Robot Control Via Head Gesture Based Interface

Vishnu Gopal Rajan
University of Texas at Arlington
Vishnu.gopalrajan@mavs.uta.edu

Rama Krishna Reddy Indurthy
University of Texas at Arlington
ramakrishnaredd.indurthy@mavs.uta.edu

ABSTRACT

Researching and developing assistive robots for paraplegic users and users with severe motor impairment disabilities is a difficult task, considering that such an interface has to be hands free. This paper focuses on developing a head gesture interface for a robot to recognize simple head gestures and interpret those signals to perform translation operations with robot arm. A state diagram is developed to navigate through different states of control.

CCS CONCEPTS

• • Human-centered computing → Human-computer interaction (HCI) → Interaction techniques → Gestural input; • Human centered computing → Human computer interaction (HCI) → Interaction paradigms → Graphical user interfaces; [1]

KEYWORDS

Head gesture recognition, robot control, assistive robotics, Human-Robot Interface [1]

ACM Reference format:

FirstName Surname, FirstName Surname and FirstName Surname. 2018. Insert Your Title Here: Insert Subtitle Here. In *Proceedings of ACM Woodstock conference (WOODSTOCK'18). ACM, New York, NY, USA, 2 pages.* https://doi.org/10.1145/1234567890

1 INTRODUCTION

People with disabilities have difficult time performing day to day tasks let alone competing in tradition work life. Namely, for people with motor impairments, unemployment is a serious problem. It is estimated by the National Spinal Cord Injury Statistical Center (NSCISC) that there are 12000 new cases of spinal cord injury (SCI) every year in the USA [2]. At the time of their injury, almost half of persons with SCI were employed and by 20 years post-injury only one third is employed.

Robots with assistive capabilities can help change that. They help people with paraplegia, tetraplegia or severe motor impairments to integrate in to work life and return some semblance of control over their destiny and power to their lives. There has been a increase in interest in assistive robots over the past few decades. Assistive robots can give them independence at workplace, which is proved through systems like FRIEND [4], wheelchair mounted robotic manipulator, which performs robotic manipulation of books and supports a person with a severe disability to work in a library.

Studies have shown that people with disabilities prefer to control the assistive devices directly as much as possible as it grants freedom, they could only dream of prior to that. [5]. Tasks like drinking, eating and preparing meals are very important for this demographic. Assistive robots can also be used for other tasks, such as manipulating objects, personal hygiene, dressing, and opening doors [1]. It is not easy for people with severe motor impairments to control robots.

However, the control of robotic manipulator by a person with quadriplegia is a very challenging task. Several "hands-free" interfaces have been proposed to directly control robotic manipulators, such as eye-movement gesture-based [6], Brain-computer interface (BCI) [7] and head motion [8]. In this paper, a head gesture-based framework for hands-free robot control is presented. The proposed framework works on the principle of recognizing different head gestures such as up, down, left and right and then mapping those gestures to movements of robot arm in Cartesian space. The proposed framework aims to enable people with disability (end-users) to control the robotic arm, which in the future could be developed into a full-fledged system to manipulate objects and perform highly complex tasks.

Figure 1**: Kinova's Robot Arm turns the wheelchair into an assistive Robot [15]**

2 RELATED WORK

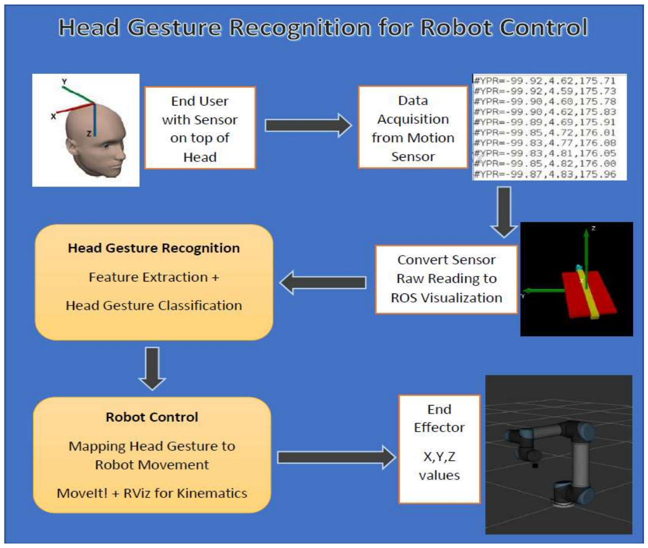
Robots can be controlled with head gestures by two ways: motion sensor based and vision based. Vision based approach is where the sensor cameras can be used to track the head movement and in turn control the robot movement. Motion sensor based Inertial Measurement Units (IMU) can be used to measure angle of orientation, acceleration, angular velocity of head which in turn can be used to control the robot arm movement.

In this paper, Inertial Measurement Unit (IMU) is used for controlling the robot. Analytical approach has been used in the work [9] for head gesture recognition using a motion sensor. In this method, the algorithm used fits a Gaussian function to the linear acceleration signal and based on simple thresholds and the accuracy of gesture recognition is 87.56% [1]. Recognized gestures, Left-Gesture (LG), Up-Gesture (UG), Down-Gesture (DG) and Right-Gesture (RG) are used as commands to move the robot arm along the cartesian axes. In [10] a usability study was done with 30 test subjects which included six tetraplegics and 24 able-bodied people. In [10], a 3 Degrees of Freedom microcontroller-based automated system was presented, which enables wheelchair users to control the motion of a wheelchair using the head movements. Data collected from the motion sensor was directly sent to wheelchair control unit using wireless technology. Those signals were converted to wheelchair control commands.

The main goal of this system is for the disabled user to control the robot with head gestures and perform tasks such as pick and place. They can control the robot arm to pick and place items from one location to another. This job can be done merely with just head gestures which enables them to secure these simple application jobs. This system is augmented by use of machine learning for feature based classification improving accuracy of gesture recognition when compared to the method in [9].

Figure 2: **Robot arm enabling a disabled person perform daily tasks [16]**

3 HEAD GESTURE RECOGNITION FOR ROBOT CONTROL


Head Gesture Recognition for Robot Control
Y
X
Z
End User with Sensor on top of Head
Data Acquisition from Motion Sensor
Convert Sensor Raw Reading to ROS Visualization
Head Gesture Recognition
Feature Extraction +
Head Gesture Classification
Robot Control
Mapping Head Gesture to Robot Movement
MoveIt! + RViz for Kinematics
End Effector
X,Y,Z values

```
#YPR=-99.92,4.62,175.71
#YPR=-99.92,4.59,175.73
#YPR=-99.90,4.60,175.78
#YPR=-99.90,4.62,175.83
#YPR=-99.89,4.69,175.91
#YPR=-99.85,4.72,176.01
#YPR=-99.83,4.77,176.08
#YPR=-99.83,4.81,176.05
#YPR=-99.85,4.82,176.00
#YPR=-99.87,4.83,175.96
```

Figure 3: **Head Gesture Recognition for Robot Control conceptual diagram**

The overview of the proposed method of head gesture recognition for controlling robots is illustrated in fig 3. It has three main modules. They are Motion Data Acquisition, Head Gesture Recognition and Robot Control. To test this system, we used the robot arm simulation of Universal Robot. Specifically, we used the simulation of UR5. This simulation is done in Gazebo using ROS framework. The UR5 is a lightweight, adaptable collaborative industrial robot. It can perform medium-duty applications with incredible flexibility and maneuverability. The UR5e is built for seamless and effortless integration into a wide variety of platforms [14]. It has rotation capabilities from +360 to -360. It has six rotating joints giving it six degrees of freedom.

This UR5 robot is a 6DOF robot arm able to perform various tasks like a human arm. The gripper at the end of the robot arm helps the user to grab items and also move it from one place to another.

In order to perform these tasks we made use of head gestures. The system involves a module called Head Gesture recognition which extracts certain head movement features and classifies them into respective head gestures.

3.1 Data Acquisition from Motion Sensor

The motion sensor used with UR5 robot in this paper is Spark Fun 9Dof Razor IMU is used. It is an Inertial Measurement Unit. The sensor comes with 3-axis gyroscope, accelerometer and a geomagnetic sensor. Spark Fun 9Dof Razor IMU has a on board data fusion to calculate the rotation in Euler angles and quaternion [1]. The sensor is worn on head using a hairband. Sensor's orientation is shown in Fig 3 and Fig 4. This IMU sensor return raw values which are converted to Yaw, Roll and Pitch which can be used to classify head gestures. The Figure 5 shows the reading of the sensor after converting them from raw readings to Yaw, Roll and Pitch using ROS Packages and Visualization tools.

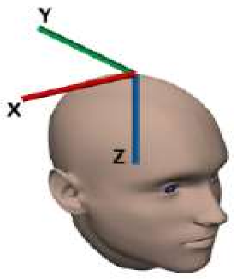
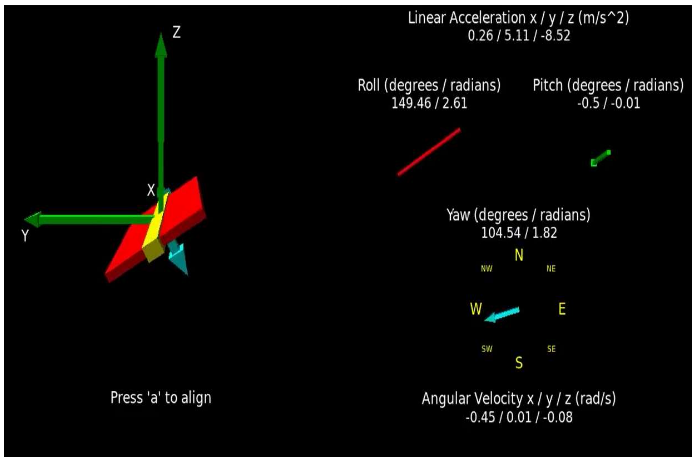


Figure 4: Sensor Orientation [1]



Linear Acceleration x / y / z (m/s^2)
0.26 / 5.11 / -8.52

Roll (degrees / radians)
149.46 / 2.61

Pitch (degrees / radians)
-0.5 / -0.01

Yaw (degrees / radians)
104.54 / 1.82

Angular Velocity x / y / z (rad/s)
-0.45 / 0.01 / -0.08

Figure 5: Ros Visualization showing Sensor dat

3.2 Head Gesture Recognition

UR5 Robot is controlled using four basic head gestures. They are Left-Gesture (LG), Up-Gesture (UG), Down-Gesture (DG) and Right-Gesture (RG) as shown in fig.6. These Gestures are recognized by the python code which in real time monitors the sensor readings. Each Gesture has a certain feature to it and these respective features are extracted in order to classify them as a gesture. Upon classifying these gestures we can map them to perform the required tasks.

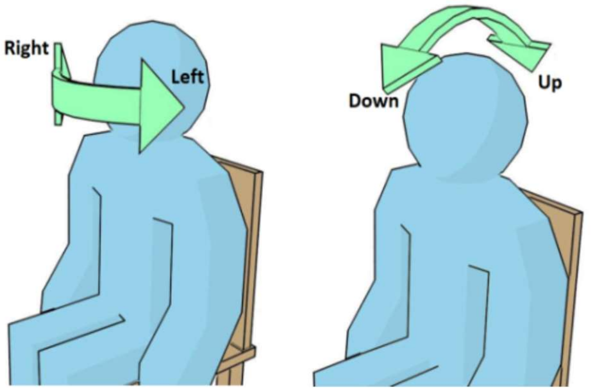


Figure 6: **The four head gestures classified and recognized by the system.** [1]

To collect the data in order to classify these gestures we use the IMU sensor attached to a headband which is worn by the user on top of the head. This sensor has to lie flat on top of the head and a slight tilt can also cause bad readings which cause the gesture not being recognized.

Since the range of head motion differs from person to person, we need to calibrate this based on the user. Thus, the head gesture data is collected from test subjects. The test subjects performed four basic head gestures i.e, Left-Gesture (LG), Up-Gesture (UG), Down-Gesture (DG) and Right-Gesture (RG). The data collected has 3-d angular velocity and 3-d acceleration. From this data maximum and minimum values of pitch and yaw are collected. (Fig 7)

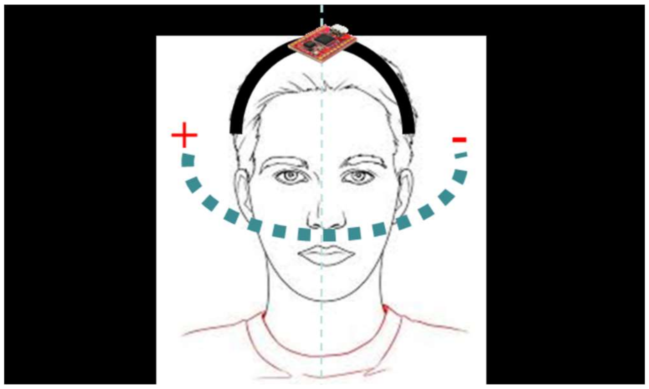


Figure 7: **Minimum and Maximum values of Pitch and Yaw readings [1]**

This pitch and yaw data is used to set the range and recognize Left-Gesture (LG), Up-Gesture (UG), Down-Gesture (DG) and Right-Gesture (RG).

3.3 Robot Control

The robot control involves the user controlling the simulated UR5 robot arm. Before the user can control the robot, we have made sure the robot is back to its default position. The python User Interface allows user to navigate through the state diagram [1].

The interface informs the user the Left-Gesture (LG) is used to decrease the value and Right-Gesture (RG) is used to increase the value. The Up-Gesture (UG), Down-Gesture (DG) is used to move from one state to another. These gestures can be used to move through the state diagram as shown in Fig 8, to select the coordinates in the cartesian space along which the user wants the robot arm to move. But first to work with the robot, calibration has to be done. Calibration is used to measure the full range of head motion around the z-axis which is yaw. This is done to map the speed of the robotic arm with the range of head rotation around z-axis (yaw angle). Maximum speed along the negative axis is defined as range of motion in the left direction. Maximum speed along the positive axis is defined as range of motion in the right direction.

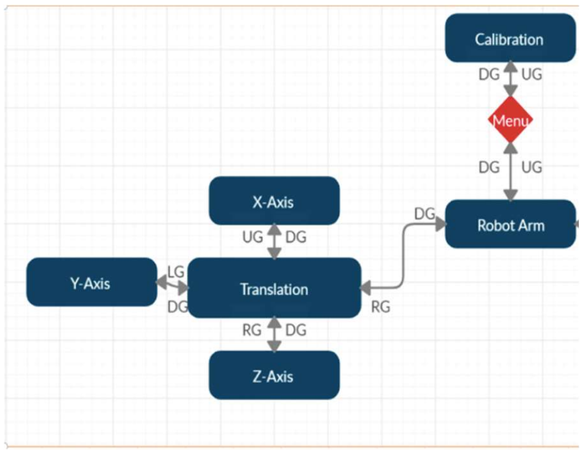
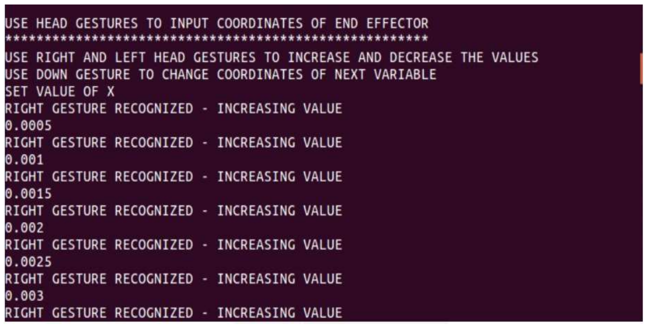


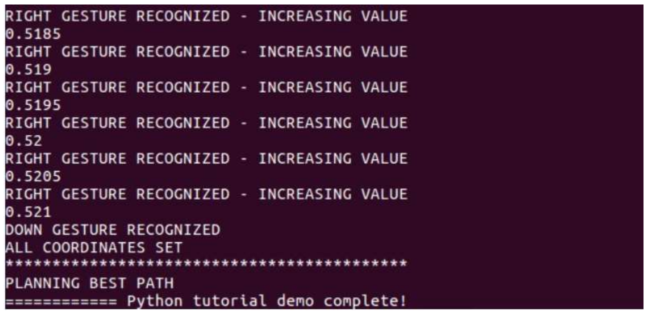
Figure 8: **State Diagram**

Different states in the state diagram are Calibration, Robot Arm, Translation, X-axis, Y-axis, Z-axis. In the future, a state for Rotation can also be added to increase the functionality of the system. But since we are using MoveIt! Kinematics plugin the rotation is automatically calculated depending on the position it needs to go to.

The python interface guides the user when to perform the gesture and when a certain gesture is recognized. This helps the user stay on track and also see the change in coordinates upon performing the gestures. After setting the coordinate values the python interface informs the user that all the coordinates are set successfully. Once this is done the Moveit! And Rviz plan the path to the given coordinate, if its feasible. If this path cannot be reached then it gives an error stating that it cannot be reached. Figure 9, Figure 10 shows the Python interface.



```
USE HEAD GESTURES TO INPUT COORDINATES OF END EFFECTOR
*****************************************************
USE RIGHT AND LEFT HEAD GESTURES TO INCREASE AND DECREASE THE VALUES
USE DOWN GESTURE TO CHANGE COORDINATES OF NEXT VARIABLE
SET VALUE OF X
RIGHT GESTURE RECOGNIZED - INCREASING VALUE
0.0005
RIGHT GESTURE RECOGNIZED - INCREASING VALUE
0.001
RIGHT GESTURE RECOGNIZED - INCREASING VALUE
0.0015
RIGHT GESTURE RECOGNIZED - INCREASING VALUE
0.002
RIGHT GESTURE RECOGNIZED - INCREASING VALUE
0.0025
RIGHT GESTURE RECOGNIZED - INCREASING VALUE
0.003
RIGHT GESTURE RECOGNIZED - INCREASING VALUE
```



```
RIGHT GESTURE RECOGNIZED - INCREASING VALUE
0.5185
RIGHT GESTURE RECOGNIZED - INCREASING VALUE
0.519
RIGHT GESTURE RECOGNIZED - INCREASING VALUE
0.5195
RIGHT GESTURE RECOGNIZED - INCREASING VALUE
0.52
RIGHT GESTURE RECOGNIZED - INCREASING VALUE
0.5205
RIGHT GESTURE RECOGNIZED - INCREASING VALUE
0.521
DOWN GESTURE RECOGNIZED
ALL COORDINATES SET
*******************************************
PLANNING BEST PATH
=========== Python tutorial demo complete!
```

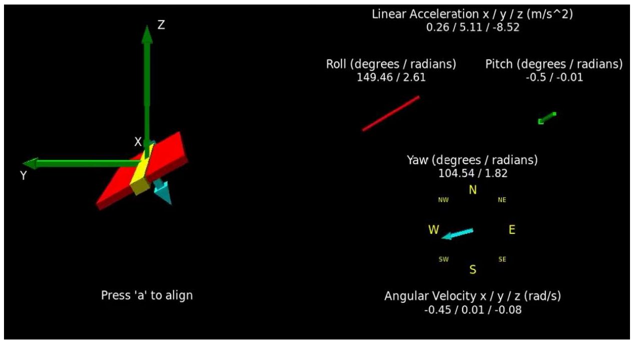
Figure 9 & 10: **Python interface guiding and recognizing gestures.**

4 PROPOSED METHOD

This project involves the active involvement of multiple systems and programs working simultaneously.

4.1 Spark Fun Razer IMU Sensor

The IMU sensor is used mainly to get the head movement values. These raw values are converted to Yaw, Roll and Pitch. This is done using the Arduino IDE. The respective sensor firmware is uploaded to the sensor which converts the raw readings into the required readings i.e Yaw, Roll and Pitch. Since we need to have all the programs on the same ROS Kinetic Environment we also installed the required ROS Kinetic packages for the IMU Sensor. This enabled the sensor to be visualized in real time allowing us to gather data to the ROS Environment much easily. The Figure 11 shows the ROS Visualization on the sensor along with the data its providing.

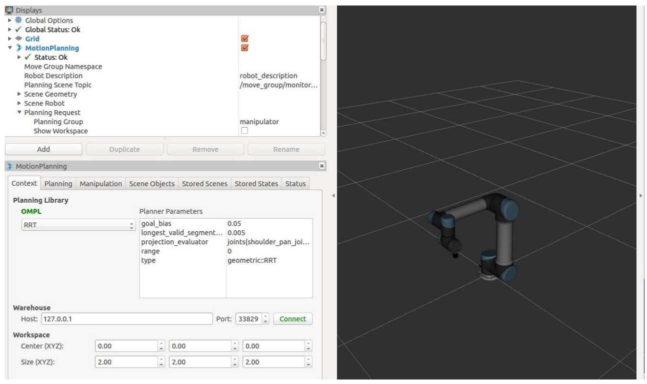


```
Linear Acceleration x / y / z (m/s^2)
0.26 / 5.11 / -8.52
Roll (degrees / radians)    Pitch (degrees / radians)
149.46 / 2.61               -0.5 / -0.01
Yaw (degrees / radians)
104.54 / 1.82
Press 'a' to align
Angular Velocity x / y / z (rad/s)
-0.45 / 0.01 / -0.08
```

Figure 11**: IMU sensor ROS Visualization**

4.2 Simulated UR5 Robot arm

To get this simulated Robot arm running we need ROS Kinetic support. Thus this could only be installed on Ubuntu 16.04. Upon installing ROS Kinetic other dependencies for this simulation included installation of Gazebo, MoveIt! And RViz. These tools help us simulate and also plan paths for movement of robot arms. Upon setting up of the required software we had to load the respective robot model (URDF) which in or case is the Universal Robot UR5. Upon loading the robot model, the robot arm joints and manipulators had to be set up. Upon setting this up, it creates a ROS MoveIt! Package with this robot arm. The Figure 12 shows the Robot arm simulation loaded on the MoveIt! Program.



```
Global Options
✓ Global Status: Ok
Grid                              ✓
MotionPlanning                    ✓
  ✓ Status: Ok
  Move Group Namespace
  Robot Description               robot_description
  Planning Scene Topic            /move_group/monitor...
  Scene Geometry
  Scene Robot
  Planning Request
    Planning Group                manipulator
    Show Workspace                ☐

Planning Library
  OMPL
  RRT
Planner Parameters
  goal_bias                   0.05
  longest_valid_segment...    0.005
  projection_evaluator        joints(shoulder_pan_joi...
  range                       0
  type                        geometric::RRT
Warehouse
  Host: 127.0.0.1                 Port: 33829
Workspace
  Center (XYZ):   0.00   0.00   0.00
  Size (XYZ):     2.00   2.00   2.00
```

Figure 12: **MoveIt! UR5 Simulation**

This Robot simulation shows the joints and the Fixed Base of the UR5 robot arm. As we can see in Figure 13 the simulated robot has 6 Degrees of Freedom (DOF). It can move in x, y and z negative and positive axes. RViz and MoveIt!'s KDL Kinematics Plugin was installed to help move the robot arm to the desired end effector position set using the head gestures. This Kinematics plugin helps plan the best path to the desired location.

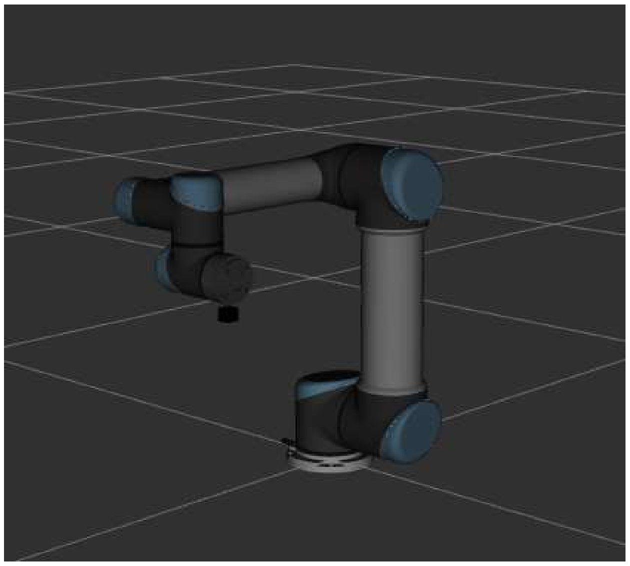


Figure 13: **Simulated UR5 Robot arm in Default state**

The Python interface code developed is used to communicate with this simulated robot. It brings the robot arm to default position before the user can set the desired location, they want it to move to. The Left Gesture (LG) is used to decrement the coordinate value and Right Gesture (RG) is used to increment the coordinate value. The python interface also shows the user the value of coordinate being manipulated as they perform the gesture. Upon completely setting all the coordinate values the RViz and MoveIt! Kinematics plugin plan the path for the robot arm to take to reach the destination coordinates. Upon finding a plan it executes it and reaches the destination. If the Kinematics cannot find a path then it informs the user that, the set coordinates cannot be reached.

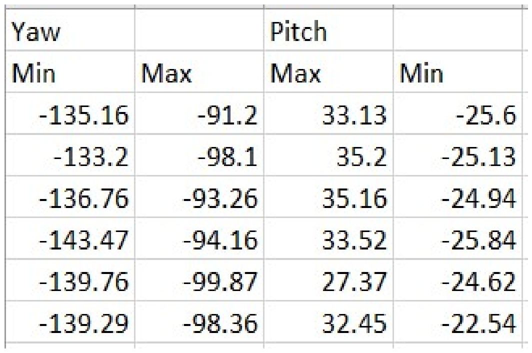
5 EXPERIMENTAL RESULTS

5.1 Accuracy of Head Gesture Recognition

The accuracy of the head gesture recognition is 100% since we have hard coded the threshold values beyond which the gestures will most definitely be recognized. There are few cases where certain anomalies are seen, where the accuracy is hindered. This happens in cases where the sensor is either not properly positioned on top of the head or if the gesture is not performed correctly.

5.2 Data collected from users

Since each individual has a different range of head motion we wanted to calculate the maximum and minimum yaw angles. This helps us set an average range while calibrating the sensor. The table below shows the values of yaw angles taken from different users.



Yaw		Pitch	
Min	Max	Max	Min
-135.16	-91.2	33.13	-25.6
-133.2	-98.1	35.2	-25.13
-136.76	-93.26	35.16	-24.94
-143.47	-94.16	33.52	-25.84
-139.76	-99.87	27.37	-24.62
-139.29	-98.36	32.45	-22.54

5.3 Robot performing task

The Robot simulation in Gazebo is connected to the Razor 9Dof IMU through ROS framework. To test the system, basic translation operations along x, y, z axes were performed. This was achieved using just four basic head gestures namely Up Gesture (UG), Down Gesture (DG), Left Gesture (LG), Right Gesture (RG). These gestures used to navigate through different states such as calibration, Robot arm, Translation, X-axis, Y-axis, Z-axis. When the python interface prompted the user to set the values of X, Y and Z coordinates the users had to either perform Left Gesture (LG) to decrement the value or Right Gesture (RG) to increment the value. Upon setting the X, Y and Z coordinate value the Moveit! and RViz was seen planning the best path to reach the set coordinates. Upon finding the best plan the path was executed and the robot arm simulation was seen moving in order to reach the desired location. Figure 14 shows the final state of the simulated robot after executing the path planned to reach the set coordinates.

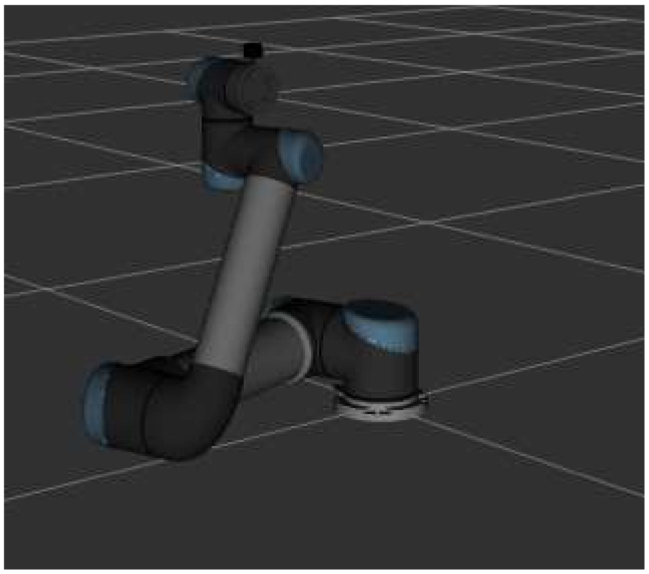


Figure 14: **UR5 Simulated robot arm after performing the planned path to reach the desired coordinates.**

5 CONCLUSION

In this paper, the head gesture recognition for robot control is presented. The end-user is able to perform “hands-free” control of the movement of the robot arm in Cartesian space so to move to the end effector location set using head gestures. Robot control using head gesture recognition was also implemented in this paper. User is able to control the Gazebo simulated robot arm and make it move along X-axis, Y-axis & Z-axis. In the future, tasks such as pick and place, as well as gripper functions can be added. And also, another dimension of controllability like eye tracker-based controls can be added in to system to give it more functionality as controlling robot arm with just the four basic head gestures makes it very tedious. This will make it easy to perform complex tasks like picking up things and placing them in a pre-designated position.

ACKNOWLEDGMENTS

This study and project were supported by the Heracleia Lab at The University of Texas at Arlington. We would like to thank Professor Dr. Fillia Makedon for this opportunity.

We also would like to thank Maria Kyrarini, who has previously worked on this same type of project for all the support and advices.

Lastly, we would like to thank the teaching assistants, Akilesh R, Ashwin Ramesh and Varun Kanal for their help and support in doing this project.

REFERENCES

[1] M.A. Haseeb, M. Kyrarini, S. Jiang, D. Ristic-Durrant, A. Gräser. 2018. Head Gesture-based Control for Assistive Robots. In Proceedings of 11th ACM International Conference on PErvasive Technologies Related to Assistive Environments (PETRA), Corfu Greece, June 2018 (Petra’18), 5 pages. https://doi.org/10.1145/3197768.3201574

[2] World Health Organization (WHO), "International Perspectives on Spinal Cord Injury", 2013. [Online] Available: http://www.who.int/disabilities/policies/spinal_cord_injury/en/

[3] Rechy-Ramirez, E.J. and Hu, H., 2015. Bio-signal based control in assistive robots: a survey. Digital Communications and networks, 1(2), pp.85-101.

[4] Graser, A., Heyer, T., Fotoohi, L., Lange, U., Kampe, H., Enjarini, B., Heyer, S., Fragkopoulos, C. and Ristic-Durrant, D., 2013. A supportive friend at work: Robotic workplace assistance for the disabled. IEEE Robotics & Automation Magazine, 20(4), pp.148-159.

[5] Kim, D.J., Hazlett-Knudsen, R., Culver-Godfrey, H., Rucks, G., Cunningham, T., Portee, D., Bricout, J., Wang, Z. and Behal, A., 2012. How autonomy impacts performance and satisfaction: Results from a study with spinal cord injured subjects using an assistive robot. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 42(1), pp.2-14.

[6] Alsharif, S., Kuzmicheva, O. And Gräser, A., 2016. Gaze Gesture-Based Human Robot Interface. Technische Unterstützungssysteme, die die Menschen wirklich wollen, p.339.

[7] Zhang, W., Sun, F., Liu, C., Su, W., Tan, C. and Liu, S., 2017, October. A hybrid EEG-based BCI for robot grasp controlling. In Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on (pp. 3278-3283).

[8] Jackowski, A., Gebhard, M. and Gräser, A., 2016, May. A novel head gesturebased interface for hands-free control of a robot. In Medical Measurements and Applications (MeMeA), 2016 IEEE International Symposium on (pp. 1-6). IEEE.

[9] Rudigkeit, N., Gebhard, M. and Graser, A., 2015, December. An analytical approach for head gesture recognition with motion sensors. In Sensing Technology (ICST), 2015 9th International Conference on (pp. 1-6). IEEE.

[10] Jackowski, A., Gebhard, M. and Thietje, R., 2017. Head Motion and Head Gesture Based Robot Control: A Usability Study. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 26(1), pp.161-170.

[11] Montaser N. Ramadan, Mohammad A. Al-Khedher., 2015. Wheelchair control using 3 DOF head gesture recognition. International Journal of Mechanical And Production Engineering, ISSN: 2320-2092, 3(4)

[12] Cortes, C. and Vapnik, V., 1995. Support-vector networks. Machine learning, 20(3), pp.273-297.

[14] https://www.universal-robots.com/products/ur5-robot/

[15] https://www.digitaltrends.com/cool-tech/kinova-robotics-wheelchair-robot-arm/
[16] https://www.dlr.de/rm/en/desktopdefault.aspx/tabid-12426/#gallery/27644