CS CAPSTONE TECHNOLOGY REVIEW

FALL TERM

NOVEMBER 21, 2017

# KINECT BASED VIRTUAL THERAPY SOLUTION

PREPARED FOR

OSU HEALTHCARE SYSTEMS ENGINEERING LAB

MEHMET SERDAR KILINC

PREPARED BY

GROUP 24

LOUIS LEON

**Abstract**

This document outlines the technologies investigated and chosen for the execution of the Kinect Physical Therapy project. These technologies include the tracking sensor, organization of UI, and exercise recognition. The three are components of the Kinect Physical Therapy project. Each component will be further analyzed and have their options explored and discussed.

## CONTENTS

# 1 Introduction

## 1.1 Role in Project

My role in the project is going to be developing the user interface and exercise recognition.

## 1.2 What We Are Trying to Accomplish

We are trying to produce some software that is able to help users with Parkinson's disease perform in-home exercises prescribed by their physical therapists. My work will be divided into producing a simple-to-use user interface for the user and writing the exercise recognition code that will detect user movements and compare them to the correct motions.

# 2 Body Tracking Sensor

## 2.1 Overview

The sensor will be the means by which the user will be observed and act as the main interface between the user and the software. It should be noted that our client preferred us to utilize the Microsoft Kinect for this project.

## 2.2 Criteria

The criteria by which we will be comparing these options will be mainly on their cost and effectiveness. Moreover, how accurate the sensor is at detecting a user in front of it as well as its feature set. This includes space requirements, light requirements, and hardware requirements.

## 2.3 Potential Choices

1) **Microsoft Kinect v2 Sensor**
2) **ZED** — 2k stereo depth camera sensor
3) **OpenPose** — Not a device, but body tracking software

## 2.4 Discussion

**Comparison Breakdown**

**Cost:** Our first two choices cost in the range of under $500 and our third choice is open source software than can be obtained for free if used personally or $25,000 for a yearly commercial license. Specifically, the Microsoft Kinect v2 can be purchased for around $200 and the Stereo Labs ZED camera for $449 [1], [2], [3].

**Accuracy:** The Kinect v2 for Windows model allows for up to 25 nodes of body tracking and up to six bodies. It has a range of .5m to 4.5m and is capable of facial tracking, 3D scanning, and speech detection on a variety of different platforms and frameworks. The ZED Stereo camera has a range of 1.5m to 20m and can be used outdoors without needing infrared light for sensing. It works similar to how human eyes work, by the use of two cameras that estimate depth perception. OpenPose can use almost any camera to track bodies, faces, and hands. It can track multiple bodies with 18 keypoints per body and 21 keypoints per hand but offers no voice detection [1], [2], [3].

**Usability:**

- **Kinect**

  The Microsoft Kinect SDK works out-of-the-box with Microsoft application frameworks. It allows for easy use through the SDK which can be configured for full skeleton tracking and a wider field of view. This means a user can stand relatively close to the device. It works on Windows 8 and higher only, but the biggest downside is that Microsoft will no longer be producing them after 2017 [1].

- **ZED**

  The ZED Stereo camera offers 6-axis pose accuracy and high framerate and resolution output. It is compatible with different operating systems such as Windows 7 and higher or Linux. It also allows for third-party framework support from Unity, MATLAB, ROS, and OpenCV. However there is no SDK for body tracking. Any tracking of objects, hands, faces, or bodies would need to be implemented by the developer [2].

- **OpenPose**

  OpenPose is an incredible piece of open-source software that provides easy-to-use functionality once installed on a machine and works with a standard webcam or IP-camera. There are good options for data-capture and development with its API. It allows for saving and loading results in various formats including: JSON, XML, PNG, JPG. It also comes pre-packaged with a small graphical user interface and display for a simple result visualization. The downside of this software is the system requirements needed to run the program. The user is recommended to have sufficient hardware resources which may be too costly. Since there is no device, just software, there also needs to be a camera that is purchased. If decided to be used commercially, a license of $25,000 is required yearly [3].

### 2.5 Conclusion

For the purposes of this project, we decided the best option is the Microsoft Kinect v2 for Windows because of its ease of development. It has an official SDK developed by Microsoft with many tutorials on how to integrate and configure a lot of usable features such as voice detection. There are many online resources and projects that have used the Microsoft Kinect so the development of this project less time-consuming and relatively straight-forward. Another factor in our selection for this sensor is our clients initial request to use and produce a Kinect based solution.

### 2.6 Comparison Chart

| Technology | Comparison |
|---|---|
| Microsoft Kinect v2 for Windows [4] | • Easy to develop with<br>• Relatively low cost<br>• Body, face, hands tracking with good accuracy<br>• Voice Detection |
| Stereo Labs ZED Sensor [2] | • Can be used outdoors and indoors<br>• IR not needed to detect depth<br>• Works on Windows 7+ and Linux<br>• Higher cost<br>• No SDK for body tracking |
| OpenPose Software [3] | • No price if used for personal use<br>• Easy to use API<br>• Can be used on low-cost webcams and IP-cameras<br>• Accurate body, face, and hands tracking<br>• High system requirements |

## 3 Organization of UI

### 3.1 Overview

The organization of the UI will be very important for this project because of the nature of the user demographics. The UI will need to be presented in a very simple, easy-to-read fashion and have no confusing elements.

### 3.2 Criteria

The criteria for the organization of UI involves the level of simplicity. Specifically, how many elements will be displayed on-screen at any given time. The menu that the users see when they first start up the program needs to be straight-forward in order to avoid confusion. When the program provides feedback to the user when using the program and performing exercises, the feedback needs to be easily read and placed in the correct position.

### 3.3 Potential Choices

1) **Web Interface** — Through a browser
2) **Windows Presentation Foundation** — Framework compatible with Kinect
3) **WinAPI** — Part of the Windows operating system using only mouse and keyboard

### 3.4 Discussion

The idea of having a web interface for this project is interesting because it has the potential to make an application of this project more portable and useable remotely as well as highly customizable. The downside to this is the requirement for an internet connection and the potential for more security issues through web protocols. In contrast, our first two options do not require any internet connections and are hosted locally within the software.

Windows Presentation Foundation (WPF) provides developers with a unified programming model for building modern desktop applications. It is compatible with the Kinect and allows for helpful feature sets such as voice commands and gesture pre-sets. It is developed by Microsoft and is simple to set up with Visual Studio which will be the development environment we will be using for this project [5].

WinAPI has less human interactive features compared to WPF such as gestures and voice command recognition but is built directly into the windows operating system so it requires very minimal set-up time and is simple to develop with as well. Developer support is available in the form of a software development kit, SDK, which provides good documentation for using it. It is mainly for use with keyboard and mouse input which may not be ideal with users who have reduced movement in their hands [6].

### 3.5 Conclusion

We will use the Windows Presentation Foundation UI Framework for the development of this project. This UI framework is compatible with Kinect and with Visual Studio which already makes it quicker to learn and use. Microsofts developer network website provides samples that are explained thoroughly and documented well enough for new users to develop with. These examples will prove very useful when developing the user interface organization for this project. It also will allow us to implement voice commands and gestures for the navigation of our software. [5]

### 3.6 Comparison Chart

| Technology | Comparison |
|---|---|
| Web Interface | • Highly Customizable<br>• Wide Range of Implementation techniques<br>• Portable and remote<br>• Potential Security Issues<br>• Requires internet connection<br>• Longer development time |
| Windows Presentation Foundation [5] | • Easy to develop with<br>• Compatible with Kinect and Visual Studio<br>• Gesture supported<br>• Voice Detection supported<br>• Internet connection not required |
| WinAPI [6] | • Included in Windows operating system<br>• Keyboard and mouse support<br>• Minimal and fast<br>• No gesture support or voice commands<br>• Internet connection not required |

## 4 Exercise Recognition

### 4.1 Overview

Exercise recognition is the manner in which the software we create will be able to recognize a users movements and determine whether or not they are performing the correct movements.

### 4.2 Criteria

The criteria for this component should be how accurate the recognition is at recognizing both correct and incorrect movements. The speed of which it does this is also a factor because the system needs to recognize the movements quickly in order to provide timely feedback to the user. The third factor is versatility, how diverse of a movement may be recognized at any given time. The software should be able to recognize exercises involving multiple limb and joint movements.

### 4.3 Potential Choices

1) **Existing Library** — Created by an existing open-source project
2) **Custom Recognition** — We will be developing it from scratch
3) **Customize Existing Library** — Use existing project's implementation as template

### 4.4 Discussion

If we decide to go with the route to use an existing open-source library created from a similar project, we may be able to save a lot of time on the development. It is highly likely that the library may not have all of the movements defined that we are looking for, so we would have to use the ones that do exist. This method will most likely lead towards good performance and accuracy whereas developing the recognition from scratch ourselves may not result in speed and accuracy.

Developing the exercise recognition completely from scratch will allow for far more customization and control over the pre-defined exercises and data storage. This will take the longest to accomplish because we must write all of the code by ourselves, going through trial and error to achieve what is desired.

By using the third method of customizing and extending the functionality of an existing implementation, we have to potential to save more time on development compared to our second choice of developing from scratch. We will be able to use a template and write in any pre-defined and custom movements sets on top of what exists. Using the template, we may be able to see how fast the recognition occurs and we can utilize our own method of data capture.

### 4.5 Conclusion

The selection for exercise recognition will be done using the third option listed. Using a previously implemented project that uses exercise recognition as a template that can be customized and extended seems to be the best option. This will allow us to keep the functionality, speed, and accuracy while adding our own customization and adapt our own method for data capturing and storage. This method combines our first two options so the time it will take to develop will be decreased when compared to building it all ourselves.

### 4.6 Comparison Chart

| **Technology** | **Comparison** |
|---|---|
| Use Existing Library | • Significant decrease in development time<br>• Speed and accuracy<br>• May not contain all of the needed functionality<br>• Not very customizable |
| Develop Custom Recognition | • Highly customizable<br>• Allows for more control<br>• Longest development time<br>• Speed and accuracy is dependent on our ability |
| Extend Existing Library | • Customizable<br>• Decrease in development time<br>• Speed and accuracy relies on template implementation<br>• Combination of two above |

## REFERENCES

[1] "Kinect for windows sdk 2.0." https://www.microsoft.com/en-us/download/developer-tools.aspx. (Accessed on 11/13/2017).

[2] "Stereo labs zed 2k depth sensing camera." https://www.stereolabs.com/. (Accessed on 11/13/2017).

[3] "Openpose: real-time multi-person body tracking software." https://github.com/CMU-Perceptual-Computing-Lab/openpose, 2017. (Accessed on 11/13/2017).

[4] "Developing with kinect for windows." https://developer.microsoft.com/en-us/windows/kinect/develop. (Accessed on 11/13/2017).

[5] "Windows presentation foundation samples." https://msdn.microsoft.com/en-us/library/hh855377.aspx, 2017. (Accessed on 11/13/2017).

[6] "Winapi - graphical interfaces." https://msdn.microsoft.com/en-us/library/cc433218(VS.85).aspx. (Accessed on 11/13/2017).