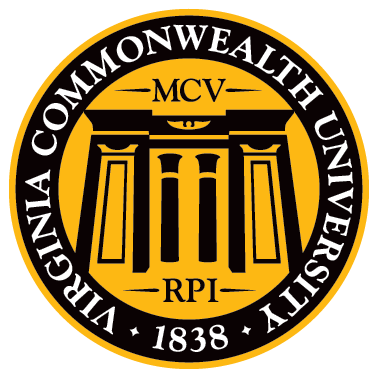
**Kinect Vision 360**

**Preliminary Design Report**

**February 11, 2016**



**Senior Design 2016**

**Joseph Crouch**

**Robert Ha**

**Ravideep Marwaha**

**Advisor:**

**Dr. Motai**

Table of Contents

[Executive Summary 3](#_Toc442889950)

[Problem Statement 4](#_Toc442889951)

[Proposed Solution 6](#_Toc442889952)

[Implementation Plan 12](#_Toc442889953)

[Testing and Project Management 16](#_Toc442889954)

[Conclusion 18](#_Toc442889955)

# Executive Summary

The Kinect Vision 360 team is designing a real-time data processing system. The design incorporates the Microsoft Kinect Sensor, which is a state-of-the-art commercial product that includes a depth imaging sensor, infrared emitter, color sensor, and a microphone array. The current state of the design will be further developed to integrate multiple Kinect sensors creating a larger field of vision and allow for communication between each separate sensor. The team will develop a graphical user interface(GUI) to manipulate the controls of the sensors and allow for visualization of each sensor’s data using streams. The project will appeal to a wide range of audience by integrating the camera and sensor data allowing for up to a 360-degree continuous collection and visualization. The team utilizes knowledge from electrical and computer engineering as well as human user interfacing and image processing. Currently, the hardware components include multiple Kinect sensors and a computer on which we do all our coding and testing. The implementation will be programmed in C#, using Microsoft Visual Studio(VS) integrated development environment to create the GUI. The team hopes to accomplish all tasks as planned and have a well-designed product by the Senior Design Expo.

# Problem Statement

1. Problem statement – a synopsis of the problem in your own words (not just a regurgitation of the project specs.) Why is this an important problem to address from a socio- technical perspective? If you are working with an industry partner, describe why this important to their industry. )

As engineering students, we understand that real-time digital image processing is becoming more important in today’s society. Our goal is to develop a system that can make our daily tasks easier as well as advance the image processing done by many machines. If we are able to create this configuration of the Kinect sensor, it provides solid evidence that complex commercial sensors can be manipulated and used as needed. Furthermore, if we are able to educate people about our work, they can grow from our roots and take this real-time imaging process to a whole new level; we want to provide the steps that show there is a whole lot we can all explore together.

This project is important and relative to advancing technology in today’s world as it uses image processing algorithms using data streams from the Kinect. Using the Kinect technology, we can accurately design and program implementations that would be used in the industry such as security, human interfacing, and new camera abilities. This means that we can possibly develop the Kinect sensor to high functionalities similar to machine learning and object following.

Real time image processing with the integration of Kinect sensors can change the way we see and interact with the world. By using multiple Kinect sensors we have already shown that a design is not limited by field of vision. We are able to get a 360-degree view of the world and collect data to process the images from every angle. During Super Bowl 50, a new technology was introduced allowing for a 360-degree view of the field with continuous and 3D view of the players and touchdowns. They were also able to create a visual graphic on screen to allow for people to understand that what they’re are seeing is the integration of dozens of cameras and sensors. Our project design is a smaller scaled and cheaper version of an innovative technology that is just now being implemented. Another example is that we could use our Kinect sensors to 3D image an object for 3D printing.

The examples above are only a couple ways we could use this design to view the world differently. We understand that this is out of our scope but this is one big example of where our project could be implemented. Machine learning algorithms combined with real-time image processing could be one of the greatest aspects of the Kinect system if we are able to implement it. If we are able to provide the base for it, we will consider our project a huge success.

# Proposed Solution

a) Concept overview of project – How does your design or design patterns address the problem.. If you have several design options your considering, how is it envisioned, what are the major functions of the project.

b) Functional diagram of the proposed architecture with justification. (Figure is mandatory). Description of the functions that need to be performed by the proposed solution

c)  Budgets for the proposed solution (cost, weight, power, area, etc.)

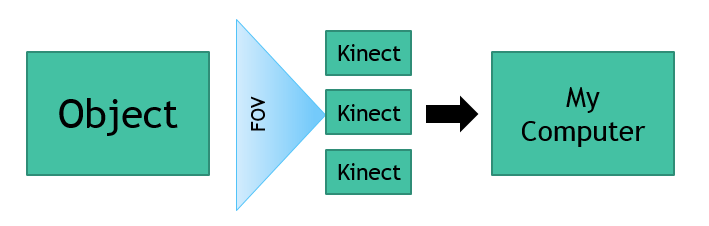
The initial steps of this project started with the idea of having a sensor find and track a specified object. We started looking into sensors that best do this but since we wanted a commercially popular sensor, we decided that it would be best to go with the first generation Kinect. As mentioned earlier, this sensor incorporates a depth sensor, infrared sensor, RGB sensor, and various other sensors. For our scope, we choose to focus on the depth and RGB sensor. We tested the Kinect 1 sensor versus the Kinect 2, a second generation sensor, and found that the Kinect 1 would be better to use because there are more documented sources and we have existing experience in working with it. Below, in Table 1, are the comparisons between an individual unit of Kinect 1 and Kinect 2.

***Table 1: Kinect 1 vs. Kinect 2***

|  |  |  |
| --- | --- | --- |
| Feature | Kinect 1 | Kinect 2 |
| Retail Price | $100 | $199 |
| Number of Units Available | 3 | 1 |
| Frames Per Second | 30fps | 30fps |
| Existing Code | Finished SDK, online resources | New SDK, limited resources |
| Depth Range | 0.4m - 4m | 0.4m - 4.5m |
| Field of View | 43° vertical by 57° horizontal | 60° vertical by 70° horizontal |
| RGB Resolution(pixels) | 1280 x 960 | 1920 x 1080 |
| Depth Resolution(pixels) | 320 x 240 | 512 x 424 |
| Infrared Resolution(pixels) | None (shared with RGB stream) | 512 x 424 |
| Maximum People Trackable | 2 | 6 |

As seen in the above fields, the biggest difference seen in favor of Kinect 1 is the retail price. The next most important aspect is that we have a finished and working SDK, as well as many online resources that can provide us good help. And the last deciding factor is that we have three Kinect 1 units available at hand, rather than one Kinect 2. We could order more Kinect 2 but that would hurt our budget a lot considering that we want multiple sensors to have a big field of view. We see that Kinect 2 has a much greater field of view and has much better resolution. Though this sounds fantastic, it is actually an aspect we want to stay away from. For the same reason that the Kinect 2 is very accurate, it means that the data stream is very heavy. We already have a big task at hand so it would actually be smarter for us to focus on a smaller data stream set and manipulate that. Perhaps in the future, this same implementation can be done using the Kinect 2. However, for now we will focus on the Kinect 1 and develop our project on that.

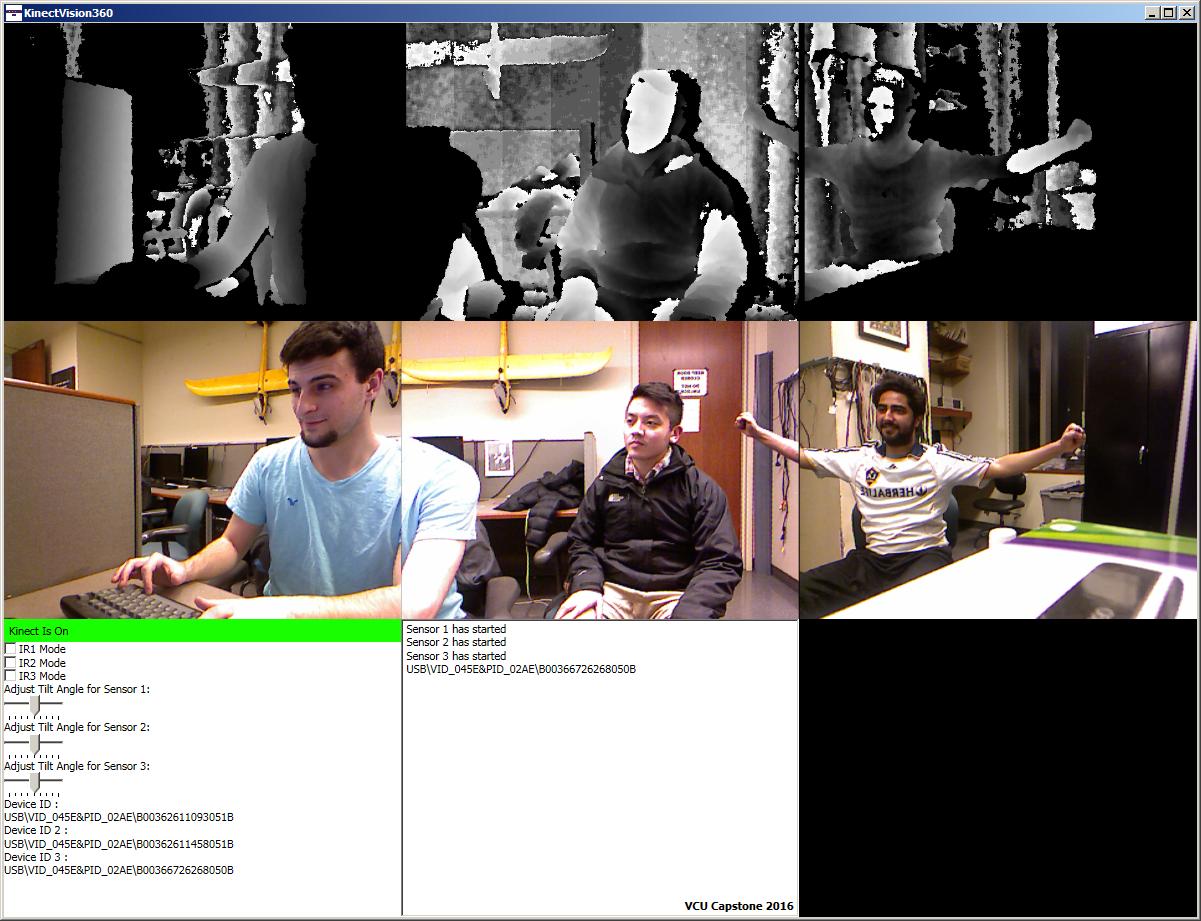
Upon further research within the university and outside resources, we found that Dr. Motai is more experienced with the first generation Kinect and happily decided to work with him to complete this project. Dr. Motai provided us with many Kinect 1 sensors and a workstation to work in in his research lab. We started having weekly meetings with Dr. Motai and began planning the outlook for the project. We discussed and researched to finally come to the conclusion that we will work with the Kinect 1 and try to make an object following system. Please refer to Figure 1 below to see the initial block diagram for this project.



***Figure 1: Initial Block Diagram for Project***

Dr. Motai advised us to first find all old code for the Kinect 1 on his old workstation computers and see what still compiles and runs. About a month of our time went into this phase in which we figured out that there is some old code but it doesn’t have the functionality we need. From there, we all decided that we must download the Microsoft Kinect Software Development Kit(SDK) and work with the demos to come up with our own GUI and implementation for this project.

The second phase of this project is to implement two Kinect 1 sensors together in order to have a larger field of view. Matter of fact, after a few hours of research, we were able to implement two Kinects together to get a field view of 114-degrees. Actually, we have three working Kinects right now on a computer that has three USB busses. After initializing two Kinect sensors, we knew that we could easily implement three because it was based off of the same code. The only big thing we had to do was add a Peripheral Component Interconnect(PCI) Express card to our computer to create another bus for the Kinect to be plugged into. As seen in Figure 2, right now have a full view of 171-degrees.



***Figure 2: GUI Screen shot with highlight of key elements***

We believe that this section of the project is vital to our success because it allows us to incorporate multiple Kinects and have a large field of view on a single interface. This is a major function of our project because now we are able to address our problem of image processing over multiple Kinects. This has been done before but we are taking a different route and will try to make a single cropped image so that all the data matches across all three Kinects.

The third phase of this project is to make the GUI for the Kinect so that we can see our results based on the data and multiple sensors. We were able to create an interface that allows three Kinects to work together in one image screen and show a larger field of view. At the moment, using all three Kinects, we have a GUI that shows the depth, the color image, the console stream, as well as other small elements such as changing the vertical tilt on the Kinect sensors. This GUI output can be seen in Figure 3 below. One of the main components of the GUI is to have that console stream working properly. Currently, we are able to print out to the console stream but we need to find a way to show the Kinect sensor data stream, such as the color image bit stream. Once we are able to do this, we need to format the data properly so that we can interpret it and use it for our system functions.

After all that is completed, we need to use that data stream to do our real-time image processing. This work consists the major portion of our project. Up till now, we have basically been setting up our system in the simplest and cleanest way possible. In these phases, we will need to develop a method that allows us to clean out dirty data from the data stream. It is important for us to do this for obvious reasons; hopefully, the Kinect is already so robust that little work will be needed here. The main reason we do this is because we need to make our GUI color image cropped into one image. Right now, as seen in figures above, all three Kinects have their own windows but match up because the windows are next to each other. This is acceptable but we need to manipulate the data stream so that we can combine the color bit stream in order to make one overlapped or cropped image. This will look good visually and provide us cleaner streams. Figure 5 below displays our goal for this phase.

The next phases of this project will be to incorporate the servo and figure out how to follow an object. There are many online resources and Microsoft Demos for the Kinect where tracking has been done. Additionally, we have experience in writing algorithms for tracking. Therefore, this step should be fairly simple and easy to implement. The key feature about this phase is to have the ability to track over all the sensors continuously. We want to pass on the data from sensor to sensor and make sure we don’t lose any vital information. We consider using a time driven system so that we can continuously sample at an acceptable rate and not lose any synchronization. However, we are still researching in this area and do not have any solid numbers to display.

The last and final phases of this project are to complete the poster and documents for Senior Design and also have our working prototype. We want to code to be very robust and well documented. Based off our timeline presented above, we should be at this step by week 7.

In terms of budgeting and risks, in the previous semester we realized that we have to make sure that we are careful with our spending. Though we have all the Kinects provided from Dr. Motai, we understand that the market value for each sensor is nearly $100. We haven’t had to buy any extra sensors but it would be very beneficial for us to make sure we don’t spend too much. Furthermore, if we plan on integrating multiple Kinects on one computer, we have to make sure we have enough USB busses on the computer to handle multiple Kinects. Buying a PCI card is the best solution to make sure we can do three sensors at a time and each PCI card runs at about $10.

All in all, the reason we chose this project is because we know how popular the Kinect sensor is and how much we can do it. It will be a huge step in the technology field at VCU if we are able to figure out how to make the field of view for the Kinect larger and implement that without any programs crashing. There is technology that has this working but is not so successful. We wish to go past them and grow the object-following systems even more.

# Implementation Plan

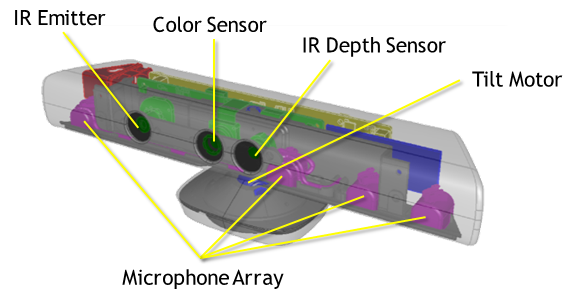
a)   Description of how the proposed architecture will be partitioned into subsystems for implementation - include a description of the interfaces between subsystems including functional or block diagrams

b) a description of the technology (FPGA, PLD, microcontroller with software, mechanical system, etc.) that will be used to implement each subsystem - include detailed part numbers, data sheets, and any initial design work such as schematics and simulation results where applicable

c) Schedule of implementation, based on partitioning above, and with narrative of progression (include Gantt chart or similar)

Based off of Figure 1, we see that this system has three subsystems. The core subsystem of this design will be the computer. The computer is responsible for all the data processing and provides as the medium through which all subsystems are connected. It is necessary to use the computer to interface with the and also having a script that shows the data processing in real-time. This continuous integration of the system is a key part in our project. The next most important subsystem is the Kinect sensor itself. This is the biggest sensor in our platform and provides all the data that must be manipulated and understood.

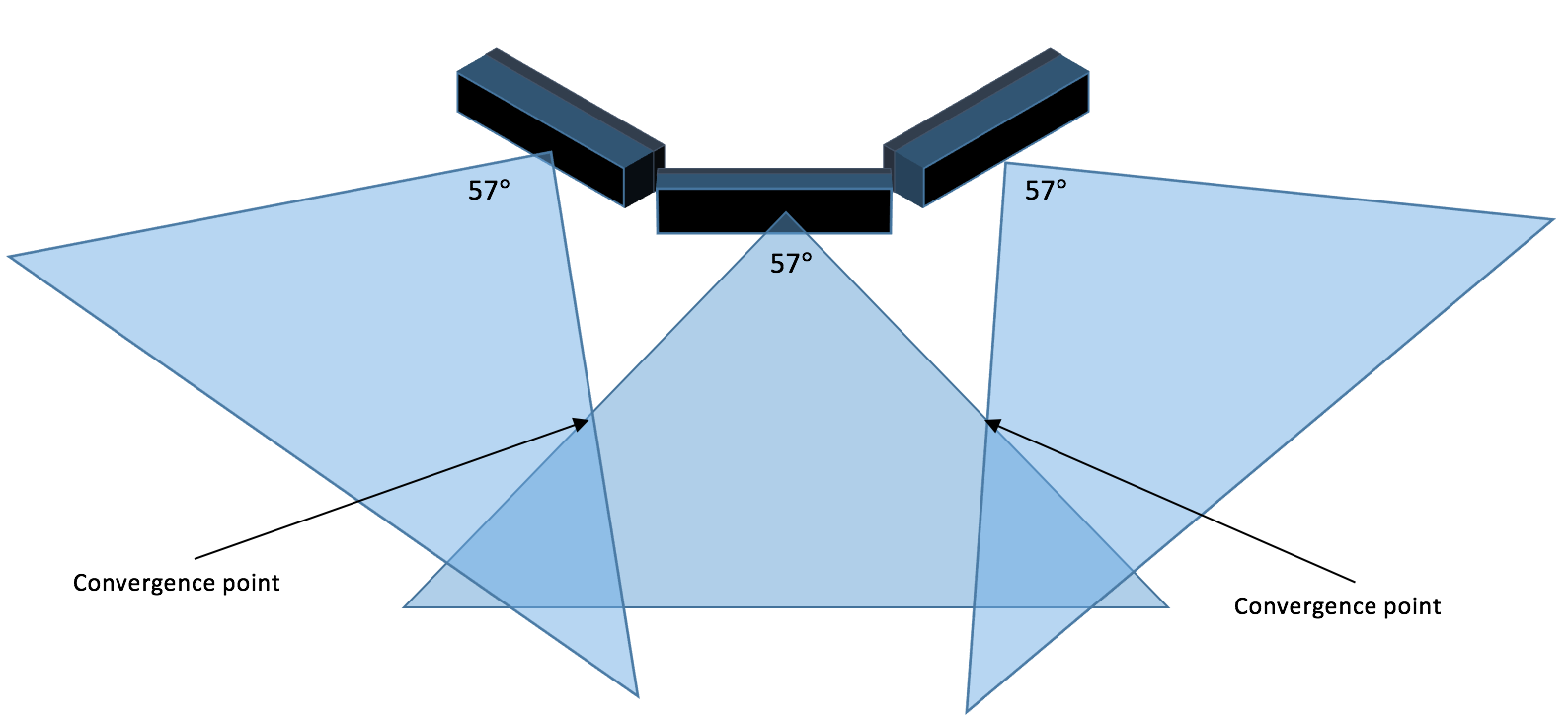
First, each Kinect sensor contains an Infrared(IR) emitter, a color sensor, IR depth sensor, a tilt motor, and a microphone array. These are visually presented in Figure 2 below. Additionally, the Kinect sensor has the following detailed specifications listed in Table 2. To go more in depth about the sensor, the viewing angle for each sensor is 57-degrees which is increased by our integration of three Kinect sensors. The three Kinect sensors will be constructed as seen in Figure 3. The horizontal field of vision is increased to about 171-degrees. Since each sensor’s field of vision overlaps we have two convergence points. At this point we will use open source and custom designed algorithms to handle the data overlap and merge them together. The frame rate for each sensor (depth and color) is 30 FPS. The IR emitter emits infrared light beams and the IR depth sensor reads the IR beams reflected back to the sensor. The reflected beams are converted into depth information measuring the distance between an object and the sensor. This makes capturing a depth image possible. All this may change as we integrate all three sensors to communicate with each other, since each sensor will have to be polled to gather the data individually. We are currently not using any audio features of the Kinect at this time, although we may implement usability in the future.



***Figure 2: Sensors on the Kinect 1***

***Table 2: Kinect specifications from Microsoft.***

|  |  |
| --- | --- |
| Kinect 1st Generation | Array Specifications |
| Viewing angle | 43° vertical by 57° horizontal field of view |
| Vertical tilt range | ±27° |
| Frame rate (depth and color stream) | 30 frames per second (FPS) |
| Audio format | 16-kHz, 24-bit mono pulse code modulation (PCM) |
| Audio input characteristics | A four-microphone array with 24-bit analog-to-digital converter (ADC) |
| Accelerometer characteristics | A 2G/4G/8G accelerometer configure |

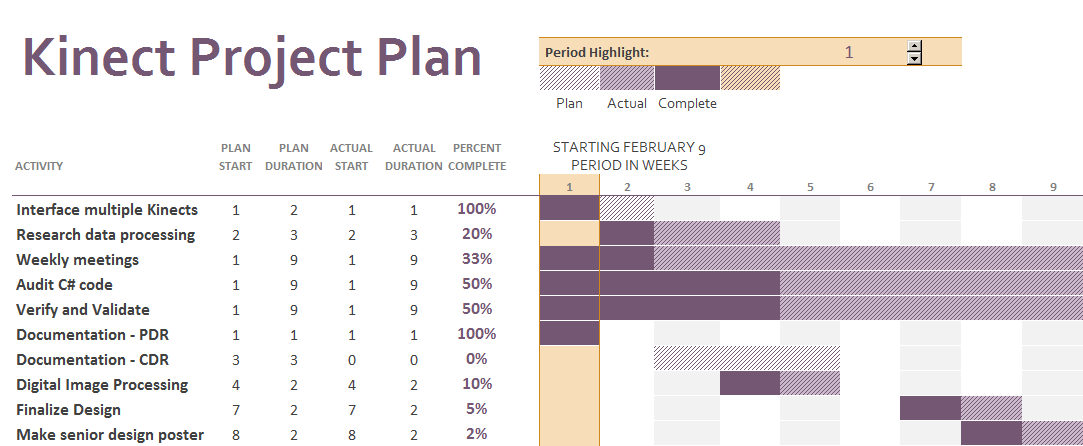


***Figure 3: Diagram of each sensor’s field of vision and convergence points.***

The computer we run VS, test, and implement the Kinect sensor has the following specifications:

* RAM: 16GB
* Intel core i7 CPU at 2.39Ghz
* 64bit OS
* Windows 7

The following Figure 5 is an overview of our project planning. Prior to designing the project we had to do preliminary research on the feasibility of the project. Our first major task was to integrate multiple Kinect sensors, which we completed. We created a GUI to handle the controls and settings of the Kinect. We also needed to visually verify that the Kinect sensors we outputting data. Our next major text is to get the Kinect sensors to communicate with each other. We have already started to research algorithms to handle the digital image processing. We also have meetings weekly with Dr. Motai in order to continue our guidance through the project. Each week we present our progression and discuss any encountered problems and future tasks.



***Figure 5: Gantt chart for Project Planning***

# Testing and Project Management

d)  testing plan describing how each subsystem will be tested independently and how they will be further tested as assembled units

e) division of labor plan describing how subsystems will be divided among group members for

implementation

f) documentation plan describing how the design will be documented as the design progresses - this is intended to help group coordination and help prepare for design reviews and final report

g) risk management plan describing the high risk areas (specific areas of the system described above, not general areas like “time budgeting,” “socket programming,” etc.) of the proposed solution and how the risks will be minimized by such techniques as early in

Currently, the Kinects are tested on startup using the VS application developed. When the GUI starts it checks for and initializes each Kinect and maps them to the correct portion of the GUI. The status of each Kinect is printed to the on GUI console. This allows users to ensure the Kinects are initialized correctly.

The GUI is the main source for visual confirmation that the program is working as designed. Each Kinect’s depth, RGB, or infrared sensors are streamed onto the GUI in real time. This assures they are visually working as designed. Moreover, implementing human or object tracking can be visually seen within each stream on the GUI. The tracking is shown via a simple box around a person or object. This visually tests that the code is implemented correctly.

The division of labor is spread evenly amongst each partner of the group. The majority of the work involved is within the developed VS's code and Kinect SDK. Considering this situation, the development is evenly spread across each member. Some work could be working on separate functions within the code or even developing within the same function. Ultimately, communication is key in working together and ensuring work is made without repetition and confusion. Figure 5 above shows the work needed for successful completion of this project.

The VS solution file is shared via public GitHub account. Only members of the group are allowed to make adjustments to these files. GitHub allows the group members to pull, push, and commit all current code to ensure each member has the most up to date code. The GitHub code also will always have the most recent working code. Documentation and various source files are also shared amongst the GitHub repository to keep all project related data in one area. Using GitHub also allows the project to be downloaded onto different devices with an internet connection. Documents such as reports and design plans are edited on a Google account so everyone can work on the files at the same exact time in order to help one another make adjustments as needed. The final documents are placed onto the GitHub repository.

Risk limitations for the project is more than likely the amount of data being polled and how the data is being processed. Kinect systems take in a lot of data across the RGB, infrared, and depth sensors. The number one priority, other than completion of the project, is to make sure our code is only using and grabbing data that is relevant. With all the examples and libraries within the SDK it can be easy to occasionally include functions that are not needed. Risking slowing down the system can be detrimental in a real time system. This is why using only relevant and clean data at the rate needed is crucial in managing possible risks.

# Conclusion

In conclusion, real-time image processing is the field we want to explore with this project. We have made a lot of progress in terms of setting up our system, however we still need to make sure we can interpret the data properly. This project involves a lot of C# coding and deciphering between weak and strong data. In our case, the reason we want to use a time-driven system is so that we can sample the Kinect sensors at a certain frequency and capture the relevant data. This would provide us new data at every sample and would allow us to keep our system idempotent. These two factors are key because if we had event driven, we could lose our synchronization between the receiver and transmitter, i.e. the computer and Kinect, respectively. Keeping our system idempotent allows us to make sure if we have multiple copies of the same sample data, we can consider it only as one copy and then move forward from there. We will always need to check temporal correctness of our system which we will most likely do by putting time stamps on each data stream we show on the console. Therefore, we will be able to validate and verify that our project is sufficient enough to deliver real-time image processing creating cutting-edge technology.

before submitting…

make sure all pronouns are set properly.

fix all the figure names and numbers

update the table of contents

no weird formatting

spell check

i hate this shit. quitting school now!