Analysis of VCOST data to assist future research

George Westergaard

Florida Polytechnic University,

Author Note

Data provided is taken from the VCOST portion of the “Vortex” research grant under Dr. Demirel.

Quick Summary / Abstract

This paper is about my research topic with Dr. Demirel, the effectiveness of virtual training on surgical simulations. The programs include R and the Unity simulation for Straight coloanal anastomosis. “Shiny Dashboard” was used as the template for which the data and its analysis is shown. The program takes in different .csv files, each specified from the output of the simulation. These files are read by the program and have their data synthesized into easy-to-understand figures, either tabular or graphical. The goal is to update this program through the thesis to obtain more coherent and relevant data.

Keywords: VCOST, R, Shiny Dashboard, Unity, Simulation

Analysis of VCOST data to assist future research

# Introduction and Motivation

## Introduction

I am working with Dr. Demirel, under a research grant, on surgical simulations relating to colons. VCOST is the division of the VORTEX research grant that I work for. Universities and doctors are working together under this grant. Florida Polytechnic University’s portion is supervised by Dr. Demirel. There are other branches that are being worked on, but I only focus on the VCOST branch.

I primarily work with a group of doctors from UT Southern in Dallas, Texas. I correspond with them regularly throughout the month to show progress of the simulation and get feedback on what needs to change. Since I am not medically trained, these meetings are vital to create a more realistic simulation of the desired operation. They are planning to run a survey/study with the final version of the simulation with their staff and specialists.

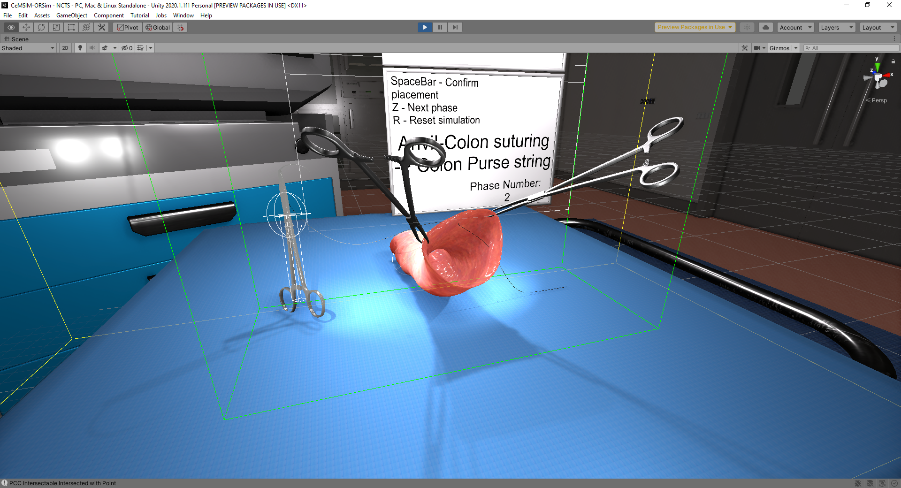
This grant is for providing virtual surgical simulations to either replace or enhance standard training practices. My division is currently working on the Straight Coloanal Anastomosis, reattaching a colon that was separated from the anus in a previous operation. The simulation that is being worked on tracks some information throughout. The simulation utilizes both haptic devices as well as a Virtual Reality (VR) headset. Tracked data will come from both hands, the headset, and scores on both a quiz and performance metrics tallied during the simulation. The program being used to make and develop the simulation is Unity. Unity provides a platform for physics, softbody simulation, 3D models, user input, and much more to create a higher fidelity simulation.

Figure 1 3D Systems Haptic Force Feedback Device

Figure 2 Unity Scene showing VCoST Simulation

All data obtained from the trials come in a .csv file format. Each of the files has its own set of data that is tracked, therefore there is no standard .csv organizational format. This means that the data’s format must be known beforehand and that each of the tabs of the program will only correspond to its specific data input. Some of the data needed to be “cleaned” since the haptic devices and headset would sometimes register an outlier that could not be possible, whether it be exceeding the physical limitations of the range of motion of the haptic devices or Graphical user interface

Description automatically generated with medium confidencethe headset moving faster than humanly achievable.

Figure 3 Example Data from Simulation, used in shiny dashboard as input data

## Motivation

The motivation for this project is to assist with analyzing data from trials to better inference results and draw conclusions from testing. I plan to use this program to help me identify trends or problems that occur through testing the simulations with doctors at the helm.

At the behest of Dr. Demirel, I made a Shiny Dashboard to display information in a pleasing manner that will allow for easy analysis and premade calculations. The dashboard was chosen because it encompasses most of what we wanted in terms of information display. The dashboard follows a format very close to that of HTML and has many premade packages to assist in development. It also can accept .csv files and run reactive code to better show results and allow for different files to be loaded one after the other.

## Information Gathered

A screenshot of a computer

Description automatically generated with medium confidenceThe .csv files come from a function call in a script on an object in Unity. When triggered, the c# script will compile the information stored and then print it to a .csv file. Those .csv files are what I use as input for my shiny dashboard. Each category of data has a separate tab on the sidebar for quick navigation. This separation is due to the differences between .csv files. Each part of the simulation is tracking something different, so there is little similarity between file organization.

Figure 4 Example of Trial Data Folder layout, outputted by the Simulation

# Problem and Methodology

## Problem

The problem to be solved with this project is quick data analysis from .csv files. The study that will be done soon, it is supposed to happen before Christmas but might be delayed till afterwards because of availability with the leadership of this grant, will give us many different people’s trials with the simulation.

The large amount of data is great but parsing and dissecting through it is time consuming and monotonous. That is where the idea for this project was birthed. Instead of taking the files and making excel sheets, I could implement R and its shiny dashboard to streamline the process of analysis. This data, along with data from other simulations that I will assist in, is intended to be used in my thesis, so having it pre-analyzed is very helpful and efficient.

## Methodology

Text

Description automatically generatedTo create this project, first the shiny dashboard must be made. This is a relatively simple process that is well documented through many resources, including those provided in the class’s canvas page. To best utilize the shiny and its dashboard, I also installed some packages to elevate the base structures and formats. These packages include ‘ggplot2’, ‘plotly’, and ‘DT’. The dashboard is comprised of the ‘ui’ and ‘server’ sections.

Figure 5 Libraries used for project

The ‘ui’ section handles the visual format of the dashboard. It offers many prebuilt structures to model the dashboard with and has plenty of packages that complement the built-in formats. I used the ‘ui’ to layout every aspect of the user’s interaction, from inputting a .csv file to navigating through the entire file inside of the dashboard. My ‘ui’ is separated into ‘tabitems’ to distinguish each uploaded file and its corresponding analysis parts. All of these ‘tabitems’ are listed in the ‘sidebarmenu’, that is a part of the dashboard’s sidebar, and labelled accordingly. Each ‘tabitem’ is organized in a similar manner. The objects/items that I want inside of each tab are within a ‘fluidRow’, a formatting tool that automatically formats objects/items into a row with and width of 12 screen-units. To make each object/item more identifiable, I placed them into their own respective ‘box’. This formatting tool allows for details such as a title, background, collapsibility, and desired width. Each of the objects/items that I want to be Text

Description automatically generateddisplayed to the user needs a unique name to be identified in the ‘server’ section. These names will all be strings, read as characters and words, and be used for tying the ‘server’ code portions to the output to the user.

Figure 6 Beginning 'ui' code showing each menu item in the sidebar

The ’server’ section handles any code parts that need either and input or display to the output. This section has the parameters ‘input’ and ‘output’ to ease development. The input parameter is for anything that is inputted into the dashboard, whether that be a .csv file, a slider, a text box, an interactive graph, etc. The output parameter is for what can be passed into the ‘ui’ section for the user to see.

Figure 7 Code for Server input/output parameters

My method of using the input parameter is with uploading a csv file[]. Since the dashboard is somewhat real-time, the functions that will be interacting with the user need to be ‘reactive’. This is a function that wraps around anything I wanted my code to do and makes it reactive to the user’s inputs and actions. The ‘reactive’ function is also widely incorporated into other packages that have data structures separate to base R and shiny dashboard. Often, this Text

Description automatically generatedincorporation will be understood if the function used has the word ‘render’ in its name.

Figure 8 Example code for accepting a .csv file as input

The code that I made to analyze the data from the sim is specialized to its section.

Text

Description automatically generated First, to analyze the hand movement, I decided to make a table for basic analysis, like mean, standard deviation, and variance. After reading in the .csv information into a data frame with the read.table() function, I can manipulate that data as I see fit. For these simple analysis functions, I start with cleaning data. The data that the haptic devices give for hand position can sometimes create erroneous points that are not possible to get due to the limited range of motion of the device. This means that I must clean any possible values that are invalid through sub-setting. Furthermore, since the .csv files don’t have any formatting, I manually rename each column’s names from the standard nomenclature of “V1, V2, V3, …” to a more apt representation with the names() function. Once cleaning and naming are finished, I created 2 new vectors to hold the name of the column and the other to hold the value of the calculations. These are combined into a matrix with the rbind() function and returned to the function call from either a reative() or render…() function to be printed in a tabular format.

Figure 9 Example of reactive code to take the input, clean it and then print a table

Text

Description automatically generated Next, I made a graphical representation of the positions over time recorded in a 3-axis graph with ‘plotly’. I do this by getting the data from read.table() again and then putting that data frame into the plot\_ly() function. I assign the x, y, and z to their respective columns as the parameters. To show the positions over time, I pipe the plot\_ly() function to an add\_markers() function where the color of the markers is based on the first column of the data frame, which corresponds to the time of the position. Doing all this outputs a graph with x, y, and z axis that is Chart, radar chart

Description automatically generatedmanipulatable in real-time on the dashboard.

Figure 10 Examples of plot\_ly graphical representation code

Figure 11 3D Plot graph of positional data over time recorded from the right haptic device

After that, I print the entire dataset in a searchable data table with ‘DT’. DT, or Data Tables, is a package that allows for very well formatted tables. I used this because it allows for the data set to be analyzed out of a .csv editor like Microsoft Excel. Additionally, it gives the opportunity to compare with the graphically equivalent data displayed before it.

To analyze the head/camera movement, similar processes take place. The .csv files for hands and head have records of the time, x position, y position, z position, x rotation, y rotation, and z rotation. With identical formats, the same code can be used but with different function and object/item names along with more specific cleaning.

The next data to analyze is the metric score .csv file. The metrics are steps of the operation that can be scored and graded, like if the doctor completed a step prior to another. Its .csv has only 3 columns, the time the metric was completed, the metric number and how many points were awarded for that metric. With such a limited amount of data, I print its entirety in tabular format. After that I search for which metric wasn’t a perfect score, through sub setting, Table

Description automatically generatedand print another table to inform the user of metrics they need to work on.

Figure 12 Data Table displaying information on a trial of the Right-Hand Haptic Data

Graphical user interface, application

Description automatically generatedThe last data to analyze is the quiz. Late into development, the doctors asked for a quiz to be administered before the simulation to gauge the user’s proficiency in the operation through knowledge before applicational skill checks. The quiz is comprised of a set number of steps that are required to be known by the head doctors at UT Southwestern. The goal is to place each question in its correct order to obtain maximum points. This .csv has only 2 columns, the step number and if it was placed in the correct order or not. Like the metrics, I can print this data entirely in a table and then place boxes of analysis after. One thing to note is that the quiz’s final score percentage is printed in the same column as the step number and an NA is in that row’s correctness checking. Chart, waterfall chart

Description automatically generated Since this is the case, I make sure to exclude the final row in the analysis of the data. The analysis is a bar graph that shows the correct vs. incorrect answers. This is done through a loop that compares if there is a true or false in the second column and then adds to counters that hold how many correct and incorrect answers there were. I use another plot\_ly() function with the x being Correct/Incorrect and the y being the counters. This plot has the type ‘bar’ and has the bars colored green for correct and red for incorrect. After the bar graph, the final grade is printed in a value box that looks for the last row of the .csv and prints the number in the first column.

Figure 13 The Dashboard will show what metric needs work

Figure 14 Display of Correct/Incorrect Quiz question totals and Final Grade

To cap off the code, the final line is the calling of the shinyApp() function passing in my ‘ui’ and ‘server’ as the parameters. When run, the code opens a new window to use the app. It can be opened in a browser if the computer conflicts with the running code to utilize the Text

Description automatically generatedbrowser’s support for R.

Figure 15 Example of how the reactive and render outputs are coded

## Data analysis

All data analysis is fully reproducible in this application/dashboard. As long as you have the specific .csv file for the tab, it will present you with the same data every time you run it. This does create a very limited number of uses for it, but that is the case for a program specifically tailored to my research and thesis.

Current analysis of the data that I have is not valid due to the study having not been ran yet. All I have is data of me testing the simulation to check that it functions properly for the doctors to provide feedback.

However, that doesn’t mean that this project is useless. On the contrary, this project is a very useful tool that can cut analysis time greatly when compared to other methods like excel or by hand. Having both a graphical representation of hand movements as well as a very robust data table gives great resources to visualize what would otherwise be overwhelming.

Graphical user interface, text, application, chat or text message

Description automatically generatedAs it currently stands, this project shows several analyses. The biggest thing is the 3D representation of hand positions. This data easily shows how long someone keeps their hand in a relative position. With this information, I can infer where rest positions lie or where the most time is taken in the simulation. This leads to multiple inferences on topics such as user friendliness, is the simulation too hard to understand at this part and thus the user is taking a long time to complete a step. Maybe the step is very important, and the user is taking their time to perform it as carefully as possible.

Figure 16 Sidebar with every menu item

Another analysis comes from the quiz. With the quiz being an indicator for the user’s knowledge about the procedure, different conclusions can be drawn from the relation of their quiz results to the actual metric results. Until the study has happened, I can only speculate that there will be a positive correlation between quiz scores and metric scores. One thing that I cannot know beforehand is if a longer simulation time will have a correlation with the metric and quiz scores.

The head movement data is something that I am interested in. I don’t know what kind of conclusions will be drawn from that data. I hypothesize that the amount the head moves will be related to the metric scores, but I am unsure in what way. Will more movement equate to a better score, or will the calm and knowledgeable lack of motion perform better?

## Future Work

With the fact that this project was made to assist me in the research for my thesis, there is little doubt that I will be adding to this throughout my remaining semesters.

First, since the operation is separated into a list of steps, or phases as the simulations has them called, I can include the phase number along with the time in the .csv print outs. This will allow me to better compare the different steps. It will especially make it easy to tell which step the user is on at what times. In addition, I could subset data based on that step number to better isolate portions of the procedure to scrutinize the data more accurately. Shiny dashboards can take advantage of that with the slider input to give the user more agency about what they are looking at with an interactive tool.

Afterward, I can add a tab system to compare the different simulations. The tab system would work with the boxes and be within the ‘tabitem’ that shiny dashboard has. It would be very similar to the sidebar tabs with its own input system and outputs that are saved even when going to another tab. Tabs will allow for more comparisons to be made in a quicker manner than having to run the dashboard more than once and compare what is being displayed on each.

Eventually, I might be able to connect the dashboard to the internet and have it connected to a database that Unity sends to automatically. Connecting Unity to the internet and having a database that is maintained somewhere is a lofty goal, but it would streamline the work process and remove the manual step of plugging in the files from my hard drive.

References

<https://rstudio.github.io/shinydashboard/index.html> (shinydashboard simple tutorial)

<https://www.youtube.com/watch?v=ORssNrhwp-k&ab_channel=Learndataanalyticswithus> (File Input for .csv files)

<https://www.tutorialspoint.com/r/r_csv_files.htm> (tutorial on using .csv files in R)

<https://shiny.rstudio.com/tutorial/written-tutorial/lesson3/> (R shiny widgets)

<https://plotly.com/r/3d-scatter-plots/> (Plotly 3D scatterplots)

<https://plotly-r.com/d-charts.html> (Plotly color markers)

<https://rstudio.github.io/DT/> (DT package and its usage)

<https://stackoverflow.com/questions/47505893/adding-a-vertical-and-horizontal-scroll-bar-to-the-dt-table-in-r-shiny> (Adding scrolling options to DT table)

<https://stackoverflow.com/questions/7531868/how-to-rename-a-single-column-in-a-data-frame> (Renaming data)

<http://www.sthda.com/english/wiki/ggplot2-pie-chart-quick-start-guide-r-software-and-data-visualization> (ggplot2 graphs - bar)

<https://plotly.com/r/bar-charts/> (plotly bar graphs)

Appendix

Appendix A (Code) –