Neural Network Based Gesture Recognition Robot

by

Bowen Yuan

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
December 16, 2017

Keywords: Artificial Neural Network, Gesture Recognition, ROS, Autonomous Mobile Robot

Abstract

Autonomous mobile robots (ARM's) are defined by their ability to perform some tasks independently from direct human interaction. However, interaction with humans at some point is necessary. There are some situations in which humans can give instructions to robots most effectively with physical gestures. Example include scenarios that require silence, such as covert military operations or hospital rooms, loud assemblies, underwater, or in space.

Artificial neural networks (ANN's) have long been used for pattern recognition. If well-trained with a suitable dataset, an ANN can provide a satisfactory result for a complex task.

ROS (Robot Operating System) is a mainstream software framework for robotics research throughout the world. It can streamline the development of robots through code reuse.

This thesis describes a partial solution to help robots understand human gesture by combining an ANN and ROS, with a Kinect sensor as the primary input. Five gestures are trained, and then recognized when performed by people other than the trainer. The overall success rate of gesture recognition in this study is 80 percent. Some gestures are recognized with more than 90 percent success.

Acknowledgments

I would first and foremost love to give my advisor, Dr. Thaddeus Roppel my deepest appreciation for his support and guidance during my time at Auburn University. Dr. Roppel provided me with the great opportunity doing my project and research in his CRRLAB as a graduate student. He is an advisor to me, but also a professor, a mentor, and a friend. I would also love to express my appreciation to Dr. Xiaowen Gong and Dr. Stanley Reeves for their time and support as a member of my thesis committee.

I would love to show my gratitude to my parents. They provided me generous help with my life and education. I would never be able to reach my current academic level without their help. They are also great mentors for my life: helped me from suffering and confusing. I could never thank them enough.

I would also give my appreciation to my friends. They are part of my life and provided me support in many different areas. They are my treasure in life.

Table of Contents

List of Tables

List of Illustrations

List of Abbreviations

AMR	Autonomous Mobile Robot
ANN	Artificial Neural Network
NN	Neural Network
PCL	Point Cloud Library
ROS	Robot Operating System
CRRLAB	Cooperative Robotics Research Lab
PID	Proportional, Integral, and Derivative Control
RVIZ	Robot Visualization Tool
SLAM	Simultaneous Localization and Mapping
EBP	Error Back-Propagation
GRR	Gesture Recognition Robot

Chapter 1

Motivation

Autonomous mobile robots (AMRs) are emerging in recent decades in many areas including academic, military, and industrial[1][2]. AMRs have abilities to finish tasks that human will never be able to try, such as rescue in dangerous areas and restlessly working in a high-precision job with a rapid pace. AMRs have been recognized as a reliable assistant in modern and future human life. To better serve people, AMRs will benefit from a high-level intelligence that can understand instruction from humans without an external controller – to be more like communicating with creatures.

Much research has been done with the topic of increasing robot intelligence and it is also a fast-growing area in both academia and industry. This thesis will use an artificial neural network (ANN) to help the AMR with understanding human gestures with specific reactions. Gesture is a normal way to communicate between human individuals in daily life. In some kind of situations, gestures will provide better quality such as under water, or in a silent environment like a hospital room. Through this neural network implementation, people will be able to communicate with robots in a more natural way.

Chapter 2

Robot Operating System (ROS) and Gesture Recognition Review

The robot used for this thesis work was built based on a very popular robot control system – the Robot Operating System (ROS)[3]. ROS is an open source, flexible framework for developing robot software. It is a collection of tools, libraries, and conventions that try to streamline the process of creating a robust but complex set of robot behaviors. In this thesis, a Kinect Depth Camera Sensor will collect a a point cloud from the environment and send those data to the ROS environment[4]. After processing those raw data, a human skeletal pose model will be generated and then sent as a topic into the ROS environment. An artificial neural network processing node will attempt to recognize a gesture, which will be published into ROS and subscribed by the control node. Finally, the robot will follow instructions and respond with proper movements. Almost all the information communication inside the robot will go through ROS.

2.1 ROS Overview

Robotics is experiencing tremendous growth. Consequently, a lot of effort has been devoted to enabling code reuse. ROS has become the de-facto standard meta-operating system for robots. It includes support for basic level embedded functions, physical actions, and all the way up to autonomous control[3].

2.2 Basic Implementation Concepts

The fundamental concepts of ROS are nodes, messages, topics, and services. These four concepts are built directly for ROS and get involved with every ROS application[2].

2.2.1 Nodes

Nodes are processes that perform computation/function (more precisely: when the node is running, it is a process. Otherwise it is a bunch of code).

2.2.2 Messages

Nodes use messages to communicate with each other, more specifically, passing messages. A message is a typed data structure, strictly bounded, which may contain almost any data type (Integer, Double, Float, etc.). There are a lot of supported data types built inside the ROS and it is very convenient to use. In the meantime, you can always define your own data structure using a ".msg" file, such as widely used Pose2D.msg file which can provide x and y axis data in one message.

2.2.3 Topics

A node uses message to communicate and it does so through publish or subscribe such message to a topic. Topics are typically identified as different strings such as "/twist" and "/image_raw". A node will receive designated messages by subscribing to a certain topic. It can send a message by publishing messages to a topic in order to let other subscribed nodes receive the messages. A node can have many different publishers and subscribers while many subscribers can listen to one same topic – same as publisher. Actually, publishers and subscribers will not know each other's existence; what they will do is only to send out or receive messages through a specific topic.

2.2.4 Service

Other than topic, service is another communication paradigm. Compared to topic's asynchronous communication, services provide a synchronous way for communication. All the services work under a call-respond rule in which one node requests a specific other node to execute and return a one-time response. This kind of situation will occur when the modified system will perform only one-time task which is both special and unique – this kind of action will not fit the all-time broad-casting architecture, which is the topic-message architecture.

2.3 Gesture Recognition Review

Gestures have long been considered an important way to provide human-robot interaction. A gesture is a natural way for humans to interact, and they are easy to learn. According to research, 35% of human communications consists of verbal but 65% of human communications consists of non-verbal gesture-based communications[5]. Sensors used for gesture vary from optical devices, such as HD cameras: "SoftKinetic HD camera", to wearable devices, such as a data glove: "CyberGlove II" [6][7].

2.3.1 Gesture Categories

Gestures can be divided into two different categories: static or dynamic. A static gesture is represented by a hand performing no change of orientation or position in a certain time interval. A dynamic gesture is represented by hand movements involving other body parts following a certain trajectory within a limited time interval[8].

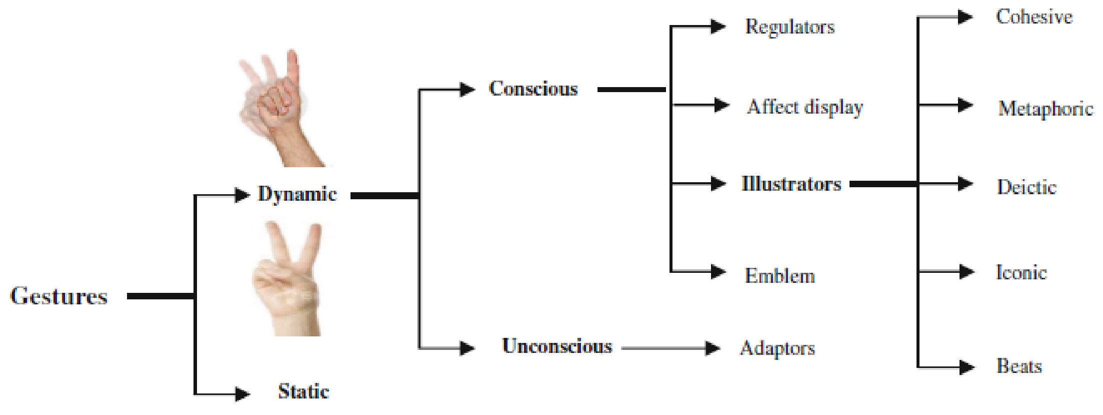


Figure 2.1: Hand Gesture Categories[8]

2.3.2 Gesture Recognition Technology

Gesture recognition technology includes three areas: gesture detection, gesture tracking, and gesture recognition[9]–[11]. There are abundant number of researches done in these areas. In this thesis, the focus is on gesture recognition using existing gesture detection and gesture tracking technology, namely PCL and Skeleton_Marker in ROS.

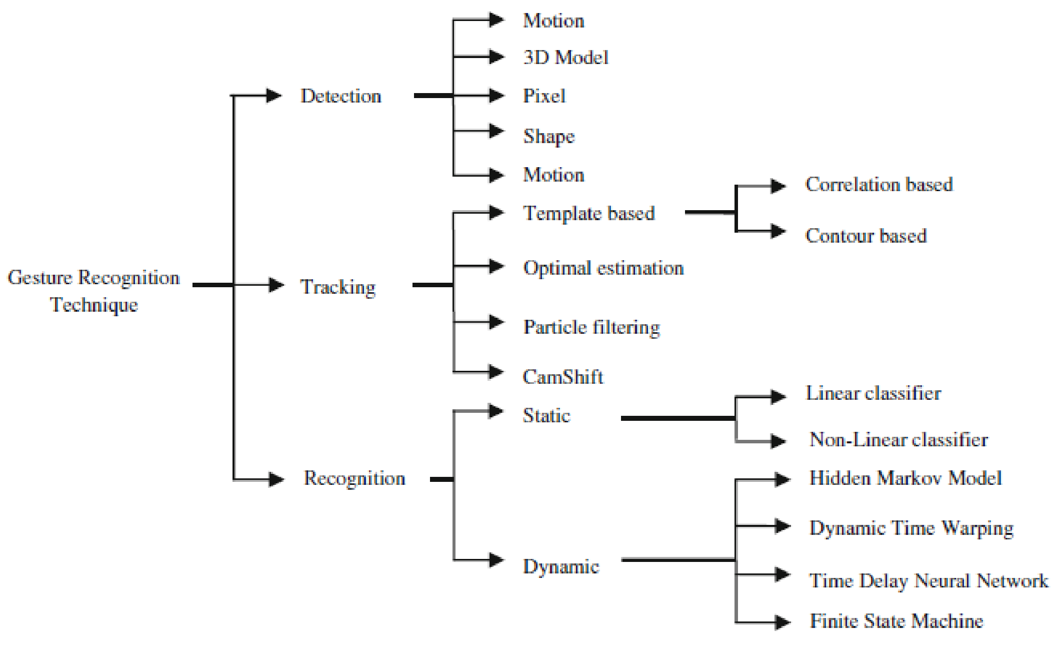


Figure 2.2: Gesture Recognition Technology[11]

2.3.3 Gesture Recognition

The target of gesture recognition is to interpret the semantics of hand(s) location and posture. For static gestures, a common gesture classifier or template-matcher will be used. For dynamic gestures, techniques that can handle temporal dimensional change will be more helpful such as Hidden Markov Models (HMM) – but for dynamic gesture with known class representatives, a supervised learning will perform better. Some of the general gesture recognition techniques are listed below:

Table 2-1 Gesture Recognition Techniques

K-means	A technique which will determine k centers (points) to minimize the clustering error and find statically similar groups in multi-spectral space[12][13]. It can provide a reliable match between gestures and unknown data income.
K-nearest Neighbor	A technique which will classify objects based on the closest training examples[14]. It will also help with the static gestures.
Hidden Markov Model	Firstly introduced in the 1990s, Hidden Markov Model is a technique which will provide a probability value for each state analysis and provide a final probability at the final state – a probability which will determine the user's most likely gesture dynamically[13][14].
Neural Network	A technique which will provide the network a memory about different gesture recognitions and thus obtain the ability for real time dynamic gesture recognition[17]. Since it's a supervised learning, a new network will need to be trained for every new gesture.

2.3.4 Applications

The KNN algorithm has been used to achieve over 90 percent accuracy in static gesture recognition for human-robot interaction. [18]. Another group established a gesture-following teleoperated robot for NASA/DARPA robonaut with exo-skeleton like hardware[19]. In 2012, a senior design group at Auburn University developed a Smart Cart which a reacted to human gesture based on the arm extended ratio – different ratios can represent different gestures[20]. The Smart Cart project is one of the motivations for this article. An easier to use and maintain, high accuracy, and robust system will help more with human daily life. There is little research or project done on gesture recognition using neural networks on an indoor robot.

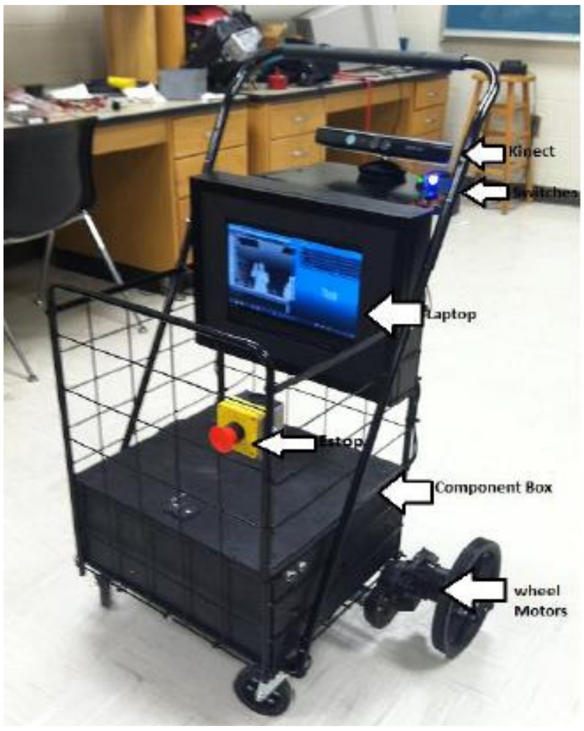


Figure 2.3: AU Smart Cart

Chapter 3

Robot Hardware

A specific autonomous mobile robot was used for the experiments presented in Chapter 7**.** This robot came from Auburn University's CRRLAB which was used for work dealing with cooperative robotics. This robot is fully integrated into the ROS system and contain different kinds of sensors and electronics which will allow it to sense and respond to gesture in a dynamic environment. Since this robot was first integrated by other researchers, we will give a brief introduction about the basic parts and focus more on the different hardware setup between the present application and the original setup.

3.1 Chassis

The chassis displayed in Figure 7.1 is the REX-16D platform from Zagros Robotics[2]. This chassis contains two support wheels, three 14-inch diameter round plastic surfaces for loads and electronics, and two differential drive motors. These three surfaces are stacked up one by one and separated by different spacing– decided by the usage of the specific layer. The lowest surface holds the basic driving system including a dual motor driver, a power distribution board, a battery and an Arduino Mega board. The next layer contains a Kinect depth camera sensor and a battery for it. The top layer holds an Acer netbook running ROS hydro which acts as the control center for all the on-board electronics.

3.2 Power

All the bottom layer electronics are powered by a 12V rechargeable lead acid battery placed on the bottom layer. The Kinect sensor is powered by another 12V rechargeable lead acid battery individually. The 12V provided by the bottom battery will partly go through a PD-101 power distribution board which will be able to provide a stable 5V power for all the embedded electronics.

Sealed Rechargeable Battery
POWER PS SONIC
MODEL PS-1270 F2
12 Volt 7.0 Amp. Hr.
Rechargeable Sealed Lead-Acid Battery
DC power, constant voltage charge
Cycle use : 14.4~15.0V(20°C)
Standby use : 13.5~13.8V(20°C)
Initial current : 2.1A MAX
Warning:
Risk of fire, explosion, or burns.
Do not disassemble, heat above 60°C, or incinerate.
www.power-sonic.com

Figure 3.1: 12V Lead Acid Battery

3.3 Lower Control (Driving System)

The robot is driven by a differential drive system. Two motors were implemented horizontally under the bottom surface. This differential driving system can provide up to 0.5 m/sec speed for the robot. The encoders with the motors can provide over 32000 encoder pulses for one revolution. The encoder signal is processed by a hardware divide-by 16 circuit resulting in 2100 pulses for one revolution of the drive draft. Two castor wheels are mounted on the diagonal of the square whose other two corners are the motors.

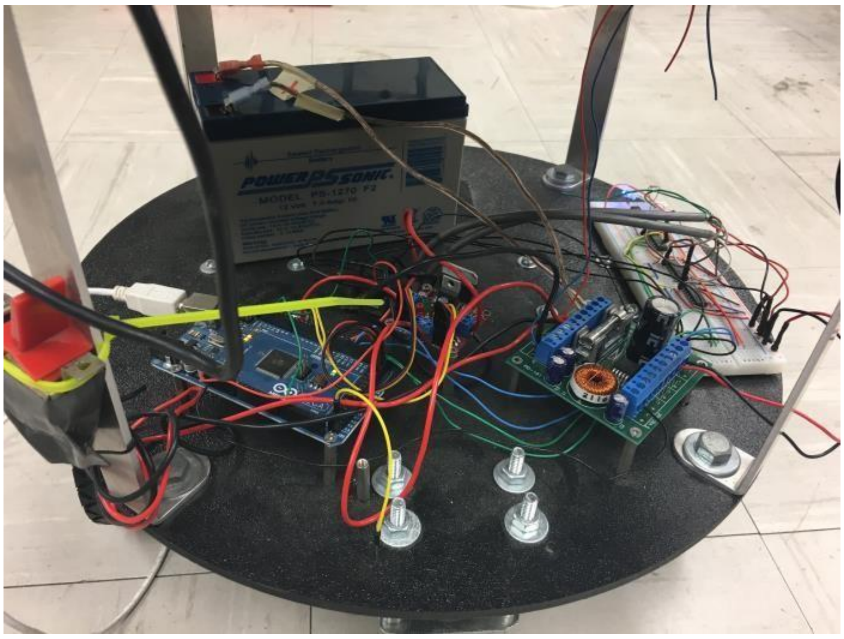


Figure 3.2: Lower Control Parts on Bottom Surface

3.4 Kinect Sensor

The Kinect^{TM} for Xbox 360 Sensor is a motion sensing input device by Microsoft for Xbox 360 and Microsoft Windows PCs. It enables users to interact with their PCs/consoles without a gaming controller through an infrared projector, a camera and a special microchip. It is a widely used sensor because of the availability and the low-cost.

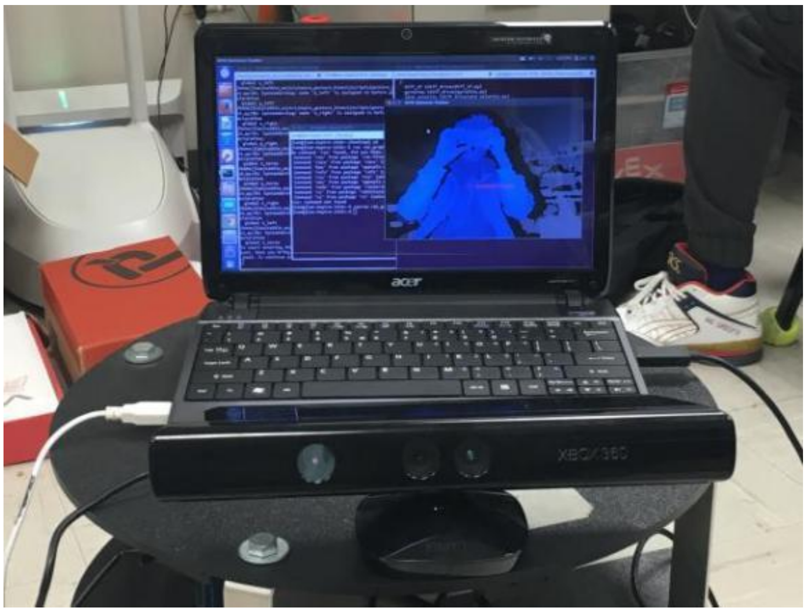
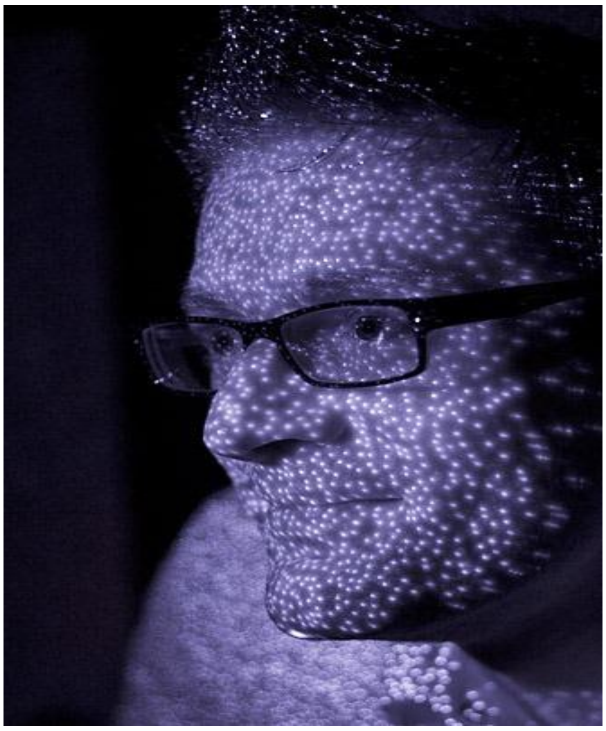



Figure 3.3: Kinect Sensor and Produced Infrared Points

However, it also has some disadvantages which will limit its performance. Since the sensor is based on the infrared signal, it cannot be used outdoor under daylight. The sunlight will block the infrared points and void the sensor, which means our robot can only be used indoor. On the other hand, due to its size, it cannot be mounted on some small robots, which will not happen on the gesture recognition robot. Also, it has a measuring range between 20cm to 3m, which means you can never leave far away from the robot and send gesture instructions to it.

3.5 PID Control System

The motors on the bottom surface are controlled by a pair of PID feedback loops. These PID loops work with an update frequency of 100Hz and will make sure the robot drives at a designate linear and angular velocity. The implementation has been introduced in Brain Pappas's thesis[2].

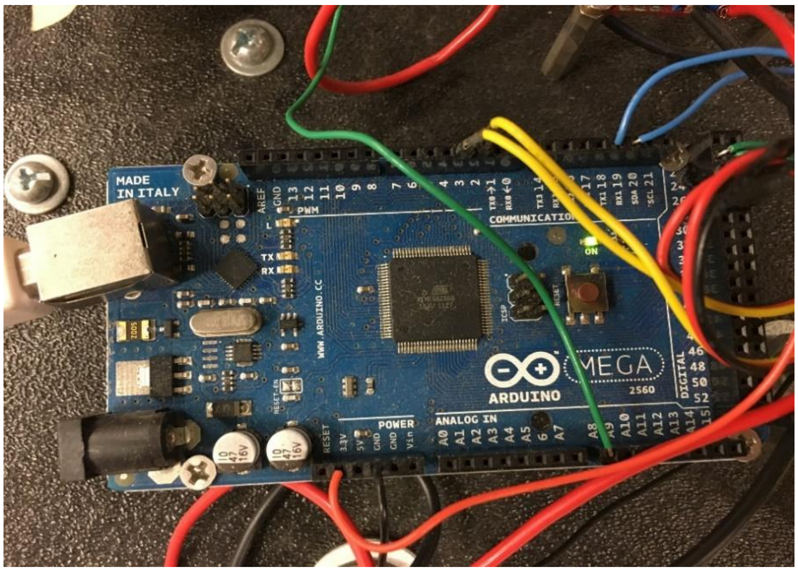


Figure 3.4: Arduino Mega Board

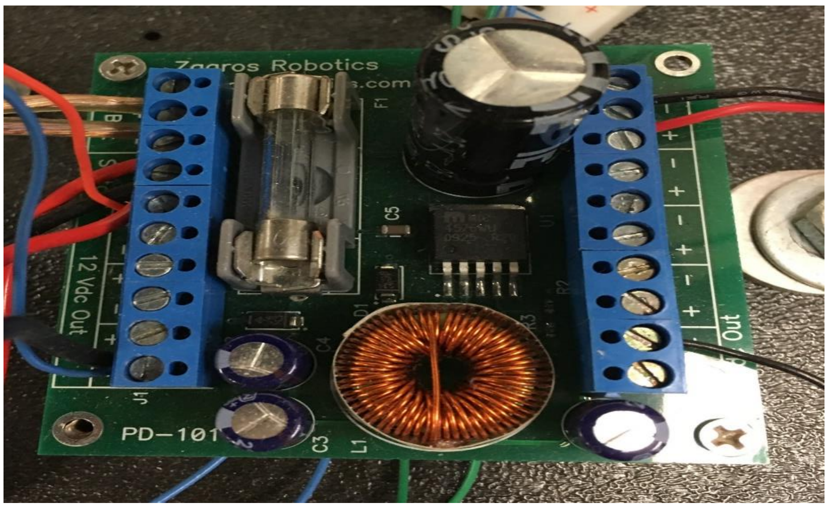


Figure 3.5: Power Distribution Board

Chapter 4

Open Source Software

The robot vision system is based on several open source software or open library, OpenNI and Pybrain respectively. The following contents will focus on their introduction and implementation on this gesture recognition robot.

4.1 OpenNI

OpenNI, as known as Open Natural Interaction, is an industry-led non-profit organization, initially built by PrimeSense -- a company which provides PrimeSense sensor for Kinect[5][6]. Due to Apple’s acquisition, the original OpenNI group was shut down. The new OpenNI website, OpenNI 2, was built based on the forked OpenNI directory and still works as an open source software for further use.

OpenNI can provide a set of open source APIs for accessing natural interactive devices, also known as OpenNI SDK. It can support voice and voice command recognition, hand gestures, and body motion tacking. In this thesis, we will use the already-built openni_kinect package in ROS to deal with the point cloud acquired by the Kinect. It will be briefly introduced in this chapter.

4.1.1 PrimeSense PrimeSensor

To understand the pointcloud, firstly, we need to know the PrimeSensor and how it helps with the Kinect sensor. PrimeSensor is the depth image processing chip provided by PrimeSense. The following figure shows the way PrimeSensor works.

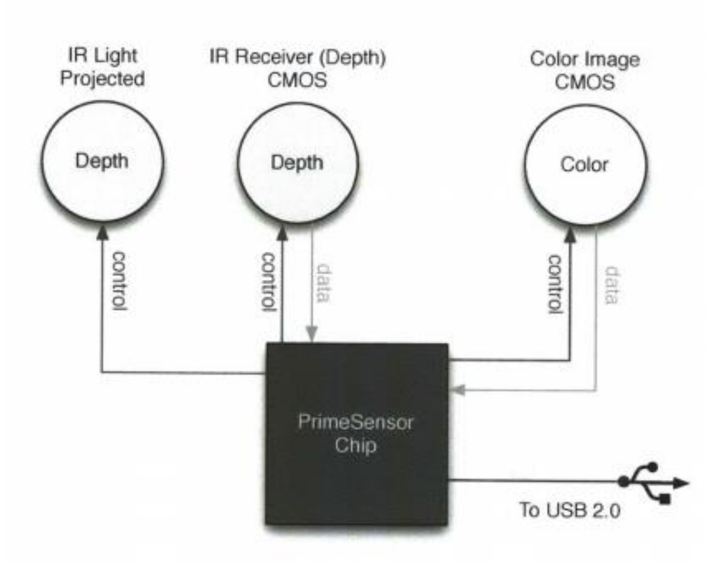


Figure 4.1: Kinect Flow-Chart

The final output of the PrimeSensor is data through USB cable containing infrared points and color image. Those data will be processed through Point Cloud Library (PCL).

4.1.2 Point Cloud Library (PCL)

PCL is a comprehensive free library for n-D Point Clouds and 3D geometry processing, which is also fully integrated with ROS[23]. A point cloud is a set of data points in some coordinate system. For Kinect, a three-dimensional coordinate system, all the points in cloud are usually defined by X, Y, and Z coordinates. PCL will process and display the point cloud and provide APIs toward some kind of post-processing. In this article, PCl will provide the basic function of human recognition and also involving further process.

Patrick Goebel[24].

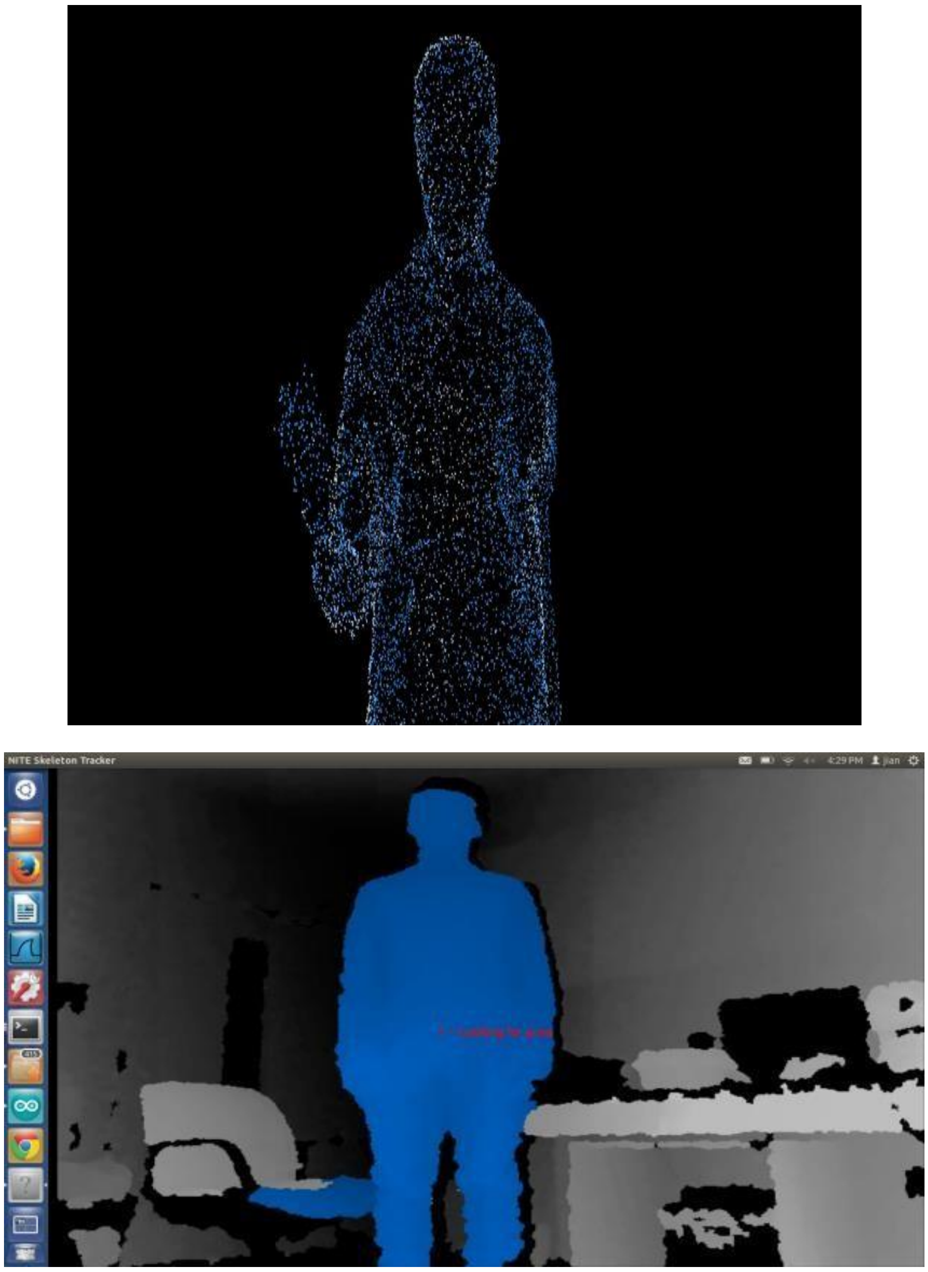


Figure 4.2 Point Cloud in Rviz

4.1.3 Skeleton_Marker

After obtaining the point cloud from Kinect sensor, the data will then go through Skeleton_Marker node and provide a human skeleton with each joint's three-dimensional coordinates. The Skeleton_Marker is provided and still maintained by Patrick Goebel[24].

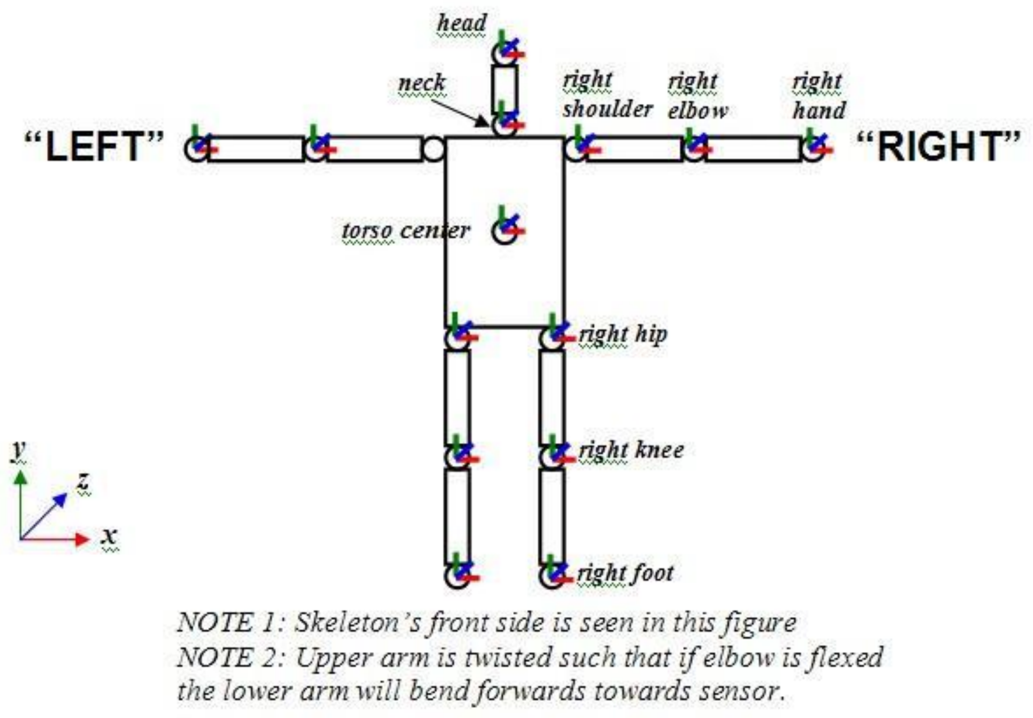

head
neck
right shoulder
right elbow
right hand
"LEFT"
"RIGHT"
torso center
right hip
right knee
right foot
y
z
x
NOTE 1: Skeleton's front side is seen in this figure
NOTE 2: Upper arm is twisted such that if elbow is flexed the lower arm will bend forwards towards sensor.

Figure 4.3 ; Skeleton Generated from Skeleton-Tracker

4.2 PyBrain

PyBrain is a versatile machine learning library for Python[25]. It has flexible, easy-to-use yet still powerful algorithm for machine learning tasks, including a variety of predefined environments and benchmarks to test and compare algorithms. It can provide following features: supervised learning, black-box optimization / evolutionary methods, reinforcement learning, architectures, compositionality, tasks and benchmarks, and speed optimization.

In this thesis, the neural network architecture in PyBrain was used to build the network for gesture recognition. The function used are buildNetwork and BackpropTrainer. A two hidden-layer MLP neural network was built and trained with EBP (error back propagation) algorithm.

Chapter 5

Artificial Neural Networks

Artificial Neural Networks (ANN’s) are able to provide enhancement for numbers of applications. They can be adapted in many different applications including academic research, commercial, and military use. There have been demonstrated improvements in pattern recognition, optimization, and scheduling in all these areas[26].

ANN’s sometimes provide higher accuracy than conventional technologies along with reasonable training time. Well-constructed ANN’s can provide high fault tolerance for system failure and provide high overall data throughput rates because of the parallel processing when embedded on hardware. A number of ANN hardware solutions have been developed, including neural network VLSI chips. They make it possible to integrate low-cost neural networks into existing systems without having to re-design the whole system, thus improving accuracy in pattern recognition, process and adaptive control, and noise filtering.

There are many types of ANN’s. Each architecture has areas of strength and preferred application.

5.1 History

The artificial neural network, by its name, was inspired by the biological neuron. Neurophysiologists explained the functions of human neurons – the movements of spikes and transmission. After that, several simulations have been developed. The first well-admitted neural network model – perceptron – was introduced in the year 1958 by Rosenblatt. Perceptron has been wildly recognized, enhanced and also provided confidence to various different architectures, such as Self-Organizing Neural Network, Associative Memory Neural Network, Multilayered Feedforward Neural Networks. In 1959, the first neural network for real world was developed, known as ADALINE and MADALINE for reducing telephone line echoes[10][11].

Neural network used to be in a dark age around 1970s due to some theoretical controversy about limitations of the perceptron and minimal funding resources. Things changed in the 1980s: books and conferences provided a number of positive feedbacks to neural network research results while media began to spread the works about neural network, even academic programs started to offer courses about neural networks at many major academic institutions.

There are many recently widely recognized neural network including: the back-propagation network, the Hopfield network, the Bidirectional Associative Memory network. Many significant achievements have been provided in this area which is attractive enough for further research with sufficient funding. Additionally, there are conferences and workshops for neural networks world-wide these years such as the Deep Learning Workshop.

5.2 Concept

ANN’s can provide estimates in the presence of uncertain information, similar to a human determining a situation based on former experience. By using electrical signal to re-format neurons in the brain, human will use changed neurons to make future decision and re-format neurons after the future result came out as right or wrong. Similarly, neurons in an ANN will modify connections during training, and respond to new situations.

An ANN is built upon numbers of neurons. Neurons in the neural network are basically individual processing units which can provide an output value from an input through a simple rule. Similar to a human neuron, a spike entering one end will or will not be transmitted to another end of a neuron, due to some electrical judgement[28]. These two kinds of movements are illustrated in Figure 5.1.

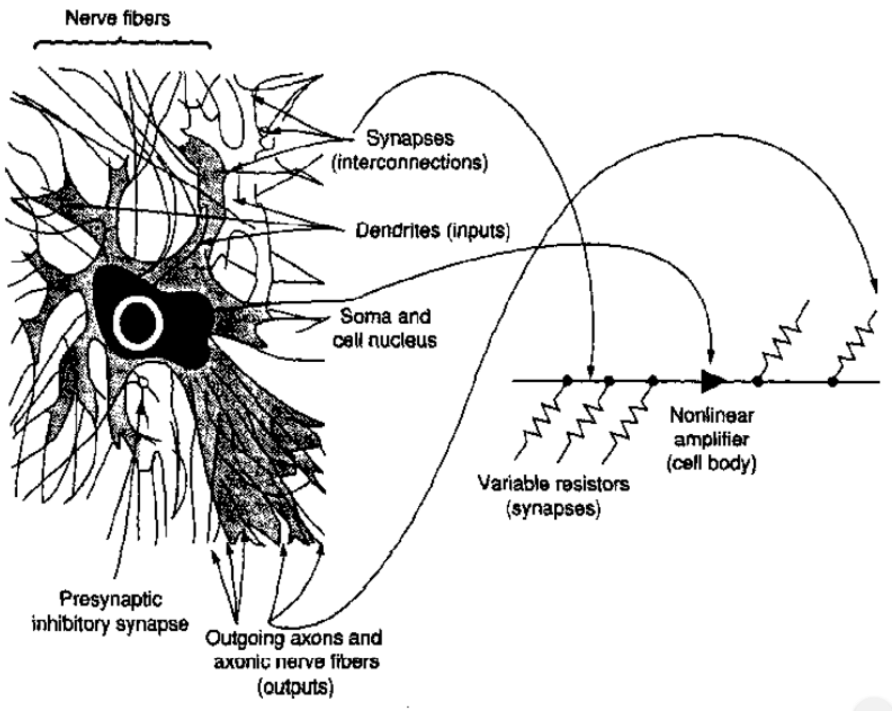

Nerve fibers
Synapses (interconnections)
Dendrites (inputs)
Soma and cell nucleus
Nonlinear amplifier (cell body)
Variable resistors (synapses)
Presynaptic inhibitory synapse
Outgoing axons and axonic nerve fibers (outputs)

Figure 5.1: Mechanism of Neural Transmission

A neuron, often recognized as a McCulloch-Pitts neuron, will be like Figure 5.2[29]:

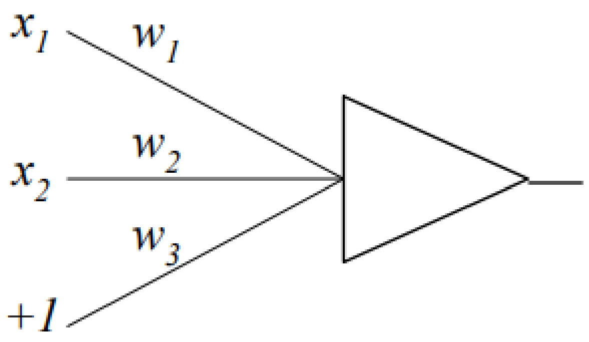


Figure 5.2： A McCulloch-Pitts Neuron.

The inputs are the x's and the bias of +1. The w's are weights which multiply the inputs. The neuron applies an activation function to the weighted sum of inputs, and thereby determines the output value.

5.3 Neuron Logic

A neuron will have multiple inputs, coming from initial input or other neurons' outputs, and a weight input. A weight input will provide a bias for a neuron to help the neuron partly independent from other neurons and adjust the judging rule, usually we assign the weight as 1. The judging rule is also known as an activation function[26][30]–[32].

A neuron could be a unipolar neuron or a bipolar neuron. A unipolar neuron means the output of the neuron will between 0 and 1. A bipolar neuron means the output of the neuron will between -1 and 1. These two kinds of neurons will provide difference convenience in different circumstances, also inspired by human neurons.

The total input of a neuron will be the summation of the inputs:

$$s1 = x1 * w1 + x2 * w2 + 1 * w3 \tag{5.1}$$

Then the input will go through the activation function. There are various types of activation function for a neuron, such as linear function (identity function), step function, logistic function, tanh function… Different kinds of functions will provide different resolution for a common problem, and will help in different kinds of situations. For example, a step function will be helpful in an efficiency-first neural network because it will have low precision and thus reduce the computation time. A step function will be shown below. There will be two different kind of step function due to the type of the neuron.

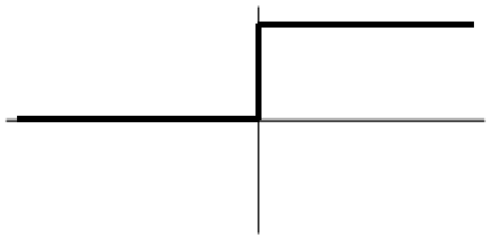


Figure 5.3: 0-1 Step Function

$$\mathrm{sign(x)} = \begin{cases} 1\ if\ x > 0 \\ 0.5\ if\ x = 0 \\ 0\ if\ x < 0 \end{cases} \tag{5.2}$$

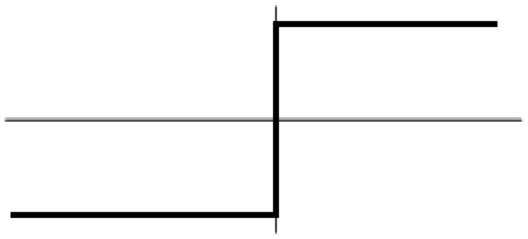


Figure 5.4: -1 – 1 Step Function

$$\mathrm{sgn(x)} = \begin{cases} 1\ if\ x > 0 \\ 0\ if\ x = 0 \\ -1\ if\ x < 0 \end{cases} \tag{5.3}$$

Another type of activation function will be Tanh function. The expression of the tanh function will be:

$$\tanh(x) = \frac{2}{1+e^{-2x}} - 1 \tag{5.4}$$

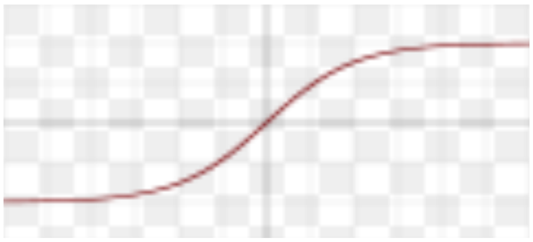


Figure 5.5: Tanh Function

The tanh function is used for bipolar neuron.

The major difference between these two types of activation functions will be the further training possibility. A step function is not suitable for high-precision training since it is not continuously differentiable, while a Tanh function can be differentiated everywhere for further training.

5.4 Error Back-Propagation (EBP) Training

The Error Back-Propagation (EBP) training will be illustrated by consideringone bipolar neuron. As mentioned before, the ANN will use some of the already known information to determine information with unknown result. Thus, the dataset for training will be patterns which includes inputs value and desired outputs values respectively.

5.4.1 Single Neuron Training

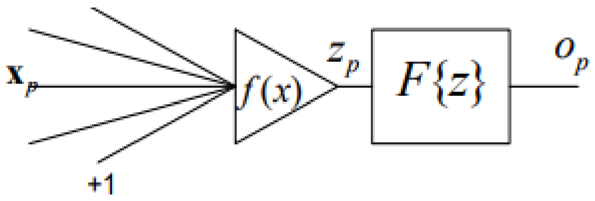


Figure 5.6: Single Neural with Output Function

For each pattern P (assuming it has n inputs and 1 outputs), the output of the one-neuron neural network will be,

$$O_P = F\{f(w_1 x_{P1} + w_2 x_{P2} + \cdots + w_n x_{Pn})\} \tag{5.5}$$

And the error between the real output and desired output will be,

$$\mathrm{TE} = \sum_{p=1}^{nP} [d_P - O_P]^2 \tag{5.6}$$

Training the neuron means changing the weight for each input to adjust the neuron's function since the weight is the only parameter that will lead the neuron to a different result. The way to change the weights will rely on the difference between the real output and the desired output.

Each weight will contribute to a different orientation for the real output. Which means, if one weight is taking the result far away from the desired output, this weight should be changed to keeping the result closer to the desired output. The gradient will help solve this problem. For each gradient, it's easy to find out that choosing the opposite direction to modify the weight will lead to a better result. Calculation will be described below,

$$\frac{d(TE)}{dw_i} = -\sum_{P=1}^{nP} [2(d_P - O_P) * \frac{dO_P}{dz_P} * \frac{dz_P}{dI_P} * \frac{dI_P}{dw_i}] \tag{5.7}$$

$$I_P = w_1 x_{P1} + w_2 x_{P2} + \cdots + w_n x_{Pn} \tag{5.8}$$

$$\frac{d(TE)}{dw_i} = -2\sum_{P=1}^{nP} [(d_P - O_P) * F'\{z_P\} * f'(I_P) * x_{Pi}] \tag{5.9}$$

For each weight, we should apply a small step opposite to the original gradient. The change that will apply to a weight will be,

$$\Delta w_{Pi} \sim \beta \sum_{P=1}^{nP} [(d_P - O_P) * F'\{z_P\} * f'(I_P) * x_{Pi}] \tag{5.10}$$

This expression can be applied to all the weights, so it can be represented as,

$$\Delta w_P = \beta \sum_{P=1}^{nP} [(d_P - O_P) * F'\{z_P\} * f'(I_P) * x_P] \tag{5.11}$$

If the neuron has only one output, we can have a simpler function version,

$$\Delta w_P = \beta \sum_{P=1}^{nP} [(d_P - O_P) * f'(I_P) * x_P] \tag{5.12}$$

This is also the activation function used for the robot.

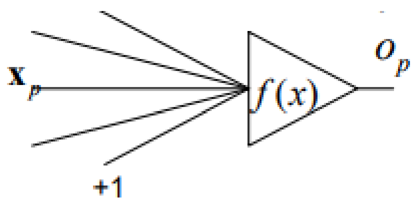
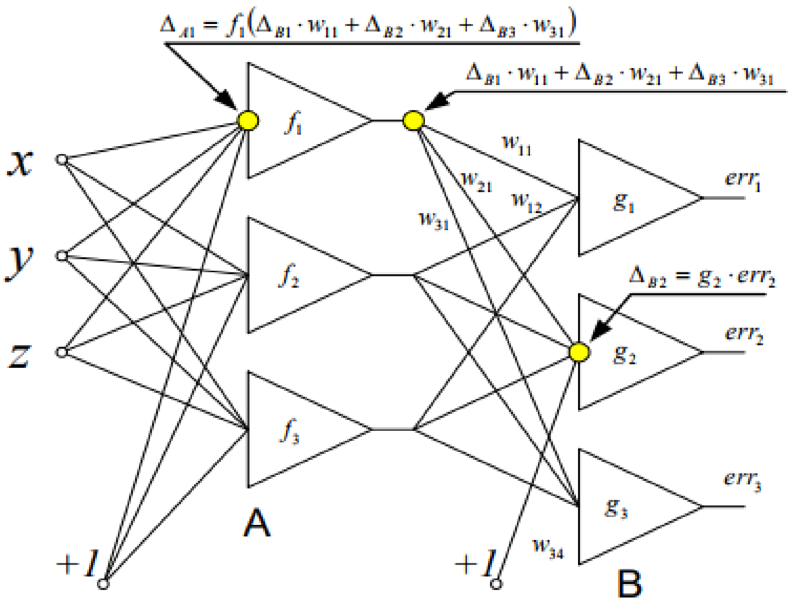


Figure 5.7: Single Neuron with Direct Output

5.4.2 Neural Network Training

For a whole neural network, the training process will be almost the same as the single neuron training.



$\Delta_{A1} = f_1(\Delta_{B1} \cdot w_{11} + \Delta_{B2} \cdot w_{21} + \Delta_{B3} \cdot w_{31})$
$\Delta_{B1} \cdot w_{11} + \Delta_{B2} \cdot w_{21} + \Delta_{B3} \cdot w_{31}$
x
y
z
f_1
f_2
f_3
w_{11}
w_{21}
w_{12}
w_{31}
g_1
err_1
$\Delta_{B2} = g_2 \cdot err_2$
g_2
err_2
g_3
err_3
w_{34}
$+1$
A
$+1$
B

Figure 5.8: A Two-layer Neural Network

The error and gradient for each neuron will be calculated first and updated at the same time after calculation.

Chapter 6

Gesture Recognition Algorithm

In this chapter, the actual implementation of the neural network will be introduced.

6.1 Raw Data Analysis

The data received from Skeleton_Marker node will include three-dimensional coordinates x, y, and z. The Skeleton_Marker node will post total 9 joints' dimensional data as right_hand, left_hand, right_arm, left_arm, torso… The neural network will be only trained with dataset collected from right_hand topic.

In this setup, a gesture can be recognized as a series of points which have a specific trajectory if listed in time sequence. On first thought, it would be reasonable to train the network with those points' coordinates. However, while implementing these data into the network, it is easily to observe that using raw value from coordinates will not be a good choice since different kinds of gestures will share the same points and probably have a high rate of overlap. Also, using the absolute coordinate value will ruin the robustness of the system – the system will only recognize gestures in a certain distance. A solution to this situation is to use the difference between every two adjacent points as the input series to train the neural network. In this way, the character for each gesture will be successfully distinguished. Gestures like clockwise and counter-clockwise will share the same trajectory but have significantly different datasets for training.

6.2 Data Acquisition

To provide sufficient data including enough knowledge for the neural network, a moderate dataset will be provided for each gesture. Through using pickle library in Python, the robot will be input with excel files including coordinates as input and output. The robot will record a 3-second data in 5 excel files for each gesture.

Chapter 7

Robot Implementation and Validation

7.1 Implementation

The trained neural network was implemented with the robot. The robot test platform is shown in Figure 7.1:

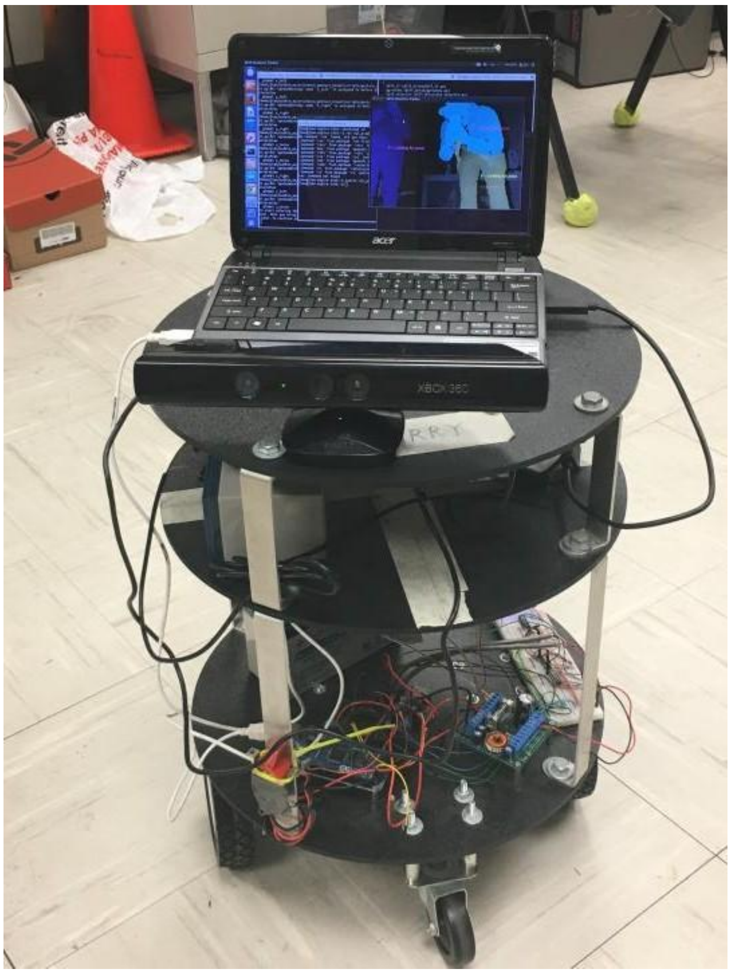


Figure 7.1: The Robot test platform.

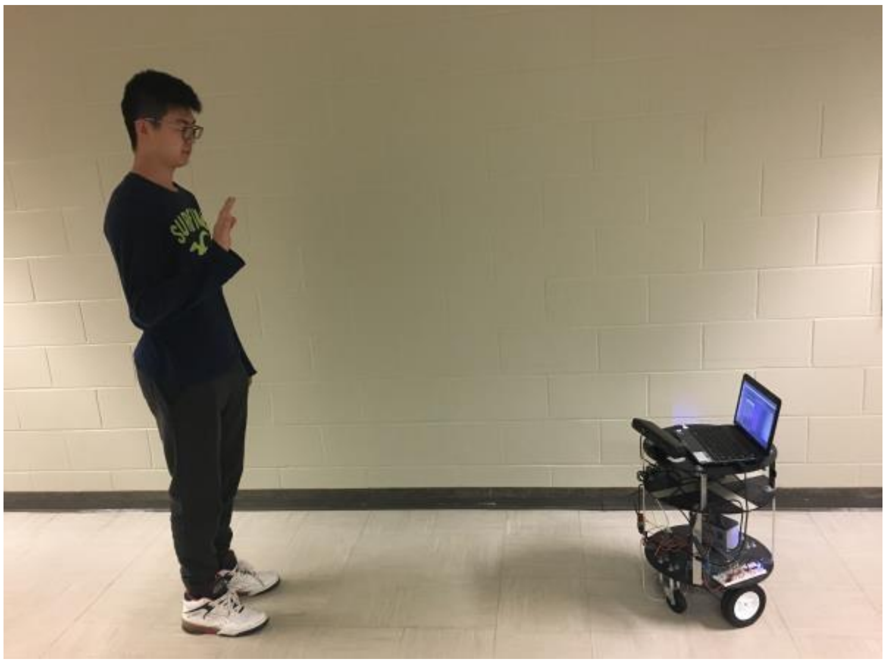


Figure 7.2: Comparison Between Human and Robot

The ROS node map will be:

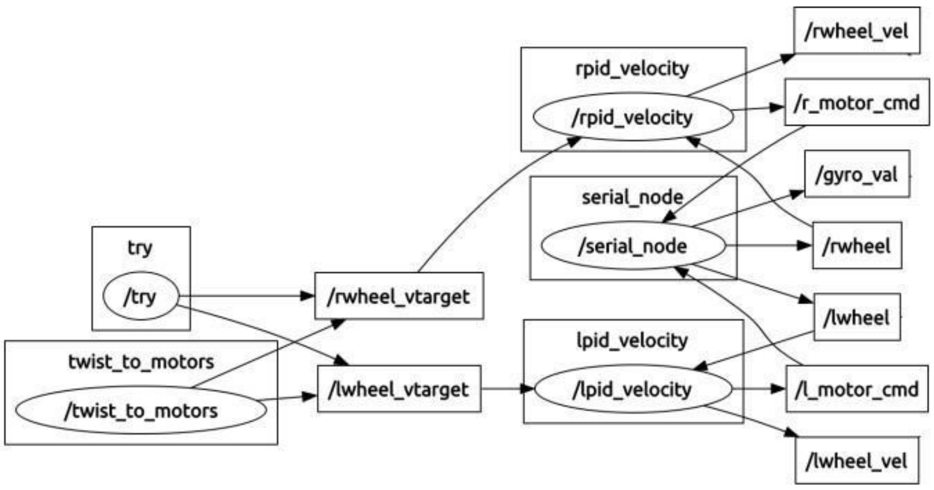


Figure 7.3: The ROS Node Map. This shows the data flow inside the ROS environment.

There will be five human gestures being trained through the neural network: forward, backward, clockwise, counter-clockwise, stop. All images are illustrated in the mirror view.

A forward gesture will be:

Right arm level as shoulder – Right forearm raise up till vertical – Level back right forearm

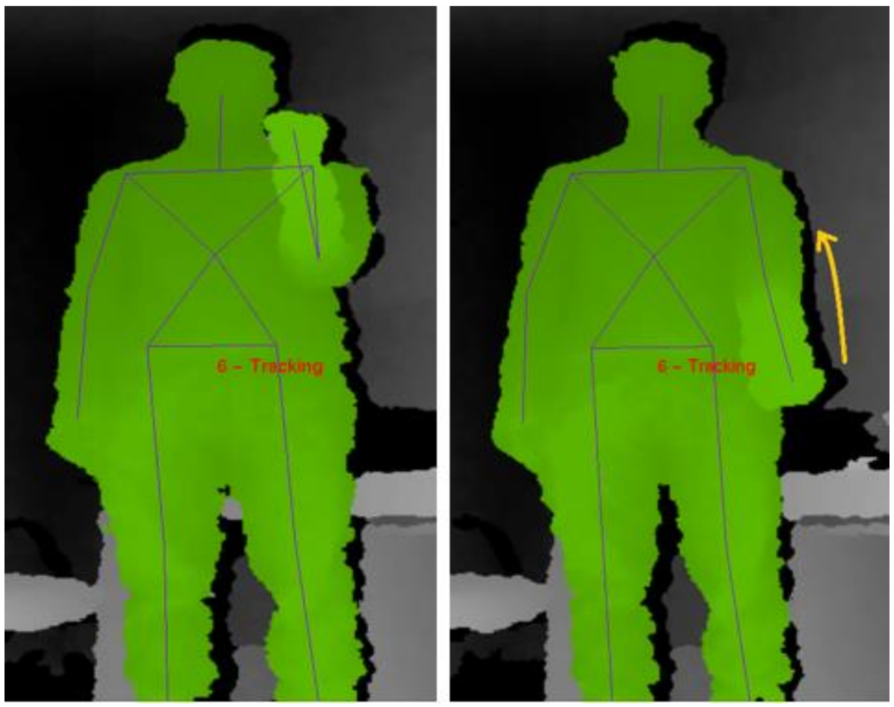


Figure 7.4: A Forward Gesture

A backward gesture will be:

Level the upper right arm and put right forearm vertically close to right ear – keep upper right arm in the same position and move right forearm away from right ear horizontally – let right forarm go back to the same position

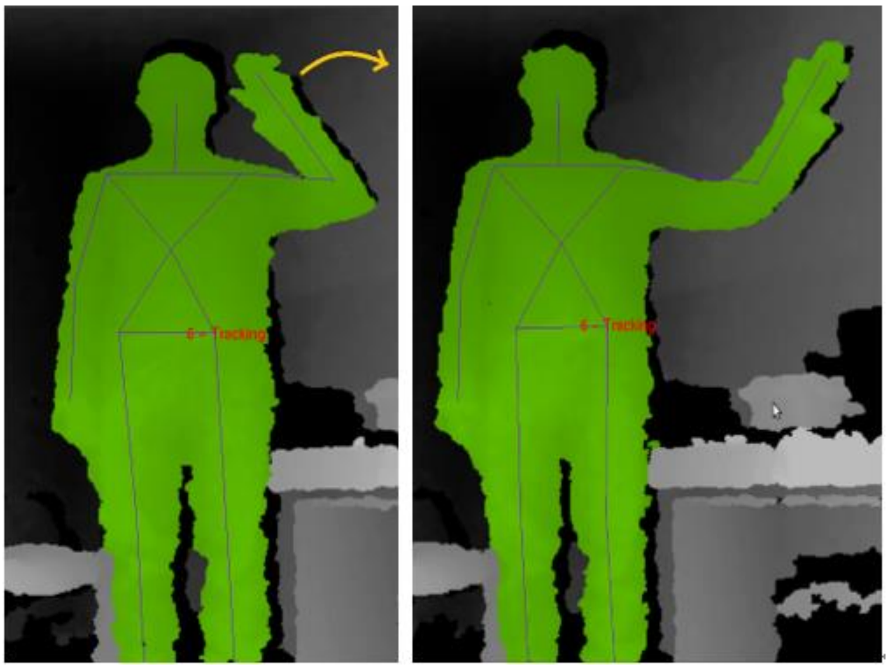


Figure 7.5: A Backward Gesture

A stop gesture will be:

Raise right hand over head while right upper arm keeping a nearly horizontal position – put down the right hand vertically till under waist – put right hand back over head

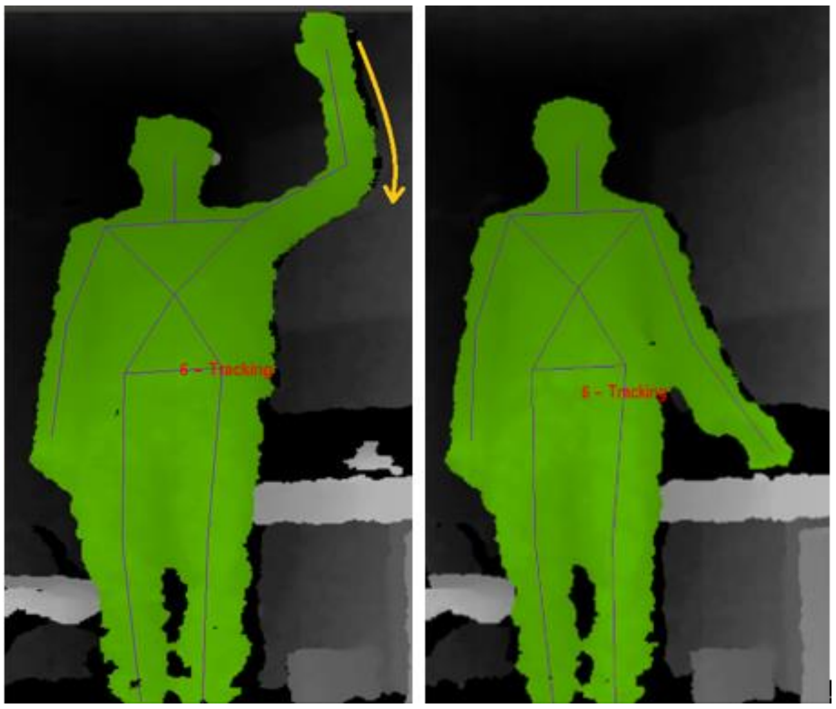


Figure 7.6: A Stop Gesture

A clockwise gesture will be:

Let right upper arm naturally relax and right forearm horiaontally places while right hand stand vertically – let right hand perform a circler clockwisely – keep 3 seconds and stop

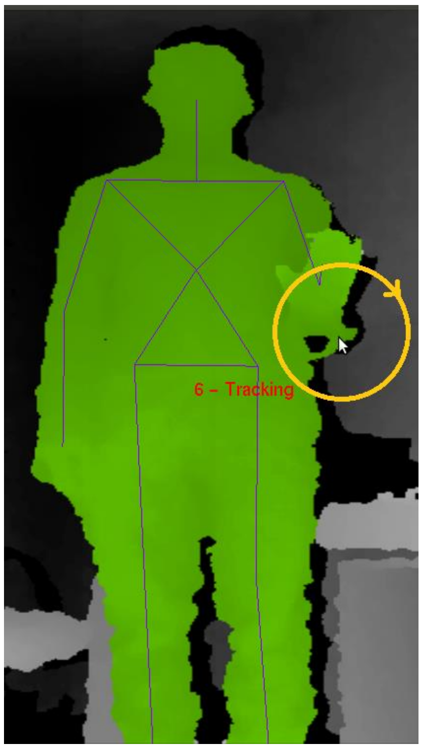


Figure 7.7: A Clockwise Gesture

A counter-clockwise gesture will be:

Let right upper arm naturally relax and right forearm horiaontally places while right hand stand vertically – let right hand perform a circler counter-clockwisely – keep 3 seconds and stop

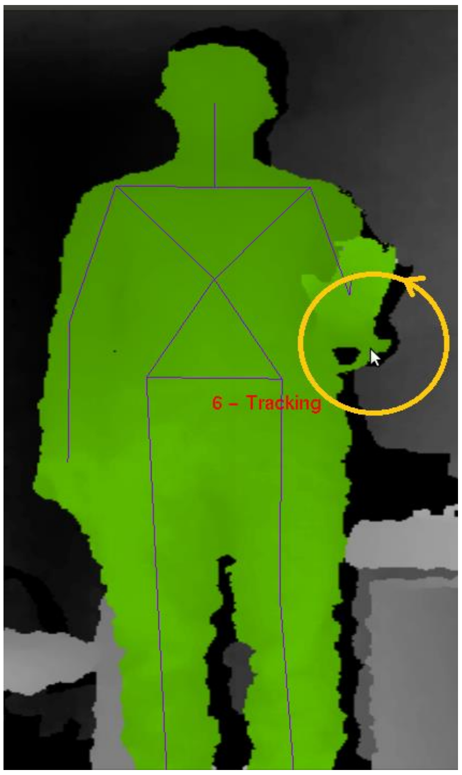


Figure 7.8: A Counter-clockwise Gesture

For training, each gesture will have 5 csv files containing three-seconds coordinate data as training dataset. Totally there will be 25 data files. For training, each gesture will be represented as a number. Here we just choose 1, 2, 3, 4, 5 as the desired output result.

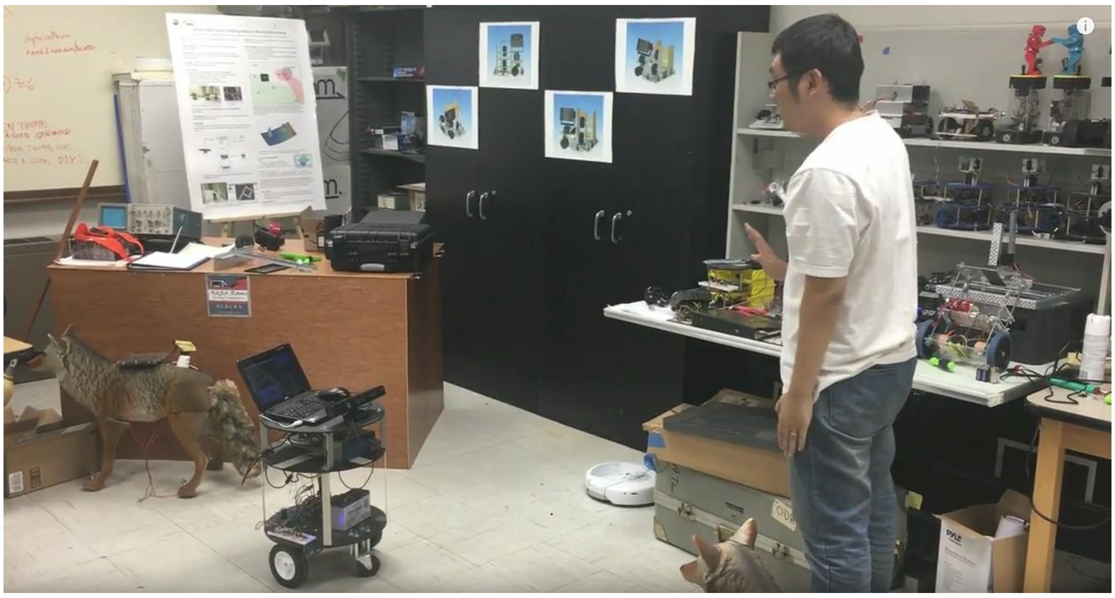
After training, the neural network is capable of recognizing the gestures respectively.

7.2 Validation

The validation process will be very much similar to the data input process. The tester will stand in front of the robot. For each gesture, the tester will need to do the Pi post to calibration at first. Then the tester will hold the right arm at the initial postion for gesture respectively while wait for left arm's start signal. The start signal will be released as: put left hand straight pointing up and pull it down below the torso point.

As soon as the left hand becomes lower than the torso point, the tester will perform right hand gesture and the robot will start to record for three seconds. Then the robot will put the recorded data into the trained neural network and return an integer between 1 and 5 indicating the current gesture.

Ten subjects were invited and each taking turns tested the functionality of the robot. Below are images for some of the subjects.



Subjects did several tests and the result is relatively good. The accuracy for all five gestures ranged from 86% to 96.8%. Among all gestures, Clockwise gesture has the best recognition rate and Backward/Stop gestures share below average performance.

The latter result is reasonable, since the Backward and the Stop gestures share a similar right hand trajectory and orientation (waving). Though clockwise and counter-clockwise share the same trajectory, they have different orientation.

Table 7-1: Gesture Experimental Data. The row headings indicate the actual gesture performed, and the column headings indicate the gesture interpreted by the artificial neural network.

Interpreted by the ANN / Actual Gestures performed	Forward	Backward	Stop	Clockwise	Counter Clockwise
Forward	90%	4%	4%	2%	0%
Backward	2%	66%	28%	0%	4%
Stop	10%	24%	62%	0%	4%
Clockwise	0%	0%	0%	94%	6%
Counter-Clockwise	0%	0%	0%	10%	90%

Chapter 8

Conclusion and Future Work

From the experimental results, it can be revealed that based on the effective data and well trained neural network, a robot can recognize human gestures and follow them as instructions for human-robot interaction with satisfying results.

8.1 Limitations

The result revealed that due to the mechanism of the neural network, distinctive features must be provided for its training process. This supervised training process limits the expansion of gestures – if a new gesture needs to be recognized, all the already trained gestures will be trained again and this will be a time consuming process. In the mean time, data features for trainging must be provided manully, which means if the features were mistakenly determined, the whole network will be suspended.

For the hardware part, currently the Kinect sensor is placed on the designated spot with limited freedom, which leads to an inconvient but necessary re-calibration after each movement.

8.2 Future Work

From the neural network perspective, there could be better training process, such as reinforcement learning which will be possibly better to handle new incoming gesture sets. Also the data features can be improved to make better distinction between gestures.

From the hardware perspective, the Kinect sensor could be placed at a better location. If mounted on a servo’d platform, the sensor could possibly track people while the robot is moving around, which would reduce the calibration time and improve efficiency. Alternatively, a remotely-located Kinect sensor would be able to provide more flexiblity when the robot is out of sight.

References:

[1] D. S. Roland Siegwart,Illah Reza Nourbakhsh, *Introduction to Autonomous Mobile Robots*. 2011.

[2] B. Pappas, “Multi-Robot Frontier Based Map Coverage Using the ROS Environment,” *Master@Auburn Univ.*, 2014.

[3] M. Quigley *et al.*, “ROS: an open-source Robot Operating System,” *Icra*, vol. 3, no. Figure 1, p. 5, 2009.

[4] Z. Zhang, “Microsoft Kinect Sensor and Its Effect,” *IEEE Multimed.*, vol. 19, no. 2, pp. 4–10, Feb. 2012.

[5] E. T. (Edward T. Hall, *The silent language*. Anchor Books, 1990.

[6] “SoftKinetic - 3D Vision Leader.” [Online]. Available: https://www.softkinetic.com/. [Accessed: 24-Oct-2017].

[7] Ng Yong, Yi Kevin, S. Ranganath, and D. Ghosh, “Trajectory modeling in gesture recognition using cybergloves and magnetic trackers,” in *2004 IEEE Region 10 Conference TENCON 2004.*, vol. A, pp. 571–574.

[8] M. B. Kaaniche, “Gesture Recognition From Video Sequences,” p. 129, 2009.

[9] H. Liu and L. Wang, “Gesture recognition for human-robot collaboration: A review,” *International Journal of Industrial Ergonomics*, 2016.

[10] S. Mitra and T. Acharya, “Gesture recognition: A survey,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 37, no. 3, pp. 311–324, 2007.

[11] S. S. Rautaray and A. Agrawal, “Vision based hand gesture recognition for human computer interaction: a survey,” *Artif. Intell. Rev.*, vol. 43, no. 1, pp. 1–

54, 2012.

[12] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” *Proc. 1999 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Cat No PR00149*, vol. 2, no. c, pp. 246–252, 1999.

[13] Wei-Lwun Lu and J. J. Little, “Simultaneous Tracking and Action Recognition using the PCA-HOG Descriptor,” in *The 3rd Canadian Conference on Computer and Robot Vision (CRV’06)*, pp. 6–6.

[14] D. Introduction and K. Neighbor, “God , Your Book Is Great !! A Detailed Introduction to K-Nearest Neighbor (KNN) Algorithm,” pp. 1–22, 2010.

[15] D. Ramage, “Hidden Markov models fundamentals,” *Lect. Notes. http//cs229. stanford. edu/section/ …*, pp. 1–13, 2007.

[16] T. E. Starner, “Visual Recognition of American Sign Language Using Hidden Markov Models MA R 2 2 1995 Visual Recognition of American Sign Language Using Hidden Markov Models,” 1991.

[17] L. Sigal, S. Sclaroff, and V. Athitsos, “Skin color-based video segmentation under time-varying illumination,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 7, pp. 862–877, Jul. 2004.

[18] G. Bernstein, N. Lotocky, and D. Gallagher, “Robot Recognition of Military Gestures CS 4758 Term Project,” 2012.

[19] S. M. Goza, R. O. Ambrose, M. A. Diftler, and I. M. Spain, “Telepresence control of the NASA/DARPA robonaut on a mobility platform,” in *Proceedings of the 2004 conference on Human factors in computing systems*

- *CHI ’04*, 2004, pp. 623–629.

[20] “AU Smart Cart Senior Design Project Report Fall 2012 http://hdl.handle.net/11200/49045,” 2012.

[21] PrimeSense, “OpenNI User Guide,” *OpenNI User Guid.*, vol. 1, no. June, p. 44, 2011.

[22] M. K, “OpenNI Standard Launched,” *kinecthacks.net*, Dec. 2010.

[23] R. B. Rusu and S. Cousins, “3D is here: point cloud library,” *IEEE Int. Conf. Robot. Autom.*, pp. 1–4, 2011.

[24] G. F. He, J. W. Park, S. K. Kang, and S. T. Jung, “Development of gesture recognition-based serious games,” *Proc. - IEEE-EMBS Int. Conf. Biomed. Heal. Informatics Glob. Gd. Chall. Heal. Informatics, BHI 2012*, vol. 25, no. Bhi, pp. 922–925, 2012.

[25] T. Schaul *et al.*, “PyBrain,” *J. Mach. Learn. Res.*, vol. 11, pp. 743–746, 2010.

[26] A. J. Maren, C. T. Harston, and R. M. Pap, *Handbook of neural computing applications*. Academic Press, 1990.

[27] B. Widrow and M. A. Lehr, “30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation,” *Proc. IEEE*, vol. 78, no. 9, pp. 1415–1442, 1990.

[28] W. Maass and C. M. Bishop, *Pulsed Neural Networks*, vol. 275. 1999.

[29] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, Dec. 1943.

[30] B. M. Wilamowski, “Comparison of Training Algorithms and Network

Architectures," *IEEE Intell. Eng. Syst. Conf. Costa Rica*, pp. 17–11, 2013.

[31] H. Yu and B. M. Wilamowski, "Neural Network Training with Second Order Algorithms," *Human–Computer Syst. Interact. Backgrounds Appl. 2*, pp. 463–476, 2012.

[32] B. M. Wilamowski, "Understanding of Neural Networks," in *Industrial Electronics Handbook, vol. 5 – Intelligent Systems*, 2011, pp. 5-1-12.