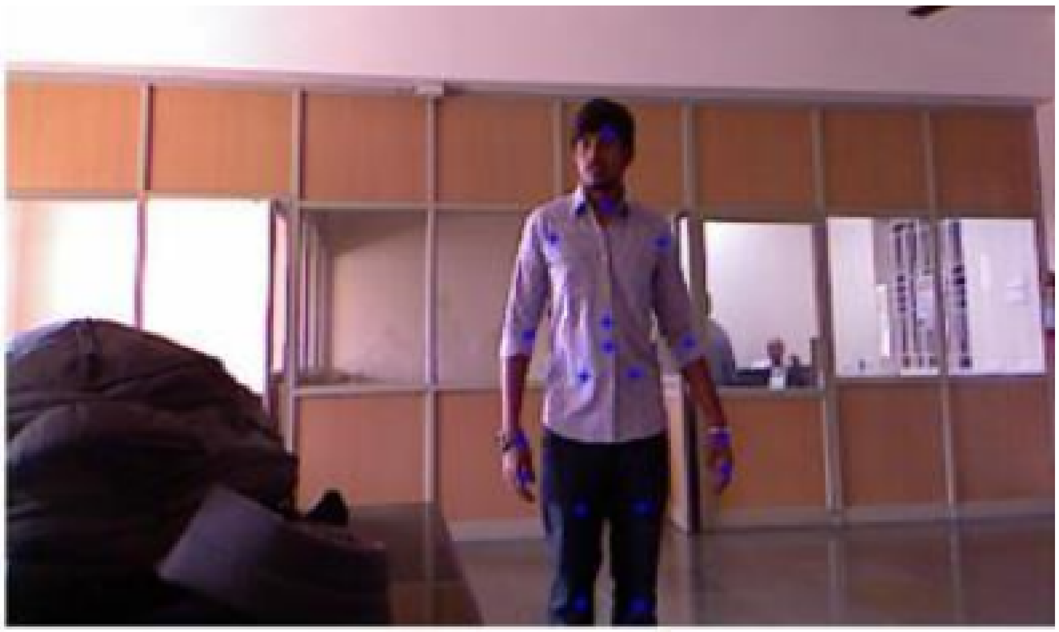


Fig 6.1 Human Tracking-Color Image

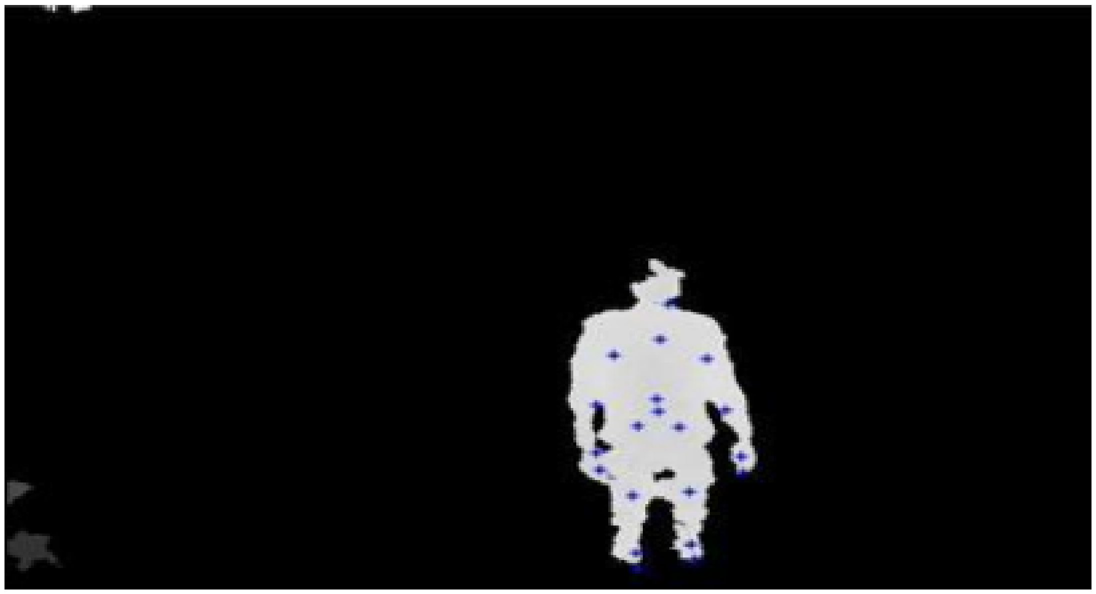
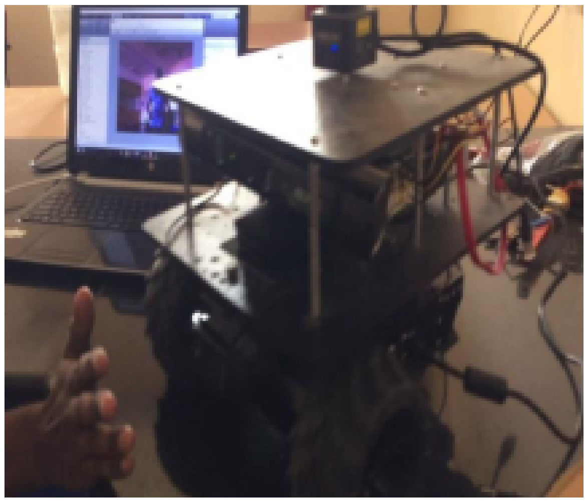


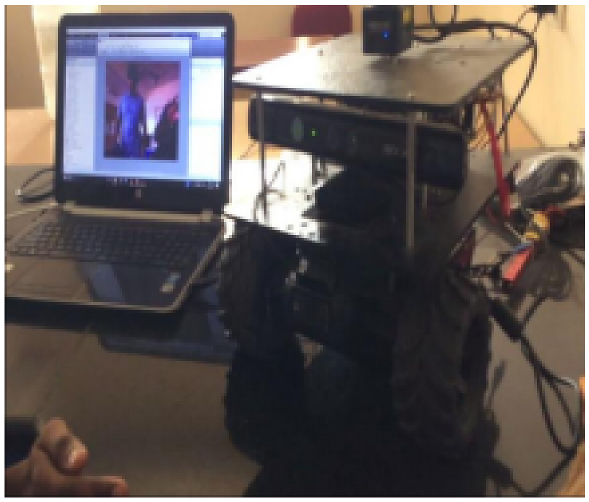
Fig 6.2 Human Tracking- Depth Image

6.2 TRACKING CONTROL AND MOVEMENT

The experiment was conducted indoors without any obstacles and difference in the floor level.

From the point of safety and keeping the person in the view the robot will be stopped if the distance between the bot and the human becomes lesser than the intended value. The verification was carried out by conducting various test cases like moving side to side, front and back, in a zigzag manner, etc.





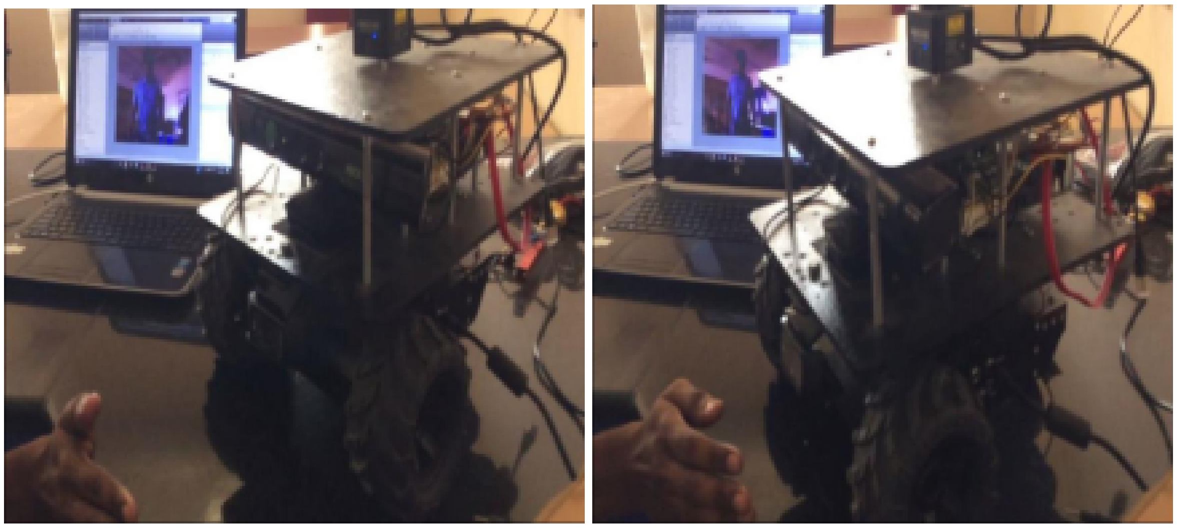


Fig 6.3 Tracking with corobot

These sequences of images are taken while testing the tracking control in the case where humans moved side to side and front and back. The robot calculates the center point of the screen and the position of the hip point of the human skeleton. The difference between these two points is used to calculate whether the robot has to turn right or left. If the hip point is within the middle range of the screen (here -40 to 40) the robot remains stationary. If the hip point is moved to the left range of the screen (here < -40) the robot is also made to turn in the tracking human's

direction by using phidget motor controller until the hip point comes within the middle range. This can be done by powering the right wheel of the robot in the forward direction and the left wheel in the reverse direction. If the hip point is moved to the right range of the screen (here >40) the robot is also made to turn in the tracking human's direction by using a phidget motor controller until the hip point comes within the middle range. This can be done by powering the left wheel of the robot in the forward direction and the right wheel in the reverse direction.

When the skeleton is within the middle range of the robot's vision. It starts checking the distance between the human and itself and calculates whether it should move forward or backward. The distance between the center point on the top edge and the skull point of the skeleton is used to calculate whether the robot has to move forward or backward. The speed of the robot is proportional to the distance between the robot and the human. If the head point is near to the top center (signifying that the human is very close to the robot), the robot moves backward. If the point is farther from the top center (signifying that the human is very far from the robot), the robot moves forward. The speed of the robot varies based on the range in which the head point lies. From the observed results, it is verified that the proposed method of estimation and traffic control is effective.

CHAPTER 7

CONCLUSION

The Kinect is an efficient device that exposes powerful APIs that can be used to recognize a human being. This procedure offers human tracking in an efficient and accurate manner which can be used with great ease. Three-dimensional information of subject for tracking provides us the ease of understanding the location of the subject in the space before the bot and the skeleton structure obtained provides us a behavioral pattern of the subject which can help in studying the subject. From the results obtained, the following conclusions were made:

- ∑ The skeleton structure is obtained with a low time lag and also gives accurate results with perfect joint placements.
- ∑ The Corobot response time is quick and precise with smooth overall movement.
- ∑ The processing time taken on the hosting computer is high, the higher versions of MATLAB can be used to overcome this disadvantage.

∑ It is found that the battery consumption is minimal and the usage is kept low by the system.

∑ Quick human movement results in improper final bot movement which can be overcome by using superior versions of MATLAB.

Finally, it can be concluded that an efficient and powerful human tracking robot can be built using the Kinect 3D sensor.

SCOPE FOR FUTURE WORK

The human tracking robot can be used for a number of fields as listed below:

∑ By studying the given environment beforehand, the bot can be made to camouflage itself escaping from human vision while tracking

∑ Enhancing code to monitor more than one human in case of Hotspot Surveillance

∑ Integrating facial recognition and flight components can provide an ideal platform for remote spying and surveillance

REFERENCES

[1] Eiji Machida, Meifen Cao, Toshiyuki Murao, Hiroshi Hashimoto, "Human motion tracking of mobile robot with Kinect 3D sensor, SICE Annual Conference (SICE), 2012

[2] Microsoft Research Kinect for Windows SDK Programming Guide

[3] Kinect Image Acquisition, https://in.mathworks.com/help/imaq/examples/using-the-ki

nect-r-for-windows-r-from-image-acquisition-toolbox-tm.html

[4] Skeletal View, http://in.mathworks.com/help/imaq/using-the-skeleton-viewer-for-kinect skeletal-data.html

[5] Image acquiring, https://in.mathworks.com/help/imaq/acquiring-image-and-skeletal data-using-thekinect.html

[6] Kinect SDK,

https://developer.microsoft.com/en-us/windows/kinect [7] COROBOT

Specifications and working- http://www.coroware.com/

[8] J. Satake and J.Miura, "Person Following of a Mobile Robot using Stereo Vision",

Journal of the Robotics Society of Japan, Vol. 28, no.9, 2010