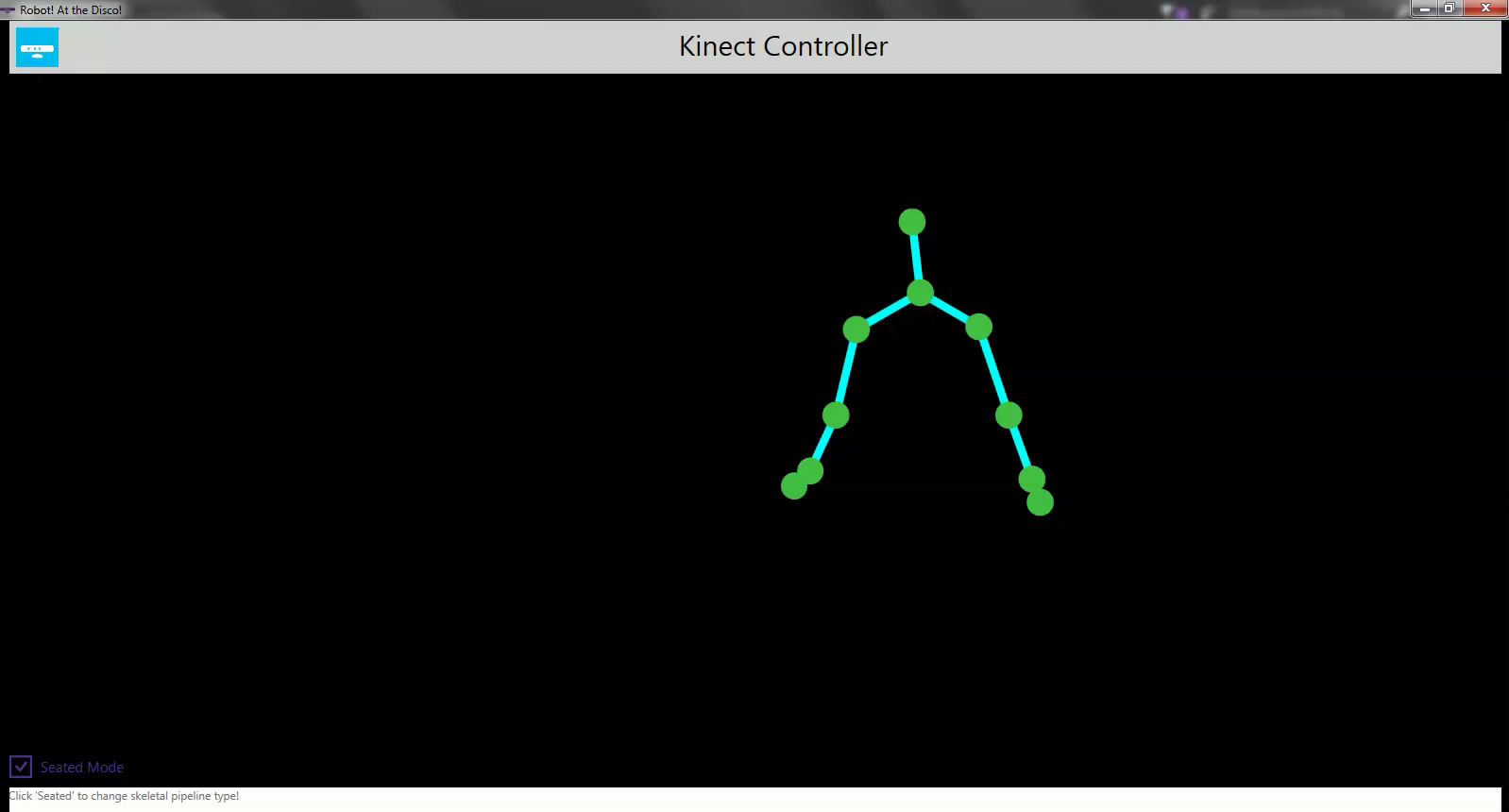
**Robot! at THE dISCO**

By: Eric Butt & Clarisse Vamos

EEL 4660 – Robotic Systems

Dr. Gita Sukthankar



**INTRODUCTION**

Robot! At the Disco is a gesture-controlled, dancing companion robot that uses generated human gesture commands to control its movement patterns. The goal of the project was to create a robotic dance partner so that no one ever has to dance alone. There are many examples of research that has been conducted into robot dancing and the mimicry of human movement. One such case is “Automation” by Amy LaViers (Georgia Tech), a Nao robot that was taught styles of human movement in an effort to move away from the often-stiff movement style exhibited by dancing robots [1]. More in the spirit of Robot! At the Disco, Lena Ting’s Neuromechanics lab (Emory University and Georgia Tech) carried out research in 2015 into partnered robotic dancing using haptic interaction from a human partner. It attempted to model the haptic input human dance partners often use to convey future intent and recreate a partnered dance with a robot that could be used in physical therapy or rehabilitation [2].

**METHOD**

The methodology for developing the Robot! At the Disco on the Elegoo platform will be discussed below. Specifically, there are three major functions that will be discussed: Kinect gesture control, IR control, and robotic movement.

Kinect Gesture Control

For this project, we defined our ‘robot control gestures’ as the XY-coordinate location of the hands (either right, left, or both) in relation to the XY-coordinate locations of certain other bodily joints (the head, shoulders, and/or elbows). This XY-coordinate data is received inherently through the Xbox Kinect sensor using its built in 3-D Depth sensors. Within the Kinect, these data-points are automatically combined to form a mathematical representation of the human skeleton, comprised of 20 distinct joints that each carry their own XYZ-coordinate position. This model is then passed onto the C# code, and the relevant joints positions for each type of movement are compared to one another. When certain conditions are met by the relation of these points, the corresponding signal to that movement is sent through the pre-defined serial port connection and onto the Arduino board. The visual depiction of the Kinect data that is generated by our C# code is shown to the right.

The conditions for each type of robotic movement are as follows (in the order of precedence in which they appear in the C# code):

A screenshot of the generated Kinect data, with the human controller currently at rest.

|  |  |
| --- | --- |
| **Conditions** | **Movement Pattern** |
| Right and Left Hand both above the Head | Zig Zag |
| Right Hand within the right elbow and above the Head | Circle Right |
| Left Hand within the left elbow and above the Head | Circle Left |
| Right Hand just above the Left Shoulder (and on the left side of the body) | Switch to IR command system |
| Right Hand within the body (inside the Right Shoulder) | Turn Right |
| Left Hand within the body (inside of the Left Shoulder) | Turn Left |
| Right Hand above the Left Elbow | Go Forward |
| Left Hand above the Right Elbow | Go Backwards |
| Anything Else | Stop |

A sample snippet of the relevant code is as follows, where “Z” is the serial code for Zig-Zag movement in the corresponding Arduino code.

// ZIG-ZAG Command

if( handRight.Position.Y > head.Position.Y &&

handLeft.Position.Y > head.Position.Y) {

serialPort.WriteLine(“Z”);

}

IR Control

The use of IR, or Infrared Ray, control can be activated by using the gesture command for ‘Right Hand just above the Left Shoulder.’ Once activated, specified keys on the IR remote can be used to send movement commands to the platform. The process employed to achieve this control is as follow:

1. The IR transmitter on the remote outputs a light signal encoded with the key of the remote button pressed. This is accomplished using NEC infrared transmission protocol, which uses pulses to transmit the logical bits that create the key.
2. An IR receiver on the platform captures the light signal and then translates it into an electric signal. The signal is then amplified, filtered, and restored to the original encoded key using gain controlling.
3. Arduino code written for the platform keeps track of the received key and attempts to match the key to defined number and symbol keys. The defined ok and arrow keys can be seen below:

#define UP 16736925

#define DOWN 16754775

#define OK 16712445

#define LEFT 16720605

#define RIGHT 16761405

1. Once the signal is matched to a key, the command associated with the key in the code is performed by Robot! At the Disco. A chart of the IR commands can be seen below:

|  |  |
| --- | --- |
| **Button Key** | **Command** |
| OK | Stop |
| UP | Go Forward |
| DOWN | Go Backward |
| LEFT | Turn Left |
| RIGHT | Turn Right |
| ONE | Circle Left |
| TWO | Circle Right |
| THREE | Zig Zag |
| STAR | Return to Gesture Control |

Robotic Movement

After receiving the input command from the serial port (from either the Kinect via Bluetooth or the IR remote), the robot’s Arduino code translates the signal and sends it through a simple *if-else* block to determine which movement pattern the robot should adhere to. Depending on which pattern is indicated, a differing amount of power is sent to the corresponding pins of the four motors, which translates into the movement of the robotic platform as a whole. A sample of the ‘Stop’ command is shown below:

Void \_mStop() {

digitalWrite(ENA,LOW);

digitalWrite(ENB,LOW);

Serial.println(“STOP!”);

}

To see a demonstration video of the project, please visit the URL listed below to view a quick demo.

www.youtube.com/watch?v=sZbnnA306HY&t=2s

**DISCUSSION**

In terms of gesture control, the platform was relatively successful in the identification of joint positions and determining the mathematical relationship between them in order to send out the movement commands to the robot. However, the Xbox Kinect tends to struggle with keeping track of the absolute locations of the various joints. Problems in precise detection often occurred when two or more joints came into close proximity with one another. Thus, the misalignment of joints while the Kinect tries to figure out the discrepancies often sends erroneous data points to the *­if-else* block in the C# code, which then produces an undesired movement pattern based off the false input data. To counter this, the use of broad and obvious movements between gesture commands was able to eliminate most of the identification issues and provided the best overall results for reliable control.

Overall, with our adjusted broad-scope gestures definitions, our system was able to accurately identify the correct motion for the core movement commands near-100% of the time, with the only difficulties arising in the purposefully-difficult ‘Switch to IR Control’ command which involved placing the right hand in a small area between the left shoulder and the head. The only difficulties with gesture detection arose when transitioning from one ‘control area’ to another, with the Kinect processing the various gestures *too well*. For example, the ‘Circle Right’ command involves having your right hand above your head. However, if the human controller wanted the robot to go directly from a stopped position into the ‘Circle Right’ movement pattern, it is likely that their right hand would briefly fall into the control area of ‘Go Forward’ before the hand physically reaches above the head. Because of how quickly the Kinect can process this change in joint positions, it would tell the robot to first ‘Go Forward’ before telling it to ‘Circle Right’, which is what not was originally anticipated. This would often cause the robot to accidently run into a wall or other obstacle as the human controller scrambles and tries to account for the unexpected new movement pattern the robot went through.

In terms of IR control, the relation of serial remote input to robotic movement was relatively simple to implement successfully. However, problems did occur when the platform moved too far away from the IR remote as the system could no longer receive the IR input being provided. Therefore, it is necessary for the Robot! At the Disco to operate within a limited radius around the remote.

In terms of overall motion, the platform was successful in creating movement based on both Kinect gestures and remote IR commands. Our system was able to quickly respond to input and send the signal along to carry out the proper commands. During testing, the robot did experience some trouble with maintaining straight forward and backward movement, identifying a need to go back and slightly adjust the servo controls for the platform’s wheel base. Turning left and right ended up being quite a struggle for the Elegoo platform under our code, as it often resorted to a sort of “hopping” motion along the ground to change its orientation rather than smoothly rotate around.

**CONCLUSION**

Overall, this project was successful in their goal to create a gesture-driven companion dancing robot in Robot! At the Disco. However, given the limited development timeframe, the team was not able to fully incorporate all if the mechanical tools present on the Elegoo platform. For example, there is the future possibility of automatic obstacle avoidance using the Elegoo platform’s ultrasonic sensor. This would allow the robot to avoid collisions with obstacles while following the input gestures and dancing around its environment as intended.

Further development of the gestures and platform’s gesture recognition with the Kinect would significantly improve the robot’s operation. The current gesture recognition algorithm used by the platform uses the XY locations of simple joints to create and track ‘gestures’. However, due to the difficulties the Kinect has in absolute positional accuracy for joints and the problem of accidentally entering the control areas for other gesture commands, it would be much more effective to map the movement commands to actual gestures, such as the wave of a hand. By capturing a series of positional data-points over time, the system would be able to maintain a much better degree of control over the robot’s motion, while at the same time, possess a much more robust user interface that would make controlling the robot a more effective and enjoyable experience overall.

The addition of music and music control to the Robot! At the Disco platform would also help further achieve the project’s intent to create a robotic dancing partner. This would be implemented using Bluetooth/gesture control to change songs that would be played on a speaker powered on the robotic platform itself. An SD card would hold a number of pre-selected songs, and a gesture command to change songs would allow the robot’s dance partner to provide a source of music. In terms of code, the implementation of music would only require a few lines, but an overhaul to our robotic platform itself would be required to physically get the SD card and/or LED lights to fit on the platform.

In conclusion, Robot! At the Disco is a fun, simple dancing robot that would be a fine dance partner to any interested party. The use of its gesture detection is somewhat simplistic in its current state, but with additional time and development, the system has ample capacity to advance and improve upon itself.

**REFERENCES**

[1] “Georgia Institute of Technology,” Engineering Style of Dance for Robots and People | Institute for Robotics & Intelligent Machines at Georgia Tech. [Online]. Available: http://www.robotics.gatech.edu/newsroom/features/dance. [Accessed: 01-Dec-2016].

[2] L. H. Ting, “Evaluation by expert dancers of a robot that performs partnered stepping via haptic interaction,” Coulter Department of Biomedical Engineering at Georgia Tech and Emory University. [Online]. Available: https://bme.gatech.edu/bme/evaluation-expert-dancers-robot-performs-partnered-stepping-haptic-interaction. [Accessed: 01-Dec-2016].

**SOURCE CODE**

https://github.com/ebutt/DiscoRobo