American Journal of Science and Technology
2016; 3(1): 17-24
Published online February 2, 2016 (http://www.aascit.org/journal/ajst)
ISSN: 2375-3846

American Association for Science and Technology

Keywords
Gesture Detection,
Kinect Sensor,
Graphical Interface and Robotic Arm

Received: December 11, 2015
Revised: December 26, 2015
Accepted: December 28, 2015

Development of a Gesture Detection System for the Control of a Robotic Arm

Osahor Uche, L. O. Kehinde

Department of Electronic and Electrical Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria

Email address
ucheosahor@gmail.com (O. Uche)

Citation
Osahor Uche, L. O. Kehinde. Development of a Gesture Detection System for the Control of a Robotic Arm. *American Journal of Science and Technology*. Vol. 3, No. 1, 2016, pp. 17-24.

Abstract
This study developed a gesture detection system for the control of a robotic arm. This was achieved with the view of developing a simple and user friendly approach for robotic arm manipulation. A Microsoft Xbox™ Kinect sensor was used as the input device for retrieving real-time joint information from the user. Kinematic algorithms were developed by applying Denavit-Hartenberg parameters and geometric equations. The acquired gestures from the human subject were used to carry out various kinematic routines as deemed fit by the user. A graphical interface was also developed to provide real time feedback to the user. The robot was able to execute basic gestures as instructed by the user rated at about 80% success rate. Most gestures developed were effective enough for the scope of this project. However, minimal gesture detection errors were recorded averaging below 20%. A standard deviation error of 3.8 at a variance error of approximately 14.1 was recorded. The study implemented an effective gesture system that was capable of executing the basic routines required for robotic arm manipulation in real time.

1. Introduction

The ability to interact with technology has become one of the greatest achievements in recent times. A myriad of techniques have been developed by engineers and scientists to manipulate, instruct and communicate with electrical/electromechanical devices. This form of interaction is mostly established between human and machine or machine to machine (M2M). Most devices that process information require some form of human interaction to manipulate the content of their memory. Various interfaces have been developed to act as a link between man and machine [1]. The basic means of interaction include keyboards, touch screens, joysticks, touch pads etc. In recent times, Human machine interaction (HMI) has experienced significant development over the years and has integrated to all areas of life from computer gaming to more sensitive roles like robotic surgery, unmanned aerial vehicle maneuvering and Automobile manufacturing [2]. The proposed project is aimed at adapting one of the latest technologies known as gesture control. A sensor known as the “Kinect”, developed by Microsoft was used to acquire gesture commands from the user. The Kinect is a sensor that comprises of an RGB (red green blue) camera, 3-D depth camera and a multi array microphone. The 3-D depth camera is designed to monitor the gestures from the human skeletal system by processing the data from the depth camera. Hence, the gestures acquired from the human is processed into instructional code for various applications.

2. Review

The typical approach that govern robotic arm manipulation apply principles which include muscular control, angular displacement, embedded control, mind control, gesture control etc. A brief review of previous works that are related to this work are described in this section.

Steven [3] developed a "Natural User Interface for Robot Task Assignment". The robotic system they employed performed object detection, grasp and motion planning, while the human operator handled high-level tasks. Their method tracked a human hand from a specified virtual reality workstation using a Kinect. The data acquired by the user interface is used to manipulate virtual objects in the robots workspace as detected by the perception system via a second Kinect sensor. However, this approach was computationally intensive and complex to implement and quite expensive for mass deployment.

Shobhitha [4] made use of an electromyogram to acquire electrical signals by mounting surface electrodes on the users forearm. The recorded muscular activity was processed and used to control a robotic arm. They acquired results from a user by executing flexion and extension motions within a duration of 10 seconds for 10 consecutive times. The signals are then processed in LabVIEW. They also incorporated haptic technology that transmitted stimulus signals from the human arm. A CAD model of the robotic arm was developed using Solid works software (Solid-Works, 2015). Other components that make up the system include an Arduino UNO and Arduino Mega for transmitting wireless signals to the robotic arm

Olawale [5] adapted the most widely used means for the control of robotic arms. Their work interfaced a robotic arm (PUMA 560) with a pre-programmed 8051 microcontroller. Most of the algorithms required to execute the routines were programmed into the processor. The configuration of their design incorporated an MCU 8051 that coordinates the operation of the robotic arm by collecting information from other components of the design, which include a LATCH 74LS373, an 8255 PIO and an EPROM 2732.

The knowledge gained from the aforementioned techniques summarize the key approaches in robot control with regards to human-machine interaction. However, a closer insight into the various concepts reveal loop holes that were addressed in this research. A key area that was improved upon was the graphical interface. A more user friendly interface providing real time view of the three dimensional data of the robotic arm as well as the values generated from the inverse and forward kinematic algorithms was implemented.

The entire work was designed to ensure that the user has full control of the robotic arm without any previous programming knowledge. Finally, a gesture recognition algorithm was developed that filtered unwanted gesture commands, so as to prevent bottlenecks that might arise as a result of false gesture commands

3. The Gesture Detection System

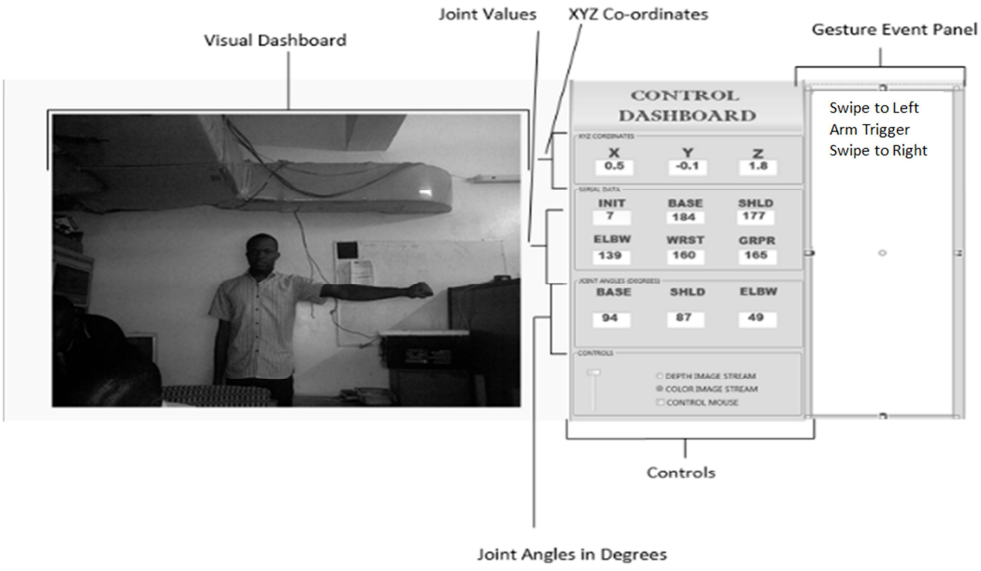

Figure 1. *A graphical user interface.*

A Microsoft Kinect 3-D depth camera acquires skeletal images of the human skeletal frame, the images were captured at 15fps for this experiment. The data acquired contain key information that represent the movements of tracked joints generated from the arm gestures. This gestures represent arm movements that are preprogrammed to activate commands if successfully executed. An inverse and forward kinematics algorithm was developed by applying the Denavit-Hartenberg (D-H) parameters and geometric equations that represent the joints of a five degree of freedom (DOF) robotic arm.

The gestures from the human subject were mapped to the

equivalent joint of the robotic arm and converted to distance and degrees by adapting various mathematical principles that represent respective joints of the robotic arm. The acquired information from the human subject was used to execute control commands. These commands were adapted into routines that imitate real life scenarios. A graphical user interface was also developed to provide a user friendly interface for the user as shown in Figure 1.

The gesture detection system comprises of four major sections as illustrated in the configuration schematic of Figure 2. The sections include:

i. Gesture Input.
ii. Image Acquisition Device - Kinect.
iii. Computer System.
iv. Robotic Arm.

3.1. Gesture Input

This form of communication depends on the discussion context (the type of discussion determines the hand gesture maneuvers) of the individual. Every conversation has a specific set of gestures that are dependent on the user's discretion. The gesture inputs from the humans were captured by the Kinect at a set speed of 15 frames per second for both color and depth images and the frame speeds can be changed to suit the design specifications. The time frame of 15fps was sufficient to acquire relevant inputs from the arm movements [6]. The two main properties that are acquired from the human joints are the speed and displacement of each joint with respect to their distance from the root joint (Hip Centre). Reference joints were set as markers to ensure that gestures don't conflict each other.

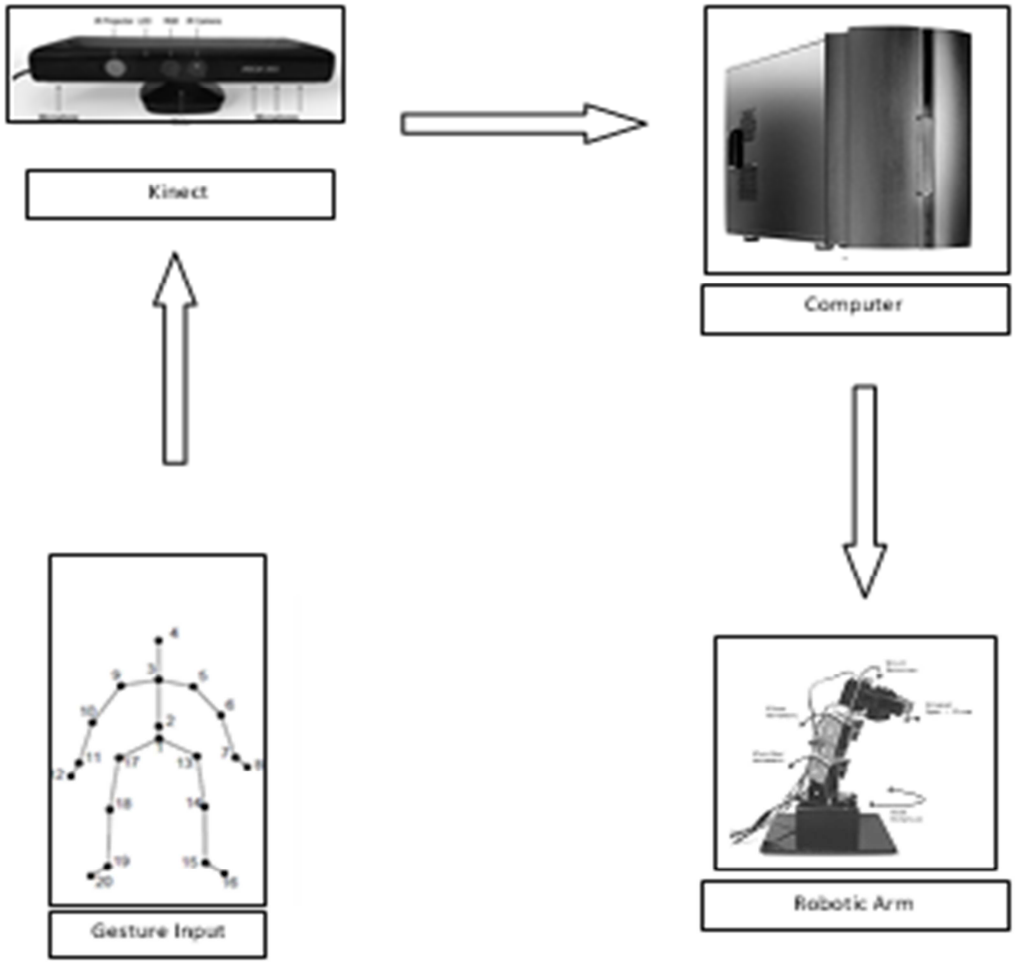

Figure 2. *The gesture detection system.*

3.2. Skeletal Image Acquisition Device

The skeletal images were acquired with the Microsoft Kinect® sensor version 1414. The sensors were designed to track skeletal data at a maximum vertical angle of 47 degrees and a horizontal angle of 57 degrees [7] However in order to suit the design configuration, the angles from the human arm were rescaled to lie within a workable range of the Kinect sensors maximum angles [8].

3.3. Computer System

The computer system plays a key role in the gesture detection system. It should be noted that due to the frame speed (30 fps) the Kinect sensor acquires the images, a lot of system resources was demanded from the computer. For this work, the major system specifications are listed below.

i. Intel® Dual core processor - 2.4 GHz Pentium ® CPU.
ii. RAM – 4.0 GB.
iii. Operating System - Widows 7 Ultimate 32 bit.

All essential system drivers such as the Video Graphics Array (VGA), Universal Serial Bus USB, Bus controller etc. were in use in the course of the project. For the software components, A Microsoft Visual Studio 2012 was installed. A version 1.8 Software Development Kit (SDK) was also installed as part of the libraries for the project.

3.4. Robotic Arm

An RA-01 robotic arm with 5 degrees of freedom was connected to an SMC-05 controller board. The controller board was connected to the computer via an RS232 serial port and the microcontroller handled. The pulse width modulation for controlling the five motors of the robotic arm. The microprocessor receives instructions in the following format: [7, #, #, #, #, #] where "7" is the initialization/start code that informs the processor that the next set of 5 bytes represent the desired angular displacements for the robotic arm joints. The first '#' specifies the desired angular position for the base motor, the second for the shoulder motor, the third for the elbow motor, the fourth for the wrist motor and fifth for the gripper motor. A baud rate of 9600 was prescribed for the serial communication of the robotic arm by the manufacturers. However several trials confirmed that a baud rate 0f 19200 would provide better results [8]. A baud rate of 19200 implies that 19200/54bits = 355.56 commands are transmitted per second where 54 bits represent the six command bytes plus a stop byte [(6+1) x 8 = 54]. This means that one command is transmitted every (1/355.56ms) = 28milliseconds. However, the manufacturers specified that commands should be sent to the motors every 18 milliseconds. If this is followed, it connotes that at a baud rate of 19200, a command is sent at for 18milliseconds while no command is sent for about 10 milliseconds i.e. for only 2/3 the required time. Although this configuration was not perfect as per the specification requirements, it was sufficient for transmissions. A higher baud rate of 38400 did not produce any improvement in outcome.

4. Forward Kinematics

The forward kinematics of the end-effector with respect to the base frame was determined by multiplying all the ${}^{i-1}_{i}T$ Matrices, where

$$ {}^{base}_{end\_effector}T = {}^{0}_{1}T\,{}^{1}_{2}T \ldots {}^{n-1}_{n}T \quad (1) $$

Alternatively, it can be represented as

$$ {}^{base}_{end\_effector}T = \begin{vmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (2) $$

Where r_{kj}'s represent the rotational elements of transformation matrix (k and j = 1, 2 and 3). p_x, p_y,p_z denote the elements of the position vector. For the joint axes of RA-01 where r_{kj}'s represent the rotational elements of transformation matrix (k and j = 1, 2 and 3). p_x, p_y,p_z denote the elements of the position vector. For the joint axes of RA-01 robotic arm (Figure 3).

The transformation from frame to the base frame is given as:

$$ {}^{0}_{1}T = \begin{vmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (3) $$

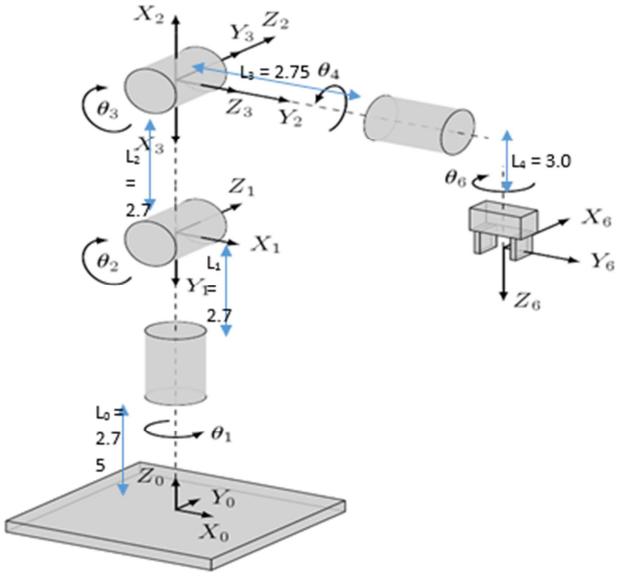

Figure 3. *Robotic Arm Joint Axes.*

Equation 3.5 shows the transformation matrix of the shoulder to the joint.

$$ {}^{1}_{2}T = \begin{vmatrix} \cos(90+\theta_2) & -\sin(90+\theta_2) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin(90+\theta_2) & \cos(90+\theta_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} $$

$$ = \begin{vmatrix} -\sin\theta_2 & \cos\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (4) $$

Equation 3.6 shows the transformation matrix of the elbow to shoulder joint

$$ {}^{2}_{3}T = \begin{vmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & L_2 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (5) $$

Equation 3.7 shows the transformation matrix of the base to elbow joint

$$
{}^{3}_{4}T = \begin{vmatrix} \cos(180+\theta_4) & -\sin(180+\theta_4) & 0 & L_3 \\ 0 & 0 & -1 & 0 \\ \sin(180+\theta_4) & \cos(180+\theta_4) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}
$$

$$
= \begin{vmatrix} -\cos\theta_4 & \sin\theta_4 & 0 & L_3 \\ 0 & 0 & -1 & 0 \\ -\sin\theta_4 & -\cos\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (6)
$$

Equation 3.7 shows the transformation matrix of the gripper to wrist joint.

$$
{}^{4}_{5}T = \begin{vmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (7)
$$

Hence, the forward Kinematics of the robotic arm from gripper to the base frame is given as:

$$
{}^{0}_{5}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T \quad (8)
$$

This implies that:

$$
x = \cos\theta_1[L_4\cos(\theta_2+\theta_4) - L_4\sin(\theta_2+\theta_3) - L_4\sin\theta_2] \quad (9)
$$

$$
y = \sin\theta_1[L_4\cos(\theta_2+\theta_3) - L_3\sin(\theta_2+\theta_3) - L_2\sin\theta_2] \quad (10)
$$

$$
z = \sin(\theta_2+\theta_4) - L_3\cos(\theta_2+\theta_3) + L_2\cos\theta_2] \quad (11)
$$

Where x, y, z are the Cartesian coordinates of the gripper.

5. Inverse Kinematics

The inverse kinematics of a robotic arm determines the joint angles and joint displacements of the robotic arm when the end effectors coordinates are known (Figure 4). The inverse kinematics is computationally expansive and generally takes a very long time in the real time control of manipulators.

The inverse kinematic problem can be resolved by two major approaches, namely:

i. The Geometric Solution Approach.
ii. The Algebraic Solution Approach.

The Geometric solution approach was adapted in this work by breaking down the spatial geometry of the manipulator into several plane geometry equations. The joint rotations are for base (θ_b), shoulder (θ_s), elbow (θ_e), and wrist (θ_w). To obtain the (x, y, z) coordinate at the top of the object, it implies that:

$$
rdist = U\sin\theta_s + L\sin\theta_{se} + G \quad (12)
$$

$$
z = B + U\cos\theta_s - L\sin\theta_{se} \quad (13)
$$

Where $\theta_{se} = 180 - (\theta_s + \theta_e)$ and U, L, and G are the lengths of the upper arm, lower arm and gripper links respectively, θ_w is the wrist angle.

$$
\frac{x}{y} = \frac{\cos\theta_1}{\sin\theta_1} \Longrightarrow \theta_1 = \tan^{-1}\frac{y}{x} \quad (14)
$$

$$
\theta_3 = \sin^{-1}\left[\frac{-b\pm\sqrt{b^2-4ac}}{2a}\right] \quad (15)
$$

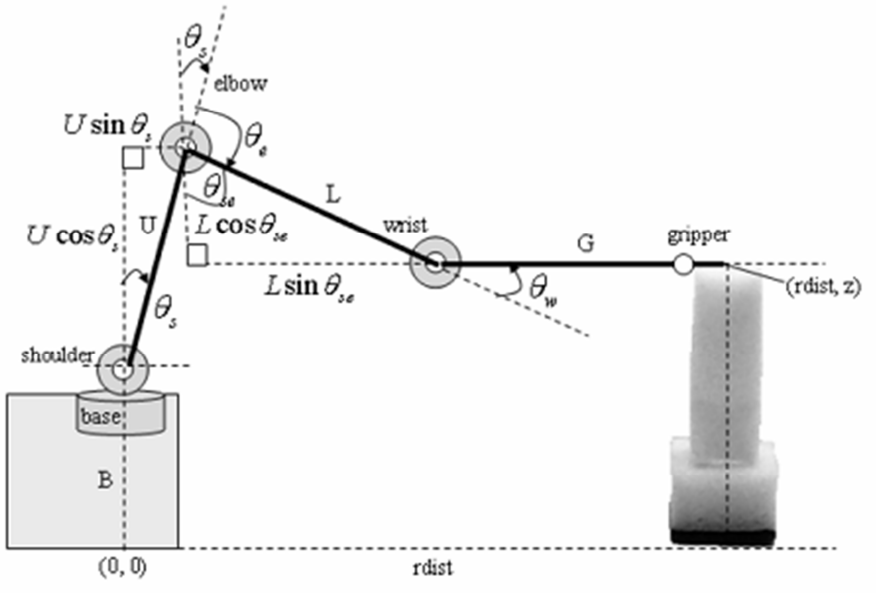

Figure 4. *Robotic Arm Geometry.*

$$
\text{where } b = 2kL_2L_4\ , a = 4L_2{}^2L_3{}^2 + 4kL_2{}^2L_4{}^2\ c = 4(k^2 - 4L_2{}^2L_3{}^2)
$$

$$
k = (x^2 + y^2 + z^2) - (L_2{}^2 + L_3{}^2 + L_4{}^2) \quad (16)
$$

And

$$
\theta_2 = \sin^{-1}\left[\frac{-e\pm\sqrt{e^2-4fg}}{2f}\right] \quad (17)
$$

$$
\text{where } f = k_1{}^2 + k_2{}^2, e = 2zk_2\ g = 4(z^2 - k_1{}^2)(k_1{}^2 + k_2{}^2)
$$

$$k_1 = L_4 \sin\theta_3 + L_3 \cos\theta_3 + L_2$$
$$k_1 = L_4 \cos\theta_3 - L_3 \sin\theta_3 \quad (18)$$

6. Software Design Concept

The software for the robotic arm was designed to present an efficient approach to the recognition of gestures for the control of a robotic arm. Earlier works related to robotic arm gesture control lacked basic design blocks and this prevented their design from being robust and adaptable by other software programmers.

Three main sections summarize the design approach adapted in this work, namely:

i. Gesture Recognition Engine.
ii. Arm Controller Engine and
iii. Graphical User Interface.

6.1. Gesture Recognition Engine

The gesture recognition engine was designed to process the gesture inputs from the user. The gesture recognition is actually a library written in C# code. When a gesture is being executed, the location of the joints with respect to time and distance is of key importance to the recognition engine.

6.2. Gesture Tracking Algorithm

The gesture tracking algorithm is at the heart of this project, the algorithm is responsible for tracking the required skeletal joint data of the human of interest. To confirm that a gesture has being correctly executed, a set of defined rules must be established to confirm that the gestures are valid and this is shown in Figure 5.

Three phases have being incorporated in the algorithm design to ensure that tracked gestures are executed according to preset standards. Due to the complexity of human gestures performed in a discussion context, it is imperative that all the incorporated detecting phases are linked to each other. This is to ensure that inter phase communication is not breached [9].

Stage one of the algorithm determines if a gesture start mode has being initiated. To establish the initialization, it checks if the joints location falls within set limits. The limits are dependent on joint markers such as the spine, shoulder joint etc. If successfully executed, the flow of command swings to stage two.

Stage two monitors the joints to ensure that the gestures still lie within the discussion context of the specified gesture command (e.g. Swipe Left, Swipe Right etc.). In an event the gesture fall out of place the command returns to stage one to await the next gesture command.

Stage three checks to confirm that the gesture has not violated any of the constraints set by the previous stages and it also confirms that the gesture command was executed within 15 frames. At this final stage, an Event Handler is fired to trigger the Arm Controller Engine into action. The Arm controller Engine verifies the gesture type and executes the relevant command which is transmitted to the robotic arm via serial communication.

6.3. Arm Controller Engine

The arm controller engine is responsible for controlling the robotic arm. All the algorithms that define the forward and inverse kinematics of the robotic arm are stored in the arm controller library. The serial communication protocol of the robotic arm is also part of the arm controller engine.

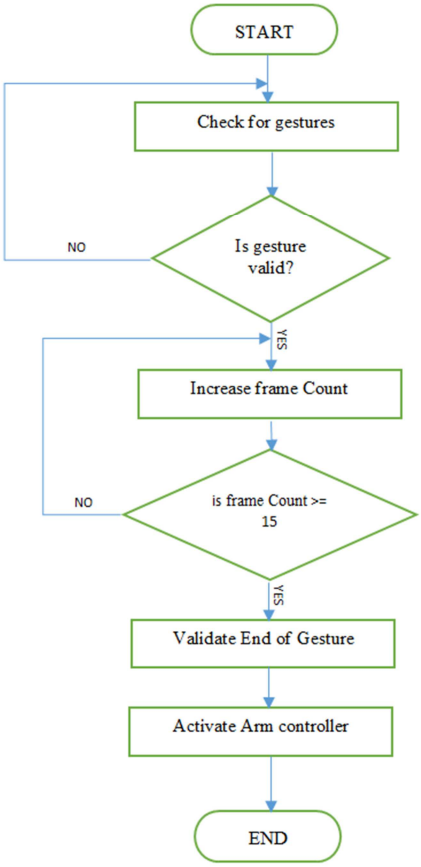

Figure 5. *Flow Chart of Gesture Detection System.*

The gestures that are processed by the gesture recognition engine generates set of commands that are passed to the arm controller. The various directions include:

i. Swipe To Right
ii. Swipe To Left
iii. Arm Forward
iv. Arm Backward
v. Shoulder Up
vi. Shoulder Down
vii. Arm Trigger

7. Hardware Communication

In order to achieve a smooth operation of the entire experiment, the hardware interaction between devices must be taken into consideration. Figure 6 depicts four stages that summaries the data flow of the setup. Human skeletal positional data was acquired at a set value of 15fps by a

Kinect XBOX360® sensor (Model no: 1414), powered externally. The skeletal data acquired is converted to binary values and transmitted via a -5v to +5v powered USB cable to the computer system. The acquired data stream from the Kinect sensor matches the USB port of the computer system powered by the Kinect USB "plug n play" preinstalled driver software.

The connection between the computer and the robotic arm is via a USB-Serial converter cable (with installed prolific USB-Serial driver). The data transmission is achieved by transmitting signals generated from the computer system via the USB port to the serial port of the controller board of the robotic arm. The serial port data is usually sent as a packet at 20ms intervals with 8 bit word, start, stop, parity bits. The Start bit is usually between +5v to +7v followed by data bits and finished by stop bit between -5v and -7v [8].

8. Results

Table 1 shows a set of data that depicts the respective values of the robotic arm for fourteen (14) tested routines. The table is divided into five (5) major sections; the xyz coordinates, expected joint angles, actual joint angles, the joint angle errors and error calculations.

The xyz coordinates of the tracked users' arm was imputed into the inverse kinematics equation so as to provide the expected joint location of the respective joints (shoulder, base and elbow). The acquired values were recorded and compared with the xyz coordinates of the arm acquired in real time for the same xyz coordinates. Slight changes were recorded in the real time computed values as compared to the computed values. This inconsistences were expected because the xyz value acquired in real time are subject to various factors such as light visibility, line of sight , poor gesture event handling, motor rotational speed, serial communication and the multi-threaded response of the robotic arm controller engine [10]. A set of graphical diagrams (Figures 7a, b, c) show expected (dotted lines) and actual (solid lines) for each monitored joint (base, shoulder and elbow joints).

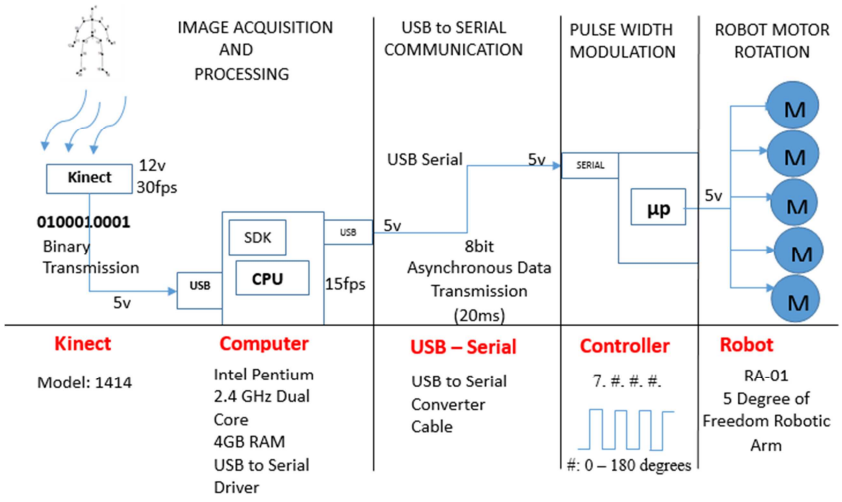

Figure 6. *Hardware Interaction.*

Table 1. *Error Test for Cartesian Position of the Robotic Base from 90 t0 180 degrees.*

Hand Cartesian Position (mm)			Expected Joint Angles (Degrees)			Actual Joint Angles (Degrees)			Error in Joint Angles (Degrees)		
X	Y	Z	Base	Shoulder	Elbow	Base	Shoulder	Elbow	Base	Shoulder	Elbow
0.3	0.1	1.3	72	19	90	68	22	80	4	3	10
0.4	0.1	1.3	77	20	89	57	18	90	20	2	1
0.1	0.2	0.8	63	39	-76	65	42	-69	2	3	7
0.1	0	0.7	90	5	-90	88	0	-83	2	5	7
0.5	0.3	1.1	59	60	96	54	45	92	5	15	4
0.5	0.3	1.6	59	58	84	60	60	81	1	2	3
0.2	0.4	0.9	26	75	-80	22	78	-75	4	3	5
0.2	0.5	1.0	21	90	10	24	90	10	3	0	0
-0.1	0.2	1.2	-26	40	38	-30	38	35	4	2	3
-0.6	-0.2	1.3	-90	-40	62	-89	-35	58	1	5	4
-0.2	1.4	1.6	-18	90	90	-20	70	84	2	6	6
-0.3	0.4	0.9	-36	39	-80	-32	40	-75	4	1	5
-0.1	0.3	1.1	-18	60	15	-19	58	20	1	2	5
-0.5	-0.2	0.4	-90	-38	-90	-90	40	-90	0	2	0
						Mean Error (mm)			3.79	3.64	4.30
Overall Mean Error(mm) = 3.9 Variance of Error(mm) = 14.1 Standard Dev. of Error(mm) = 3.8											

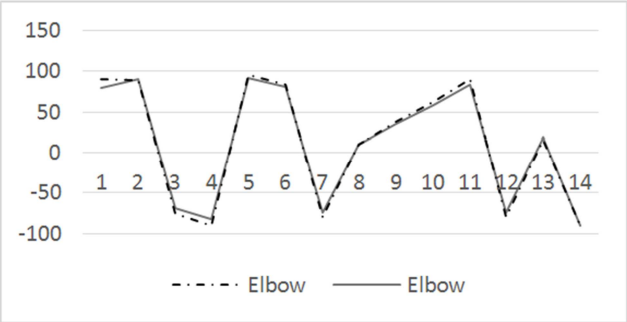

Figure 7a. *Elbow Joint gesture response.*

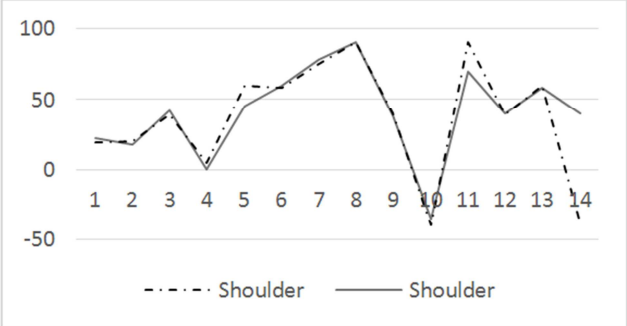

Figure 7b. *Shoulder Joint gesture response.*

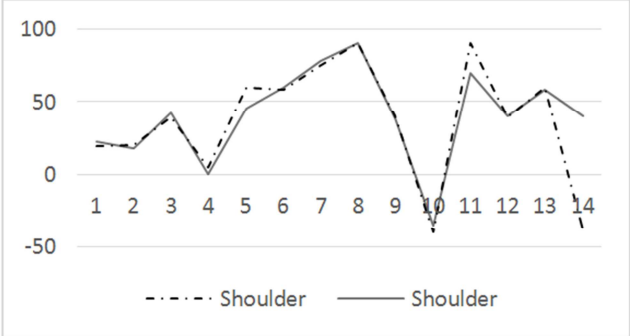

Figure 7c. *Base Joint gesture response.*

9. Conclusion

A few glitches were experienced with the gesture recognition engine, and this was caused by an overlapping of detected gestures and the lighting of the room in which the control routines was tested. The system showcased the importance of gesture control in human machine interaction. However, further work needs to be done to apply machine learning algorithms to expand the flexibility of the gesture detection system.

References

[1] Cannan, J. & Huosheng Ho, 2010. Human-Machine Interaction (HMI): A Survey, Essex: School of Computer Science & Electronic Engineering. [Online]http://cswww.essex.ac.uk/staff/hhu/Papers/CES-508%20HMI-Survey.pdf

[2] Johannsen, G., (2007). Human Machine Interaction. Control Systems,Robotics and Automation, Volume XXI, p. 1.

[3] Steven J, L., Shawn, S. & Neal, C., (2014). Natural User Interface for Robot Task Assignment [Online] http://hci.cs.wisc.edu/workshops/RSS2014/wp-content/uploads/2013/12/ levine2014 natural.pdf

[4] Shobhitha, A. J., Jegan, R. & Melwin, A., (2013). OWI-535 EDGE Robotic Arm Control Using ElectroMyoGram (EMG) Signals. International Journal of Innovative Technology and Exploring Engineering (IJITEE), II(6), pp. 282 - 286.

[5] Olawale, J., Oludele, A., Ayodele, A. & Miko, N., (2007). Development of a Microcontroller Based Robotic Arm. American Journalof Engineering Research (AJER), IV(2), pp. 8.

[6] Jana, A., (2012). Kinect for Window SDK programming guide. 1st ed. Birmingham: Pack Publishing Press. [Online] http://www.pactpub.com/game-development/kinect-windows-sdk-programming-guide

[7] Clement, G. and Massimo, F., (2013). *Kinect in Motion - Audio and Visual Tracking by Example.* 1st ed. Birmingham: Packt Publishing Press. [Online] http://www.pactpub.com/game-development/kinect-motion-%E2%80%93-audio-and-visual-tracking-example

[8] Akinwale, O. B., L. O. Kehinde, K. P. Ayodele, A. M. Jubril, O. P. Jonah, O. Ilori, and X. Chen, " (2009). A La*bVIEW-based on-line robotic arm for students' laboratory*", *2009 ASEE Annual Conference & Exposition*, Austin, Texas, ASEE, pp. Paper 2009-1179.

[9] Catuhe, D., (2012). Programming with the Kinect for Windows. Redmond,Washington: Microsoft Press. [Online]

[10] https://books.google.com/books/about/Programming_with_the_Kinect_for_Windows.html ?id=66kgXfDKdYwC

[11] Microsoft, (2015). Chapter 18. Threading and Synchronization. [Online] msdn.microsoft.com/enus/library/ff652496(d=default,l=enus,v=orm.10).aspx[Accessed 9 July 2015].