1-1-2016

Implementation Of Camera Arm Control By An Oculus Rift On A Da Vinci Surgical System Simulation

Hamid Sadeghi
Wayne State University,

IMPLEMENTATION OF CAMERA ARM CONTROL BY AN OCULUS RIFT ON A DA VINCI SURGICAL SYSTEM SIMULATION

by

HAMID SADEGHI

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan,

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

2016

MAJOR: ELECTRICAL ENGINEERING

Approved by:

_____

Advisor Date

DEDICATION

To my beloved parents and family.

ACKNOWLEGMENTS

I am thankful to my advisor, Dr. Abhilash Pandya, for giving me this opportunity to learn about robotics and perform research. I would like to thank Dr. Luke Reisner for being supportive and his useful ideas during my project. Also, I would like to take this opportunity to thank my colleagues in the CARES laboratory.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Motivation

Robotic and laparoscopic systems are increasing in use for surgery. The surgeon has to instruct a person to move the camera in the laparoscopic case which can result in positioning issues. The time of laparoscopic surgery was reduced by using an EndoAssist rather than a human camera holder. The surgeon must manually move the camera in the robotic case which can interrupt the flow of his/her surgery momentarily. Camera movement leads to problems involving mental work load and potential errors. A way to automatically move the camera fon-demand could therefore be biennial.

Self-guided robotic camera control system (SGRCCS) is one useful way to help the surgeon during a laparoscopic surgery particularly due to its ability to track colored objects. This method was proposed by Omote et al. in which the camera follows colored objects at the end of surgical instruments. It can help surgeons to avoid distraction while performing surgery and reduce surgery time significantly [9]. The need for camera assistant in laparoscopic surgery has been addressed using Automated Endoscope System for Optical Positioning (AESOP). It has been indicated that the time to learn to control laparoscopic by AESOP and manually control is almost equal [16].in addition, AESOP is difficult to use.

FreeHand is the next generation of EndoAssist. It comes with a number of new features. For starters it is much cheaper than the first generation. It is also much easier to setup. Additionally, it comes with an optical system which enables the surgeon to control it by his/her head motion [17] [15].

Another camera assistant used by surgeons, is ViKY EP. It is unique in that the endoscopic holder can be controlled by the surgeon's voice and a foot paddle. However, the ViKY EP comes with a con—the surgery operation time is oftentimes increased when it is used [15].

Figure 1. Laparoscopic Surgery (left). EndoAssistant (middle) and SGRCCS (right) [19][20][9]

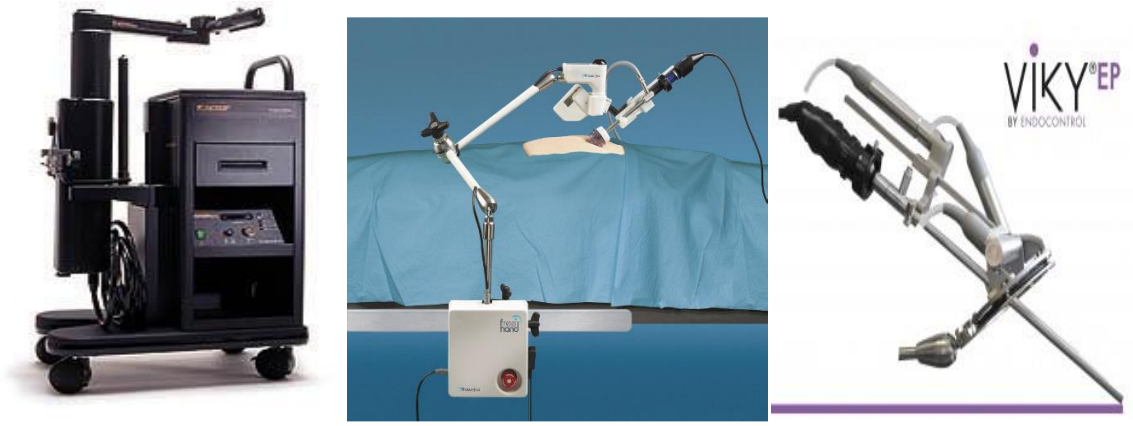

Figure 2. AESOP (left), FreeHand (middle) and ViKY EP (right) [3] [4] [22]

Medical Robotics and Importance of MIS

The definition of robotic surgery was used in 1985 where a fixture was held by an industrial robot next to a patient's body [5]. The role of robots in medical science and remote surgery becomes a significant issue when medical application is growing swiftly. One of its applications

can be Image-guided Surgery which is a process of using a stereotactic helmet to adapt frames. Another application of medical robotics can be its role in rehabilitation [1].

It is not possible to talk about medical Robotics and ignore minimally invasive surgery (MIS) role as a main form of medical surgery. In comparison with open surgery, MIS significantly reduces recovery time along with a significantly less invasive process. However, MIS does come with its downsides in that it is a more challenging surgery to perform due to the limited field of view the surgeon has.

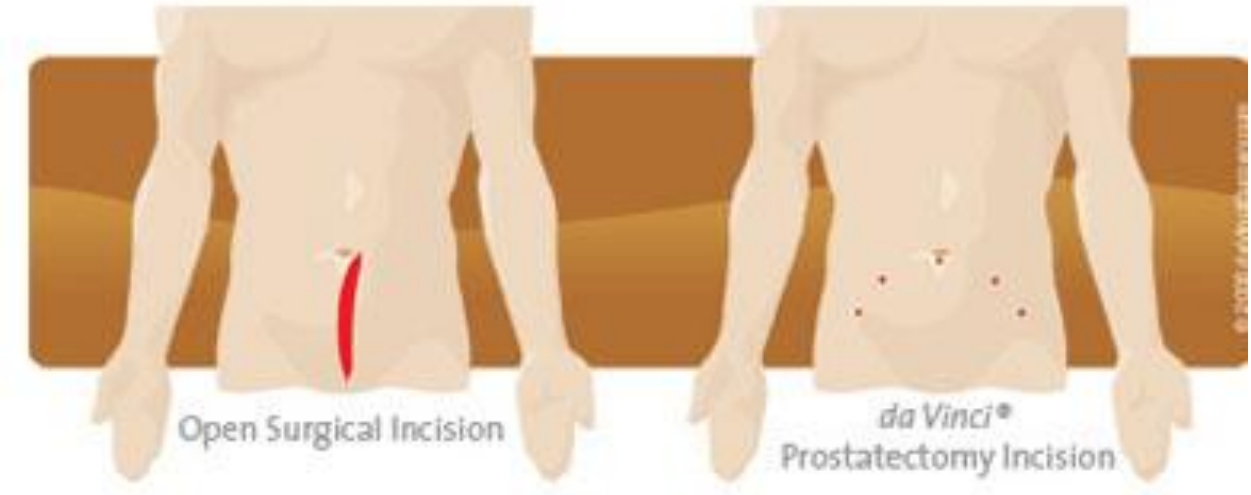

Figure 3. Incision illustration in open surgery and robotic surgery [8]. Multiple ports needed for both tools and camera in robotic surgery. In the robotic case procedures to be performed through 1-2 cm incisions.

The da Vinci surgical system consists of two master tool manipulators (MTMs), two or more patient side manipulators (PSMs) and a camera arm (ECM). It is a surgical system that is an example of MIS, in which all tools inserted into a patient body and operation can be controlled by the robot manipulators [5][6].

Figure 4. The da Vinci Surgical System surgeon console (left) and Patient Cart-da Vinci Si HD Surgical System (right) [23] [24]

CHAPTER 2: BACKGROUND

Camera Robot

Currently, two methods for camera control have been used. First, a standard clutch-based method for manual camera movement in which the surgeon can move the camera manually. This method may serve as source of distraction. The surgeon needs to reposition the camera frequently by a pair of manipulators. Second, an autonomous camera also known as auto-camera method is used. In this process the camera is moved with-respect-to the center of surgical tool arms with automatic zoom ability. This is one method has been proposed to remove the surgeon's distraction [10].

Robot Operating System and RViz

Robot Operating System (ROS) has broad applications. Zamean, Slany et al. proposed a control system method based on ROS in order to map and navigate different environments [13]. RViz (Robot Visualizer) is visualization tools which are provided by ROS. It enables researchers to test and validate their data on RViz before applying on actual robots. Markers are geometric primitives that allow annotation in the graphic environment [12]. RViz can subscribe and publish robot information using ROS nodes.

Oculus Rift

Oculus Rift provides its user a virtual environment that helps the user experience this virtual environment in the form of a 3D environment. Oculus Rift is considered as a human-machine interface. Oculus Rift improved other VR (virtual Reality) headset issues such as motion and sickness issues. This headset was presented as two different versions of developer kit: DK1 and DK2. It comes with three sets of lenses come with Oculus Rift: -A, B and C. Furthermore, the Oculus Rift comes with Head tracker sensors. These Head trackers enable the Oculus Rift to

get head movement data which is particularly useful for controlling the view. A magnetometer, gyroscope and accelerometer are VR sensor on Oculus Rift which are useful to determine a user's head position and orientation [14].

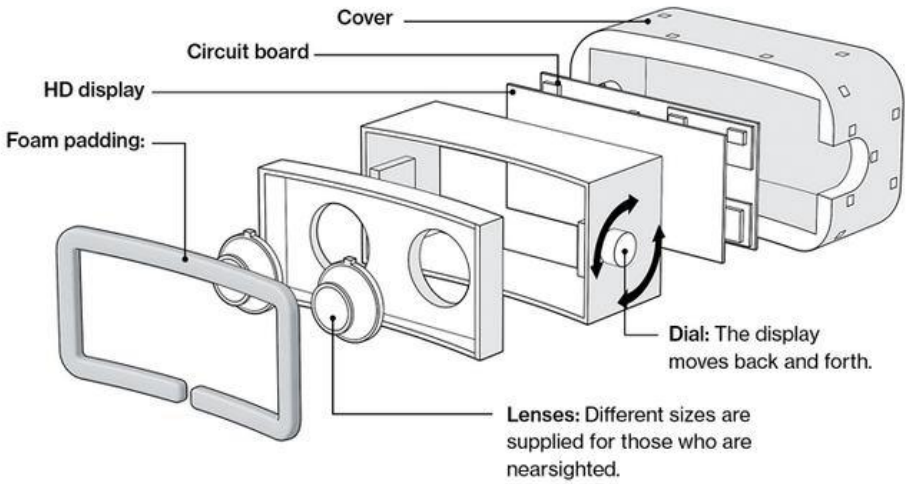

Figure 5. The internal structure of Oculus Rift headset [18]

Potential Solution

A new method had been proposed in which the camera arm of the da Vinci can be moved based on the Oculus Rift orientation and position.

Figure 6. Goal of research: Oculus and da Vinci

CHAPTER 3: METHODS

Oculus Rift runs on Windows. It provides pose (position and orientation) information and 3D view capability.

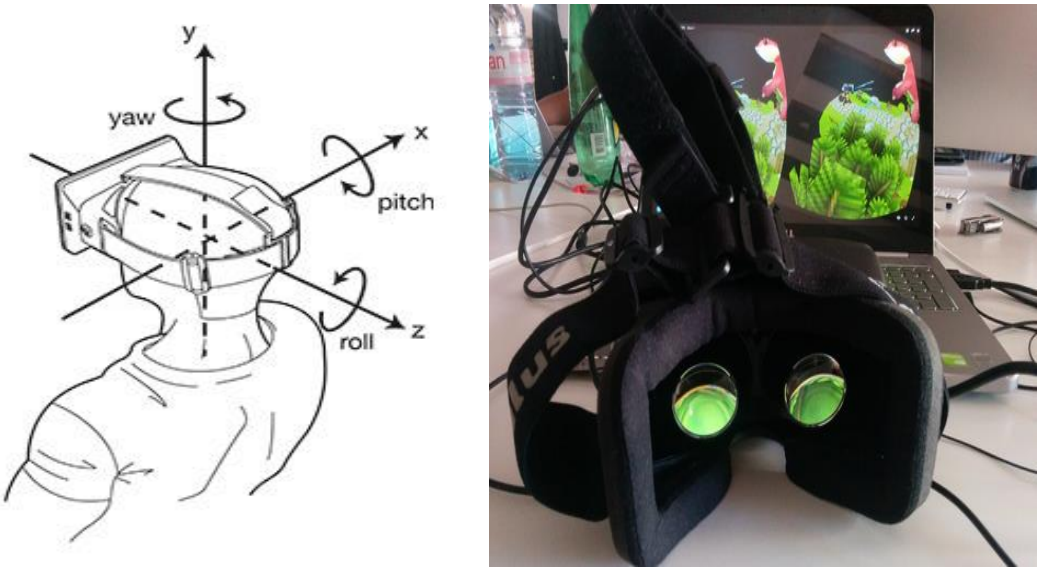

Figure 7. An illustration of three axes [21] (left) and Oculus Rift 3D view (right) [7]

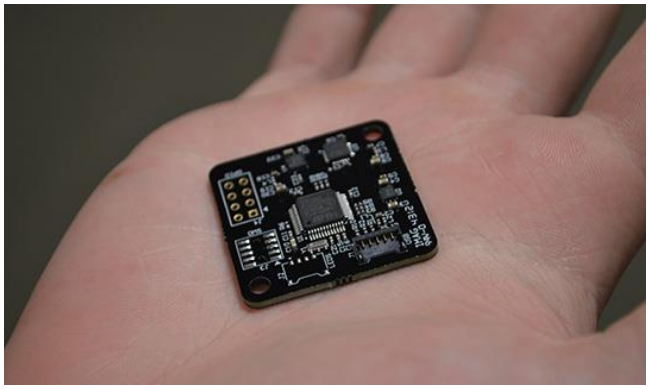

Figure 8. Oculus head tracking sensors [25]

Tracking sensors on an Oculus Rift such as:

- STMicroelectronics 32F103C8 ARM Cortex-M3 microcontroller
- Inverse MPU-6000 (gyroscope + accelerometer)
- Honeywell HMC5983 (manometer) [27]

Tracking sensors on Oculus Rift enables us to find the pose data that we need for RViz simulation. There are two ways to find the Oculus Rift pose data: First, Inertial measurement Unite (IMU) that provides pose data so fast but not accurate enough. Second, the inferred camera that detects pose data of LEDs on the Oculus Rift more accurate but slowly. The combination of these ways uses to provide pose data of the Oculus Rift

RViz Simulation Environment

The da Vinci Surgical System RViz simulation consists of model of the da Vinci Surgical System. The RViz simulation is comprised of PSM arms, ECM arm, and setup joints. These joints can be activated and moved. RViz runs on Ubuntu and uses ROS nodes. The da Vinci Surgical System hardware moves using the same interface. Since the Oculus Rift runs in Windows, the major issue for this project is how to translate the Oculus information into Ubuntu/RViz?

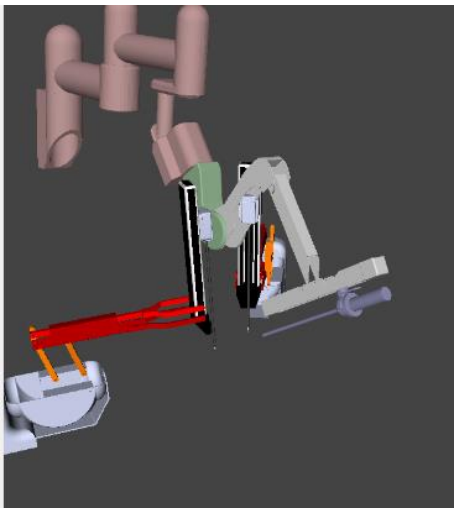

Figure 9. RViz simulation

In order to publish data on the simulation, oculus node was used to publish data on topic for joints state of the Oculus Rift.

Justification for Socket-based Communication

The Oculus Rift runs in the Windows operating system. The simulator and the da Vinci hardware run in Ubuntu. There is a need to connect the data from the Oculus to the ROS on the Ubuntu side. The solution to this is to use Socket to connect Ubuntu to Windows.

Socket Programming and Client/Server Communication

Computer networks are the combination of communication technology and computer technology. Computers networks enable two computers to communicate with each other, even when running different operating systems. Client and server connection is a way to share information between clients and servers [11]. In this study, I used socket programming in order to transmit and receive data between the Oculus Rift and RViz simulation.

Steps for server to connect to a client and transfer data:

1. Create a socket
2. Bind the socket to an IP
3. Listen on the socket for a client
4. Accept connection
5. Transfer data
6. Close connection

Steps for a client to connect to a server and transfer data:

1. Create a socket
2. Connect to the server
3. Transfer data through the socket
4. Close the connection

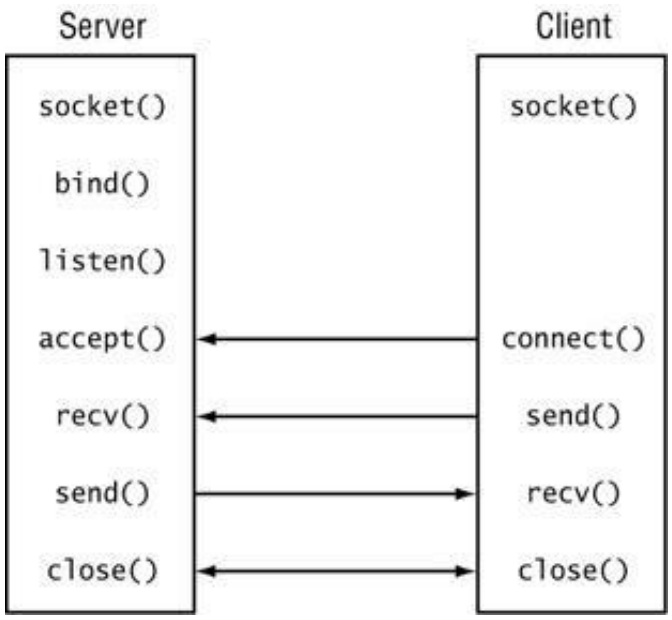

Figure 10. Server-Client connection [26]

System Integration

The Oculus Rift was used to control the camera arm by the user's head movement. The alignment between the user's head movement and the camera arm was based on the Oculus Rift pose data. The first set of data is the home position of the ECM. The pose matrix of the Oculus is then used to slave the ECM robot arm.

Whole System Block Diagrams

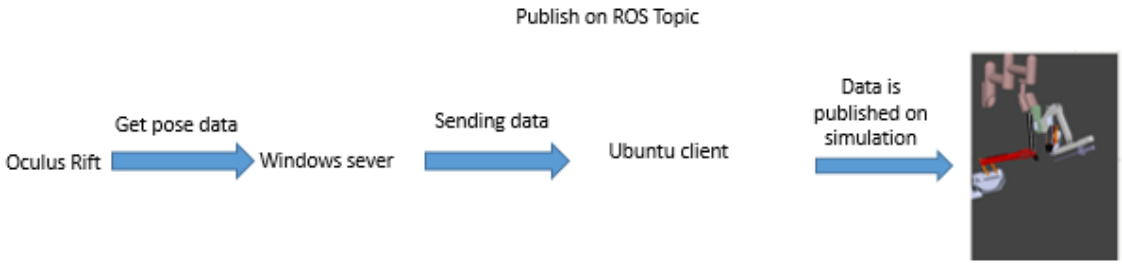

Figure 11. Illustration of the Oculus-RViz block connection

Steps:

1. The pose data of Oculus Rift is got and sent through the socket on Windows.
2. The data is received by Python.
3. Oculus node publish the data on topic of the camera arm joints angles
4. The topic was used for the camera arm on simulation

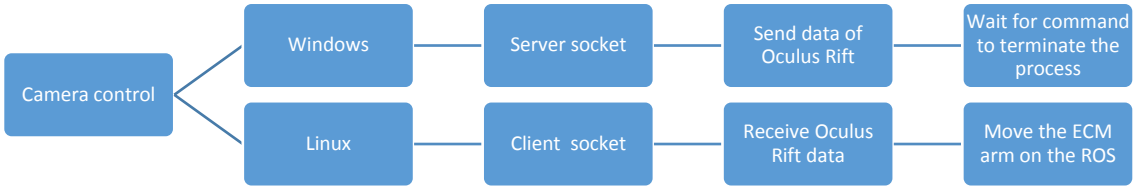

Figure 12. Windows-Linux block diagram

Python and C++ Code for Client and Server

After the socket connection is created, server (C++) code starts sending the data and client (Python) starts receiving it.

```
int main()
{
    Oculus oculus;
    Server server;
    vector<double> t(7, 0);
    int result = oculus.intialize_oculus();
    server.initialize_socket();

    while (true) {
        t = oculus.get_oculus_Position(result);
        server.get_oculus(t);
        cout << "Orientation.x" << t[0] << endl;
        cout << "Orientation.y" << t[1] << endl;
        cout << "Orientation.z" << t[2] << endl;
        cout << "Orientation.w" << t[3] << endl;
        cout << "Position.x" << t[4] << endl;
        cout << "Position.y" << t[5] << endl;
        cout << "Position.z" << t[6] << endl;
    }
    return 0;
}
```

Figure 13. Main loop for socket programming C++

```
import socket
class client:
    def __init__(self):
        self.a=[[]for _ in range(7)]
        HOST, PORT = '172.21.32.146', 27015
        self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.s.connect((HOST, PORT))
    def send_data(self):
        self.s.send('s\n')
        for x in range(0, 7):
            self.a[x]=str(self.s.recv(25))
        return self.a
    def close_socket(self):
        self.s.send('x\n')
```

Figure 14. Main loop for socket programming Python

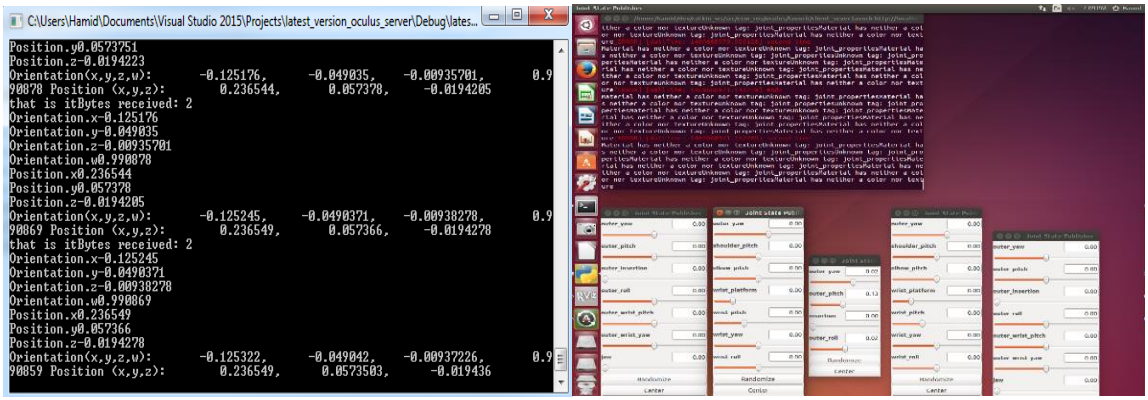

Figure 15. The pose data is being sent by server (left) and is being received by client (right)

Converting Quaternion to Rotation Matrix

$$R = \begin{pmatrix} a^2+b^2-c^2-d^2 & 2bc-2ad & 2bd+2ac \\ 2bc+2ad & a^2-b^2+c^2-d^2 & 2cd-2ab \\ 2bd-2ac & 2cd+2ab & a^2-b^2-c^2+d^2 \end{pmatrix}$$

Figure 16. General rotation matrix where a, b, c and are quaternions [2]

$$T = \begin{bmatrix} R & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix}$$

Figure 17.Rotation matrix is part of transformation matrix

Forward Kinematics

In order to find transformation matrix from joint angles the following formula is used:

$T = forward\_kinematics[(\theta_1, \theta_2, \theta_3, \theta_4)]$

ECM has four degrees of freedom

Figure 18. Forward kinematics

Inverse Kinematics

Inverse kinematics is calculated from the following formula where the output is joint angles:

$$[\theta_1, \theta_2, \theta_3, \theta_4] = inverse\_kinematics\ (transformation\ matrix)$$

Figure 19. Inverse kinematics

Algorithm Used

1. The initial ECM rotation matrix is calculated by using forward kinematic formula (Figure 17).
2. The orientation data (in quaternions) of the Oculus Rift is received and converted into a 3x3 rotation matrix (called the current Oculus rotation matrix)

Main While loop:

3. The orientation data (in quaternions) of the Oculus Rift is received and converted into a 3x3 rotation matrix (called the next Oculus rotation matrix)
4. In order to calculate how much the Oculus Rift was rotated, the following formula is used:

$$delta\ rotation = next\ oculus\ rotation * (current\ oculus\ rotation\ )^{-1}$$

5. The current rotation matrix of the ECM is multiplied by the oculus rotation matrix giving us next rotation matrix of the ECM.

$$next\ ecm\ rotation = delta\ rotation * current\ ecm\ rotation$$

6. The following sentences express how the ECM transformation matrix is found:

a. The rotation matrix of the ECM transformation matrix is set equal to the new ECM rotation matrix.

$$ecm\ transformation = next\ ecm\ rotation$$

b. The position of the ECM is fixed (keyhole). This fact is used for finding the position of the ECM transformation matrix (last column of the matrix). The keyhole position is calculated from forward kinematics formula.

$$keyhole = forward\_kinematics[(0, 0, 0, 0)]$$

$$position\ of\ ecm\_transformation = keyhole$$

7. New joint angles for ECM are the result of inverse kinematics of rotation matrix, which is used to publish on the simulation.

$$ecm\_joint\_angles = inverse\_kinematics(ecm\_transformation)$$

8. Oculus node publishes ECM joint angles on the topic of the camera arm

Testing

In order to test algorithm, we compare the rotation angle of the Oculus Rift that is measured manually with the data is publishing on the simulation. The angle between each pair of orientation data (in quaternion) is calculated by using following formula:

$$2 * arccos(absolute\ value\ of\big(dot\ product\ (q1, q2)\big) * 180/pi$$

Where q1 and q2 are quaternion value. This command shows angle between to quaternions in degree.

CHAPTER 4: RESULT

Six tests were implemented to observe and compare the Oculus Rift movement with the simulation movement and the Euler angles. In all test the Oculus Rift movement matched with the data was used to publish on the camera arm.

Test 1: 30 Degrees around Y Axis

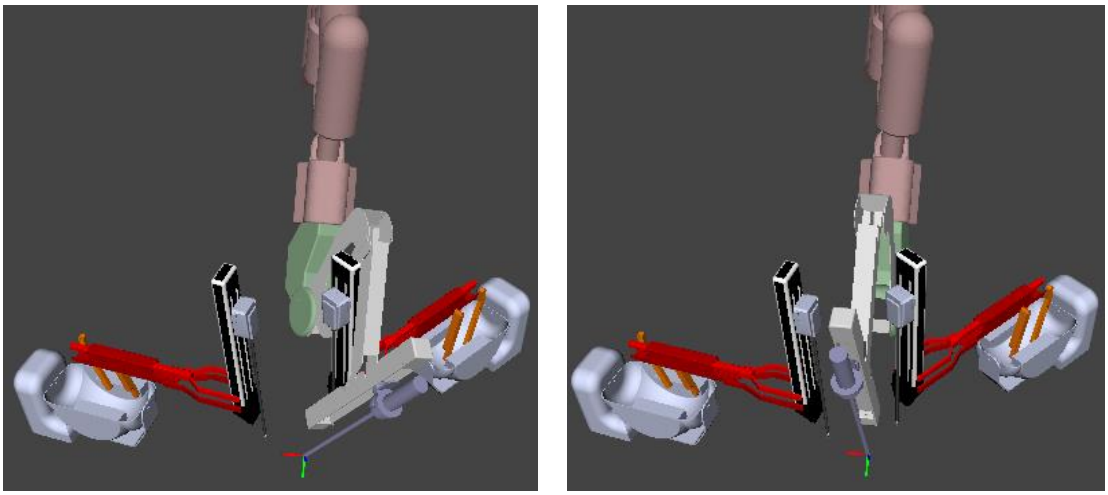

Figure 20. Simulation environment for the da Vinci Surgical System, initial position (left) and final position (right)

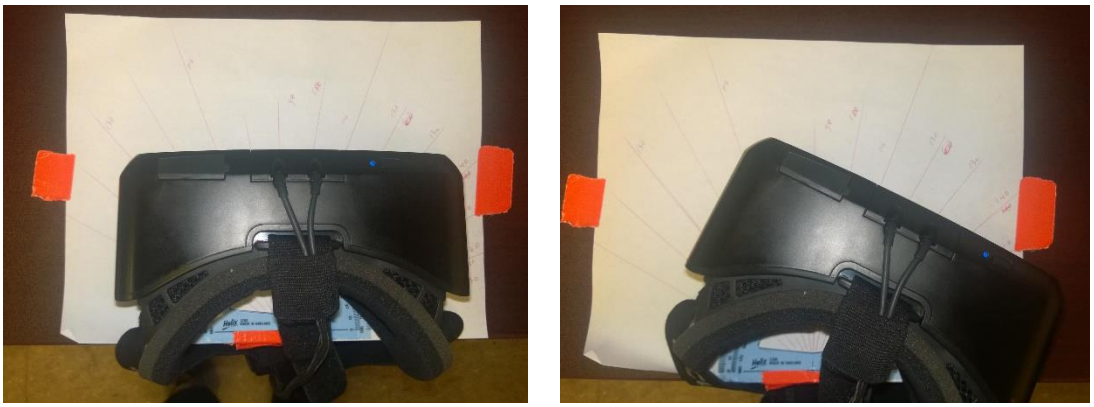

Figure 21. The Oculus Rift position, initial position (left) and final position (right)

Test	Initial quaternion of simulation	Next quaternion of simulation	Degree from the Oculus Rift	Degree from the simulation
1	[-0.63678, 0.64683, -0.33389, -0.25426]	[-0.61574, 0.63901, -0.42998, -0.16629]	30	31.029

Table 1. test1

Test 2: 50 Degrees around Y Axis

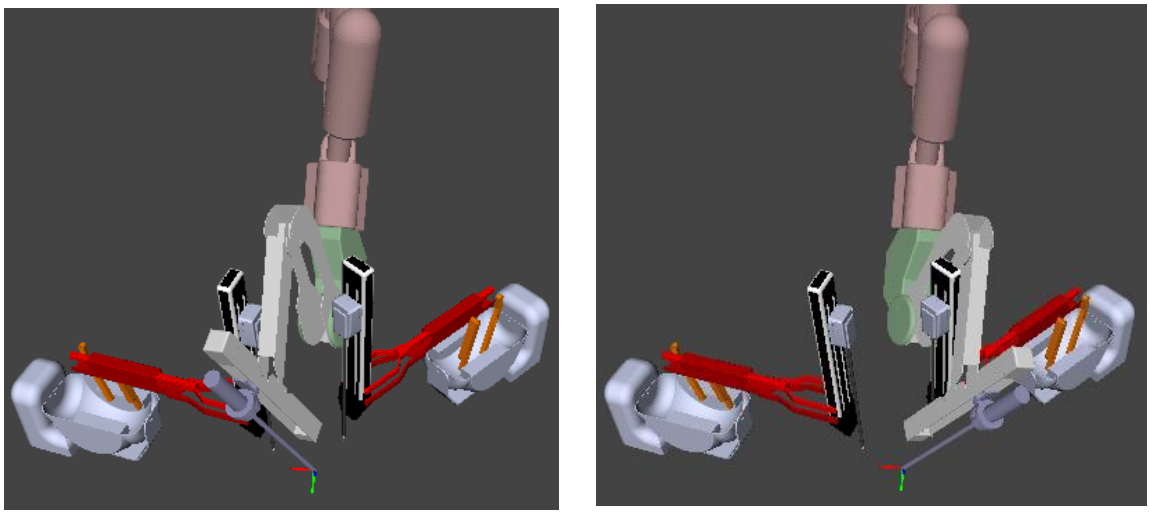

Figure 22. Simulation environment for the da Vinci Surgical System, initial position (left) and final position (right)

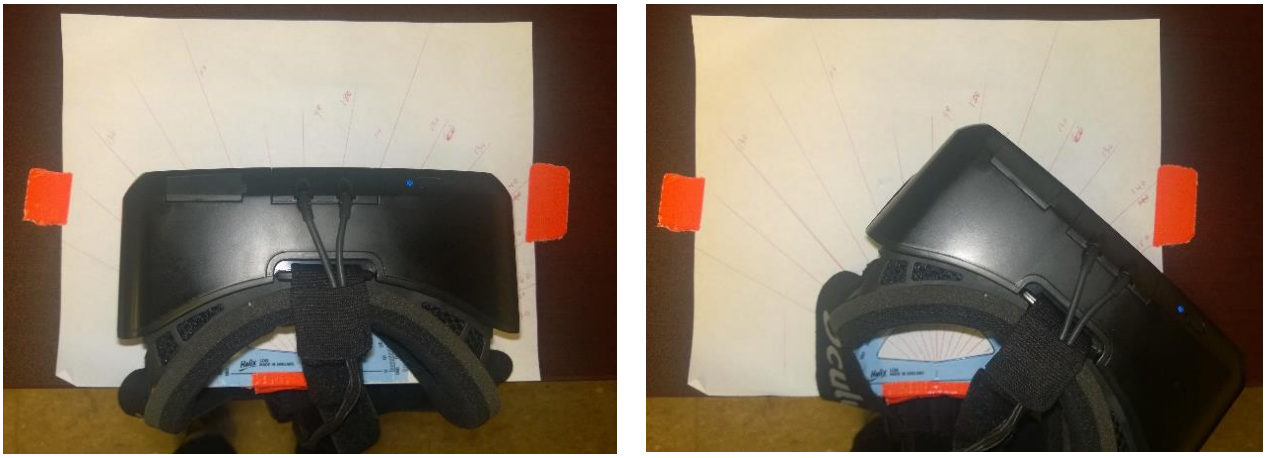

Figure 23. The Oculus Rift position, initial position (left) and final position (right)

Test	Initial quaternion of simulation	Next quaternion of simulation	Degree from the Oculus Rift	Degree from the simulation
2	[-0.63678, 0.64683, -0.33389, -0.25426]	[0.70976, -0.49331, 0.085756, 0.49551]	50	44.461

Table 2. test2

Test 3: 20 Degrees around Y Axis

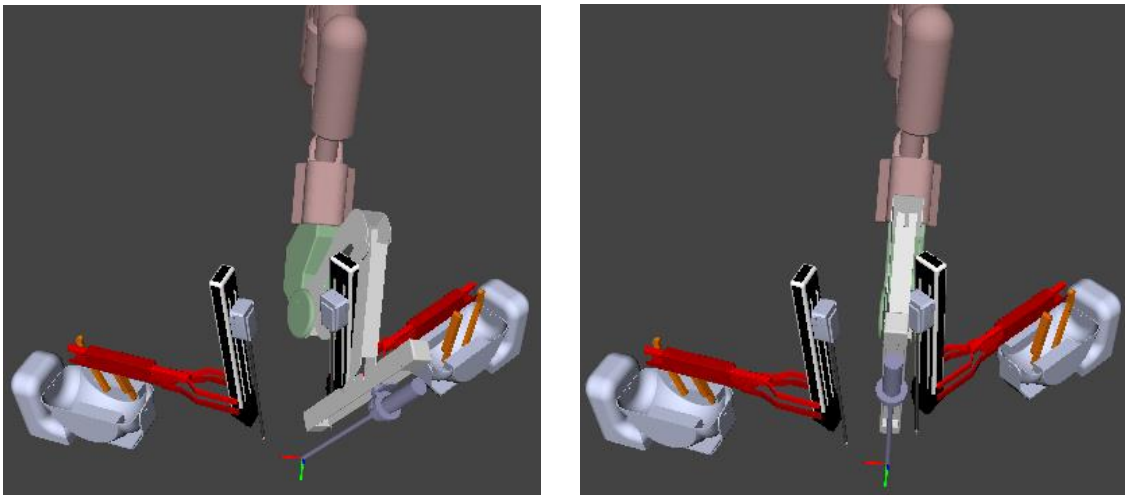

Figure 24. Simulation environment for the da Vinci Surgical System, initial position (left) and final position (right)

Figure 25. The Oculus Rift position, initial position (left) and final position (right)

Test	Initial quaternion of simulation	Next quaternion of simulation	Degree from the Oculus Rift	Degree from the simulation
3	[-0.63678, 0.64683, -0.33389, -0.25426]	[0.70272, -0.56864, 0.22082, 0.36615]	20	21.703

Table 3. test3

Test 4: 10 Degrees around Y Axis

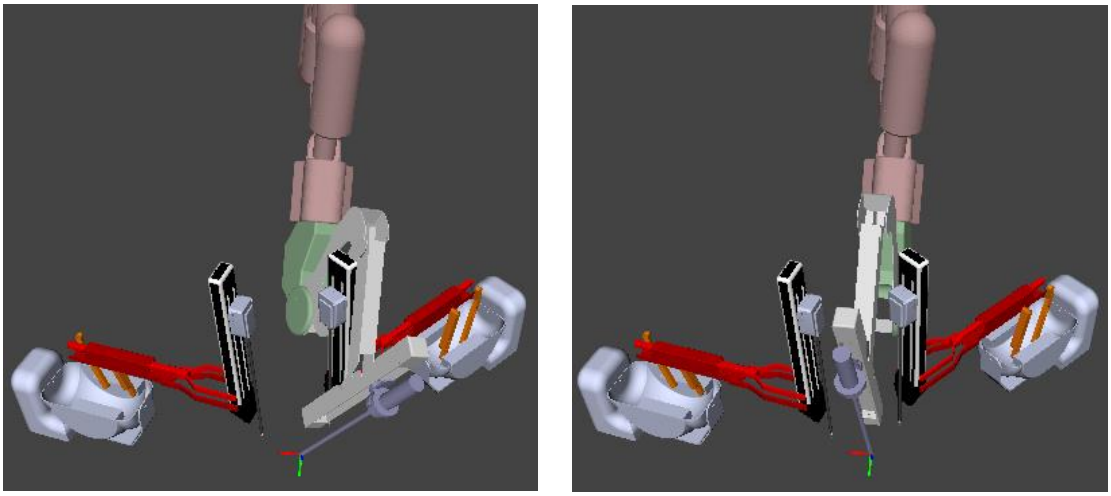

Figure 26. Simulation environment for the da Vinci Surgical System, initial position (left) and final position (right)

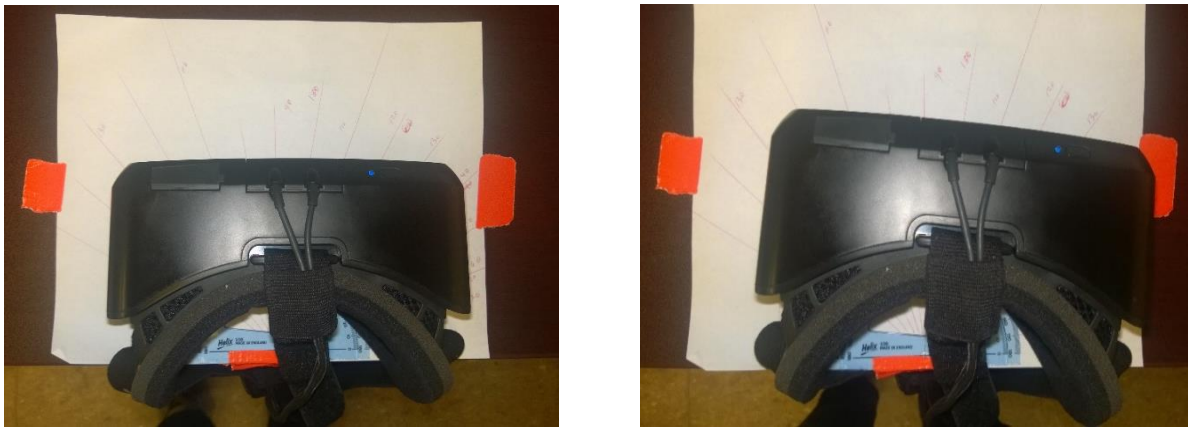

Figure 27. The Oculus Rift position, initial position (left) and final position (right)

Test	Initial quaternion of simulation	Next quaternion of simulation	Degree from the Oculus Rift	Degree from the simulation
4	[-0.63678, 0.64683, -0.33389, -0.25426]	[-0.61574, 0.63901, -0.42998, -0.16629]	10	15.154

Table 4.test 4

Test 5: 45 Degrees around Y Axis

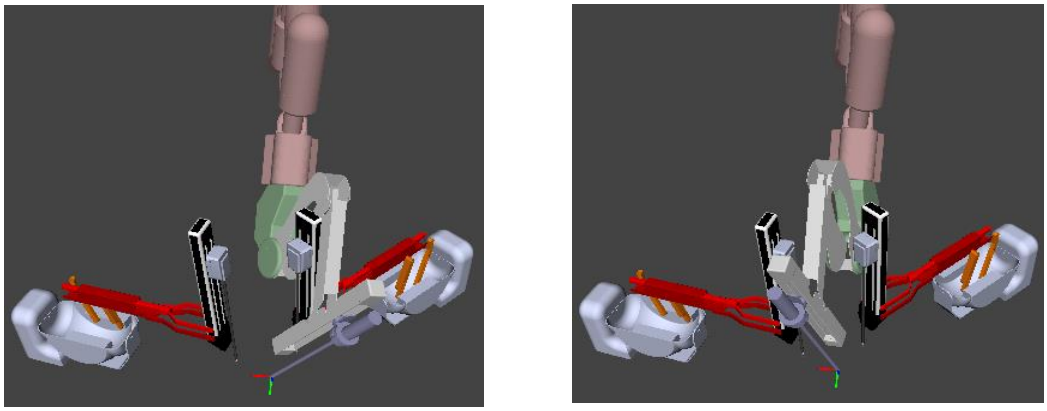

Figure 28. Simulation environment for the da Vinci Surgical System, initial position (left) and final position (right)

Figure 29. The Oculus Rift position, initial position (left) and final position (right)

Test	Initial quaternion of simulation	Next quaternion of simulation	Degree from the Oculus Rift	Degree from the simulation
5	[-0.63678, 0.64683, -0.33389, -0.25426]	[-0.61574, 0.63901, -0.42998, -0.16629]	45	41.85

Table 5.test 5

Test 6: 45 Degrees around X Axis

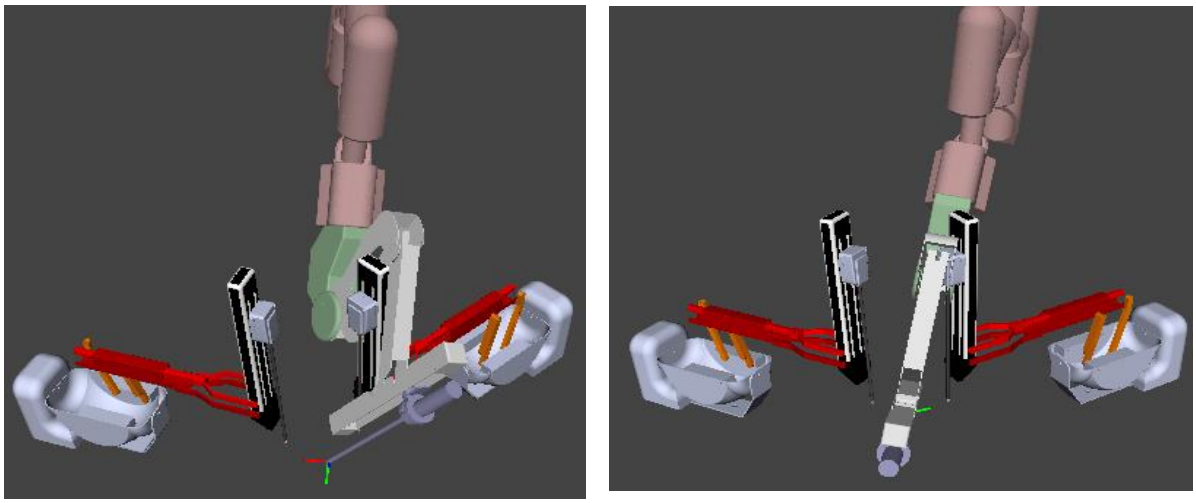

Figure 30. Simulation environment for the da Vinci Surgical System, initial position (left) and final position (right)

Figure 31. The Oculus Rift position, initial position (left) and final position (right)

Test	Initial quaternion of simulation	Next quaternion of simulation	Degree from the Oculus Rift	Degree from the simulation
6	[0.71468, -0.6557, 0.22048, 0.10333]	[0.78892, -0.46745, 0.36534, 0.16005]	30	29.331

Table 6. test6

CHAPTER 5: CONCLUSION

In this study, the data was transferred through the socket and shown in the simulation. In addition, the Oculus Rift rotation angle matched the rotation angle of the camera arm in the simulation. For system evaluation, the several headset rotation angles were compared to the camera arm rotation angles in the simulation. The results demonstrated that the user can move the camera arm using head motions, and the orientation of the camera arm closely matched the headset's orientation.

In this study, we proposed the control of the camera arm via an Oculus Rift as a new method for camera control. The RViz simulation closely reflected the movements of the headset, and the feasibility for the method was demonstrated during the tests.

CHAPTER 6: FUTURE WORK

1. Implement the joint angles on the actual da Vinci Surgical System camera arm
2. Find a method for zooming in and out. My suggestions are:
    a. Use a button on the Oculus Rift which enables the user to do that
    b. Use the zoom keep the tools in the field of view
3. Create a Graphic User Interface to enable user to start and stop process
4. Test this method with the hardware and compare results with other methods
5. Send the camera view to the Oculus Rift display
6. Check to see whether subject can see the system comfortably during a simple operation

APPENDIX A: OCULUS RIFT DK2

- Display
  - Resolution 1920 x 1080
  - Refresh rate 75 Hz, 72 Hz, 60 Hz
  - Persistence 2 ms, 3 ms, full
- Viewing Optic
  - Viewing Optics 100° Field of view (nominal)
- Positional Tracking
  - Sensors Near Infrared

    CMOS sensor
  - Update Rate 1000 Hz

- Included Accessories
  - Included Accessories

    HDMI to DVI Adapter

    DC Power Adapter

    International Power Plugs

    Nearsighted lens

    Cups

    Lens cleaning

    Cloth

- Interfaces
    - Cable 10’ (detachable)
    - HDMI Yes
    - USB Device Yes
    - USB Host USB 2.0

      (Requires DC Power Adapter)
    - Positional USB 2.0
    - Tracker USB USB 2.0
- Weight
    - Weight 440 grams
- Internal Tracking
    - Sensors Magnetometer

      Accelerometer

      Gyroscope
    - Update rate 1000 Hz

APPENDIX B: PYTHON CODE

Some ROS and robotics commands that I used in this study.

Homogenous matrix= quaternion_matrix (quaternion)

This command was used to get transformation matrix from quaternions.

ecm_pub= rospy.Publisher ('topic name', JointState, queue size)

It creates an ECM publisher

Transformation matrix= ecm_kin_forward (joint angles)

This command was used to do forward kinematics which get joint angles as inputs. Transformation matrix of the camera arm is its output.

delta_rot=numpy.matrix (next_oculus_rot)* numpy.matrix (numpy.linalg.inv (current_oculus_rot))

It finds rotation matrix between the current oculus rotation matrix and next oculus rotation. The linalg.inv finds inverse of the matrix.

Euler_angles=numpy.matrix (transpose (delta_rot))

This function finds transpose of a matrix in which delta rotation is rotation matrix. The output is Euler angles.

Joint_angles_ecm=ecm_kin.inverse (transformation_mat)

It calculates inverse kinematics of the ECM. It takes transformation matrix as an input and its output is joint angles of the ECM

rospy.init_node ('node name')

This command initializes a node where node name in my project was oculus node.

S=socket. Socket (socket.AF_INET, socket.SOCK_STREAM)

s.connect ((HOST, PORT))

This command makes a socket and connect to a server.

s.send (data)

It sends data through the socket

s.recv (25)

I used this command to receives a string data type

APPENDIX C: C++ CODE

Some C++ commands that were used in this project.

WSADATA wsaData

Makes WSADATA structure

WSAstartup (MAKEWORD (2, 2), &wsaData)

Initialize Winsock

ListenSocket = socket (result->ai_family, result->ai_socktype, result->ai_protocol)

Create a socket for server connection

iResult = bind (ListenSocket, result->ai_addr, (int) result->ai_addrlen)

Bind a socket to an IP

iResult = listen(ListenSocket, SOMAXCONN)

Listen to the socket

ClientSocket = accept (ListenSocket, NULL, NULL)

Accept a client socket

iResult = recv(ClientSocket, recvbuf, recvbuflen, 0)

Receive data

closesocket (ClientSocket)

Close the socket

WSACleanup ()

Clean up WSAC

ovrHmd hmd

Create an ovrHmd object

ovrTrackingState ts

Create ovrTrackingState object

ovrSensorData sensorData

Create sensorData object

ovrVector3f gyroData

Create ovrVector3f

ovrVector3f magData

Create ovrVector3f

T [0] = ts.HeadPose.ThePose.Orientation.x

T [1] = ts.HeadPose.ThePose.Orientation.y

T [2] = ts.HeadPose.ThePose.Orientation.z

T [3] = ts.HeadPose.ThePose.Orientation.w

T matrix consists of quatrianian data

ovrHmd_Destroy (hmd)

Destroy hmd

ovr_Shutdown ()

shutdown ovr

REFERENCES

[1] (Dario, Guglielmelli et al. 1996)Dario, P., et al. (1996). "Robotics for medical applications." IEEE Robotics & Automation Magazine 3(3): 44-56.

[2] https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation

[3] http://www.ideasforsurgery.com/wp-content/uploads/2009/01/aesop2.jpg

[4] https://www.wired.com/images_blogs/gadgetlab/2009/09/surgical_robots_5a.jpg

[5] Davies, B. (2000). "A review of robotics in surgery." Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine 214(1): 129-140.

[6] Dogangil, G., et al. (2010). "A review of medical robotics for minimally invasive soft tissue surgery." Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine 224(5): 653-679.

[7] https://blog.sketchfab.com/wp-content/uploads/2014/07/tumblr_inline_n855ucUg671rh8rjh.png

[8] http://www.ucdmc.ucdavis.edu/urology/images/body/incision_illustration.jpg

[9] Omote, K., et al. (1999). "Self-guided robotic camera control for laparoscopic surgery compared with human camera control." The American journal of surgery 177(4): 321-324.

[10] ESLAMIAN, S., et al. (2016). "Towards the Implementation of an Autonomous Camera Algorithm on the da Vinci Platform." Medicine Meets Virtual Reality 22: NextMed/MMVR22 220: 118.

[11] Xue, M. and C. Zhu (2009). The socket programming and software design for communication based on client/server. Circuits, Communications and Systems, 2009. PACCS'09. Pacific-Asia Conference on, IEEE.

[12] Gossow, D., et al. (2011). "Interactive markers: 3-d user interfaces for ros applications [ros topics]." IEEE Robotics & Automation Magazine 18(4): 14-15.

[13] Zaman, S., et al. (2011). ROS-based mapping, localization and autonomous navigation using a Pioneer 3-DX robot and their relevant issues. Electronics, Communications and Photonics Conference (SIECPC), 2011 Saudi International, IEEE.

[14] Desai, P. R., et al. (2014). "A review paper on oculus rift-a virtual reality headset." arXiv preprint arXiv:1408.1173.

[15] Pandya, A., et al. (2014). "A review of camera viewpoint automation in robotic and laparoscopic surgery." Robotics 3(3): 310-329

[16] Jacobs, L., et al. (1997). "Determination of the learning curve of the AESOP robot." Surgical endoscopy 11(1): 54-55.

[17] Beasley, R. A. (2012). "Medical robots: current systems and research directions." Journal of Robotics 2012.

[18] https://virtualrealitysb.files.wordpress.com/2014/11/interal-rift.png

[19] http://www.freehandsurgeon.com/images/CompanyHistory/EndoAssist.png

[20] http://hicarebd.com/wp-content/uploads/2015/08/laparoscopy.jpg

[21] http://oculusrift-blog.com/wp-content/uploads/2013/01/headset_fig8.jpg

[22] http://innorobo.com/wp-content/uploads/2015/06/VIKY_EP_BY_ENDOCONTROL-1-300x211.jpg

[23] http://www.pro-israel.ru/wp-content/gallery/aortokoronarnoe-shuntirovanie-v-izraile/coronary-artery-bypass-surgery-in-israel3.jpg

[24] http://static.wixstatic.com/media/5a2197_4018d89592942ce535fe86e5e2ad48d7.jpg/v1/fill/w_480,h_384,al_c,lg_1,q_80/5a2197_4018d89592942ce535fe86e5e2ad48d7.jpg

[25] https://s3.amazonaws.com/static.oculus.com/website/2013/01/DSC_0178.jpg

[26] http://www.codeproject.com/KB/IP/TCPIPChat/ClientServer.jpg

[27] LaValle, Steven M., et al. "Head tracking for the Oculus Rift." 2014 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2014.

ABSTRACT

IMPLEMENTATION OF CAMERA ARM CONTROL BY AN OCULUS RIFT ON A DA VINCI SURGICAL SYSTEM SIMULATION

by

HAMID SADEGHI

August 2016

Advisor: Professor. Abhilash Pandya

Major: Electrical and Computer Engineering

Degree: Master of Science

Camera control methods play a significant role in remote surgery. Two methods have been developed to control the camera arm of the da Vinci Surgical System: a standard clutch-based method for manual movement of the camera and an autonomous camera (auto-camera) method. In the standard method, the surgeon positions the camera manually using a pair of hand controllers. This happens frequently during the surgery and may serve as a distraction during surgical procedures. The second method was developed in order to help surgeon to remove the issue mentioned in the standard method. Auto-camera method enables the system to move the camera autonomously. In this method, the camera is moved with-respect-to the center of surgical tool arms with automatic zoom control ability. There are still many issues with automatically moving a camera. We will show the feasibility of an intermediate solution using an Oculus rift head mounted stereo display.

Achieving the optimal camera viewpoint with simple control methods is of utmost importance for remote surgical systems. We propose a new method to move the camera arm based on sensors within the Oculus Rift. Can a surgeon put the Oculus Rift (virtual reality headset), get

a stereoscopic view and control the camera with simple head gestures? In this case, the surgeon will be able to see the 3D camera view of scope inside of the Oculus Rift and move the viewpoint by his/her head orientation. Position and orientation of the Oculus rift is measured by an inertial measuring unit and optical tracking sensors within the Oculus platform. These data can be used to control the position and orientation of the camera arm.

In this thesis, a complete system will be created based on the Robot Operating System (ROS) and a 3D simulation of the da Vinci robot in RViz. In addition, a usability study will be conducted to analyze system accuracy. For this system evaluation, headset orientation will be compared to corresponding orientation of the camera in simulation. We will also check whether subjects can use the system comfortable during a simple operation.

In this study, we propose controlling of the camera arm by Oculus Rift as a new method for camera control. It is anticipated that the headset movement will be the same as its corresponding simulation in RViz (simulation environment for the robot). We anticipate that our results will demonstrates feasibility for this method to control a camera. We will propose next steps for testing this system on the da Vinci hardware leading towards a system for the operating room of the future.

AUTOBIOGRAPHICAL STATEMENT

I started my undergraduate study in the year 2009. My major was Electronics. During my undergraduate study, I learned about different fields of Electrical Engineering that helped me to understand my dream major. I found myself interested in robotics and I was looking an opportunity to work on this field. Finally, I got that opportunity by starting my graduate study at Wayne State University in the 2014.

During my graduate projects, I learned about robotics, MATLAB, Python and C++ Programming Language as well as control and image processing. My mathematics skills helped me to understand my research.