Oregon State University
College of Engineering

CS Capstone Final Report

June 12, 2017

Object Velocity Tracking

Prepared for

Alex Neighbors

Prepared by

Group 37

Alex Bailey
Dylan Washburne
Benjamin Wick

Abstract

This document is the final written document for our capstone project. It will give an overview of everything we did this year.

CONTENTS

1 Introduction

This project was requested by our client Alex Neighbors. This was a personal project the he had envisioned. The goal was to create a proof of concept of an alternative method of speed tracking. In our case, our alternative was a video camera. Specifically, the Microsoft Kinect. The members of the team included Alex Bailey, Dylan Washburne, and Benjamin Wick. We all had similar roles in the project however, we all specialized in different areas when contributing. Alex focused on hardware connectivity and the speed algorithm. Dylan's main focus was the speed algorithm. Finally, Ben focused on the computer vision library and object tracking. Our client Alex, played the role of our supervisor and was there to help along the way.

2 Requirements Document

(Begins on next page.)

Object Velocity Tracking

Requirements Document

Alex Bailey, Ben Wick, Dylan Washburne

CS 461, Fall Term

Abstract

Using a stationary camera, we are attempting to detect objects and determine the velocities of those objects relative to the Observer (camera). This will be done by having the camera recognize a specific type of object, yet to be determined, in space and determine their velocities based on the rate at which they travel through the frame. The speed of the objects will be displayed on a computer application window. To make this work, we will have to research the varieties of cameras available to use, as well as the APIs they operate with. We will review the available computer vision software and determine which is the most appropriate for our needs. We will also design a computer application that will be the user interface for the product. From these we will create an object tracking program that will not have the shortcomings of the current object tracking methods, such as being only able to track one object and having trouble in the rain.

I. Introduction

A. Purpose

The purpose of this document is to present a detailed description of the requirements for the ”Video Radar” software. This document is intended for the main use of the client, as well as the professor and the teacher assistants and will be proposed to the client for its approval.

B. Scope

Our product, the ”Video Radar”, will be able to identify a specific type of object, such as a person or a car, calculate the object’s velocity, then display the velocity on the screen. Our product will be beneficial over other products in that it can function unmanned, will specify which target is being tracked, and will be able to track multiple objects.

C. Definitions, Acronyms, and Abbreviations

Term	**Definition**
API (Application Program Interface)	A particular set of rules and specifications that software programs can follow to communicate with each other.
User	Someone who is interacting with the software.
Object	The entity being tracked by the video feed.

D. Overview

The rest of the document contains two additional sections. The first section is the overall description. This section will describe the intended use of the software and give background. The last section is the specific requirements section. This section contains all the software requirements.

II. Overall Description

A. Product Perspective

Our product will be a self-contained product. Our product’s interface will consist of a window where the user can activate the camera and velocity tracking. There will be a menu bar in the application that will allow the user to specify aspects of the product, as shown by Section C in the picture A below. There will be a table on the side of the application, Section D of picture A, that will have a list objects being tracked and their speed. Our products application window will consist of a small button row at the bottom for essential buttons, such as the start and stop buttons, as depicted in Section B of picture A below. The application will have a central section for displaying the video captured by the camera with our information overlay, such as Section A of picture A below. We will be using computer vision software in order to identify the type of object and track its location across the frame. The product will have only one mode of operation, the on mode, where the product constantly processes the images from the camera and velocity is displayed. This mode requires no input from the user except to turn off. The camera will not have any automated motion. The user will have to manually adjust the camera. The camera will have a fixed zoom that the user will not be able to change.

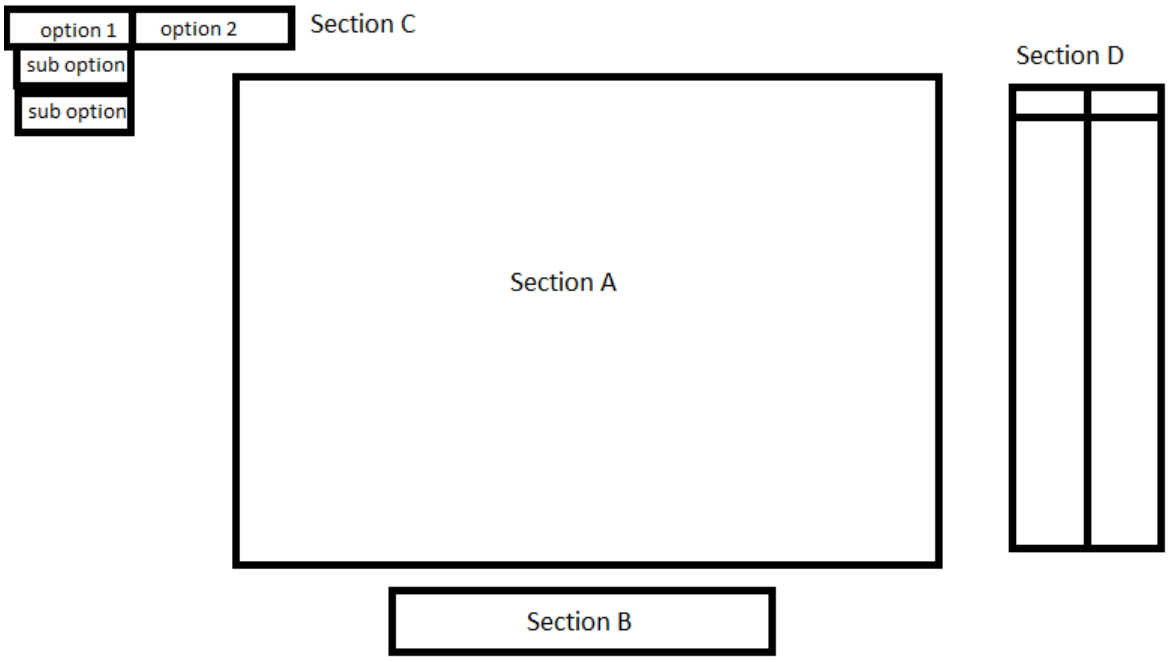


Figure 1. Picture A: Mock-Up of the Application Window

B. Product Functions

The software will be connected to a camera with a live video feed. It will then be able to detect objects that are specified for the users needs. The software will then be able to acquire the velocity at which the objects are traveling.

C. User Characteristics

This software is intended to be used by many different users. Because we have yet to decide on the specific type of object to track, the education level and experience. However, because our product is meant to replace current methods of object velocity tracking, such as the radar gun, our intended users will not need a high education or experience level. In fact it will probably be expected taht minimal training is needed to operate our product.

In order to examine the possible users for our product, we will examine possible object types.

If we decide to use cars as our object type, the main users would be police officers, to be used in enforcing speed limits. In this case the user would have good education level, but likely not high technical expertise.

If we decide to use people as our object type, then the main users will likely be analysists, analysis human movement patterns and speeds in various contexts such as sports or emergency evacuation. In this case, our users will be highly educated, but they too would likely not have high technical expertise.

D. Constraints

This product will likely require a nontrivial amount of resources to perform its task at the constant interval we require. As a result, our product will either need to come with a dedicated computer to do the processing, or it will have to interface with existing computers which we expect to be in the locations of use.

In accordance with federal laws, the camera is allowed to be recording anything described as ”within plain sight”. The proper use of this falls on the user, and must be disclaimed before use.

The product’s reliability of use can be described in a number of ways. The camera should be recording at a constant and stable rate. The objects in the scene should be properly identified with 70% accuracy. The tracking of an identified object’s velocity should be within 90% of the objects actual velocity. The velocity tracking should also be reliably given every 0.5 seconds, with 30 frames per second 90% of the time.

While the video recorded is protected by federal laws, the information nonetheless must meet certain security standards. If the video data is to be saved locally, it should also respect encryption of the product it is saved on.

E. Assumptions and Dependencies

We assume that if our products camera needs to move that our product would be able to either handle the motion of the camera and still be able to track velocities or be mounted securely enough to minimize motion of the camera, allowing the product to track velocities. We assume that the computer vision software will be able to recognize an object within enough time after entering the frame so that there will be enough time for the velocity algorithm to calculate the velocity.

III. Specific Requirements

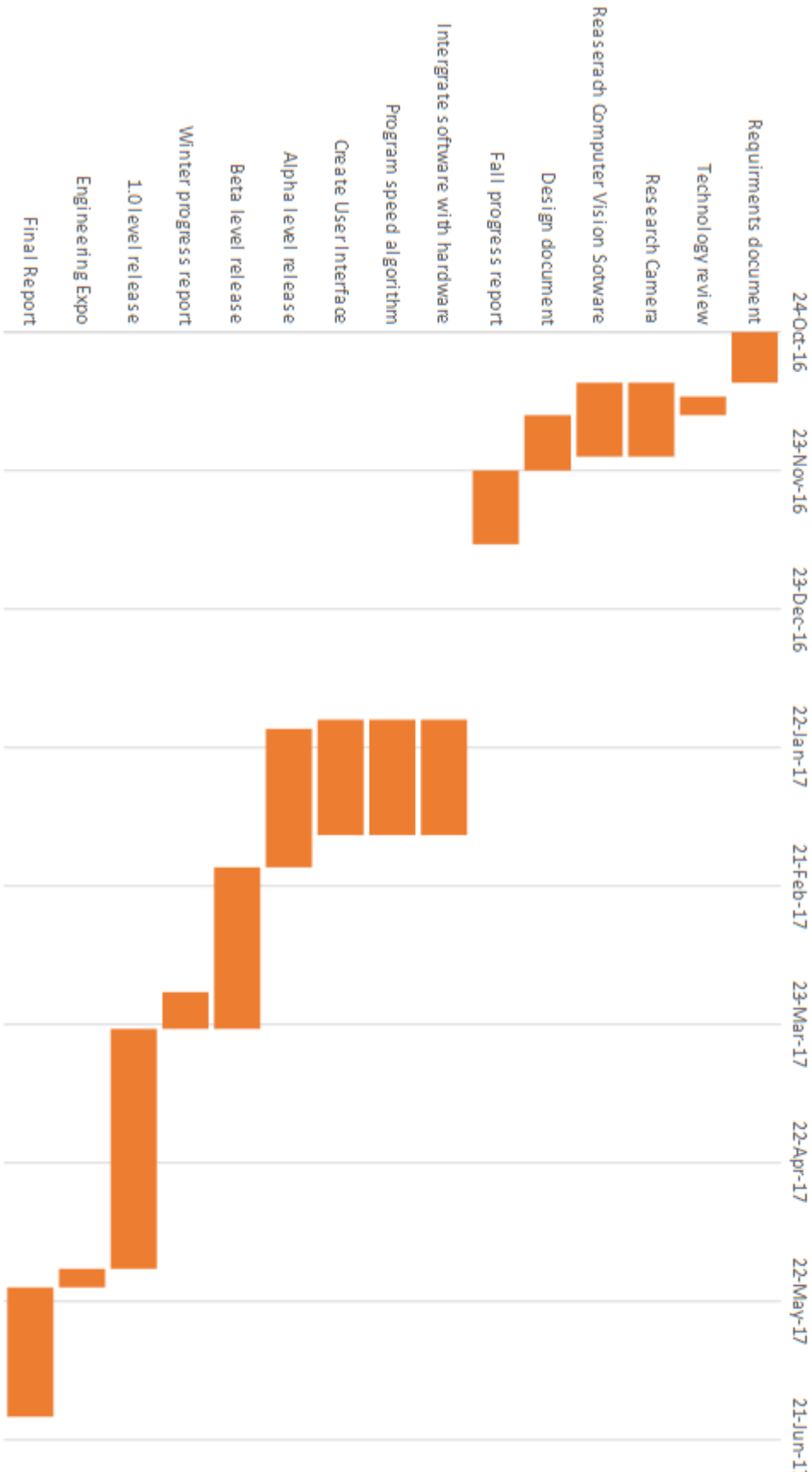
The product must be able to identify objects in frame, and differentiate them from the background. When the product identifies an object as the desired type, at least 70% of the time this must actually be the desired object type.

The product must be able to track the velocities of identified objects. For each object identified in the scene, the velocity the program displays to the user must be within 90% of the objects true velocity.

The product must have the capability to identify and track multiple objects simultaneously. It must be able to identify and track at least 4 objects simultaneously. At the same time, should the proper conditions present themselves, the product should be able to automatically identify as many objects it can locate in frame, should that number exceed 4.

The product should be able to perform its velocity analysis on objects that are moving in various directions. While it should track the velocities of objects moving perpendicular to the camera, it should also be able to identify the velocities of objects moving parallel to the camera, or at any variation in between. The velocities returned are within 90% of the objects’ true speeds, as stated above, however direction should not cause the product to fail its purpose.

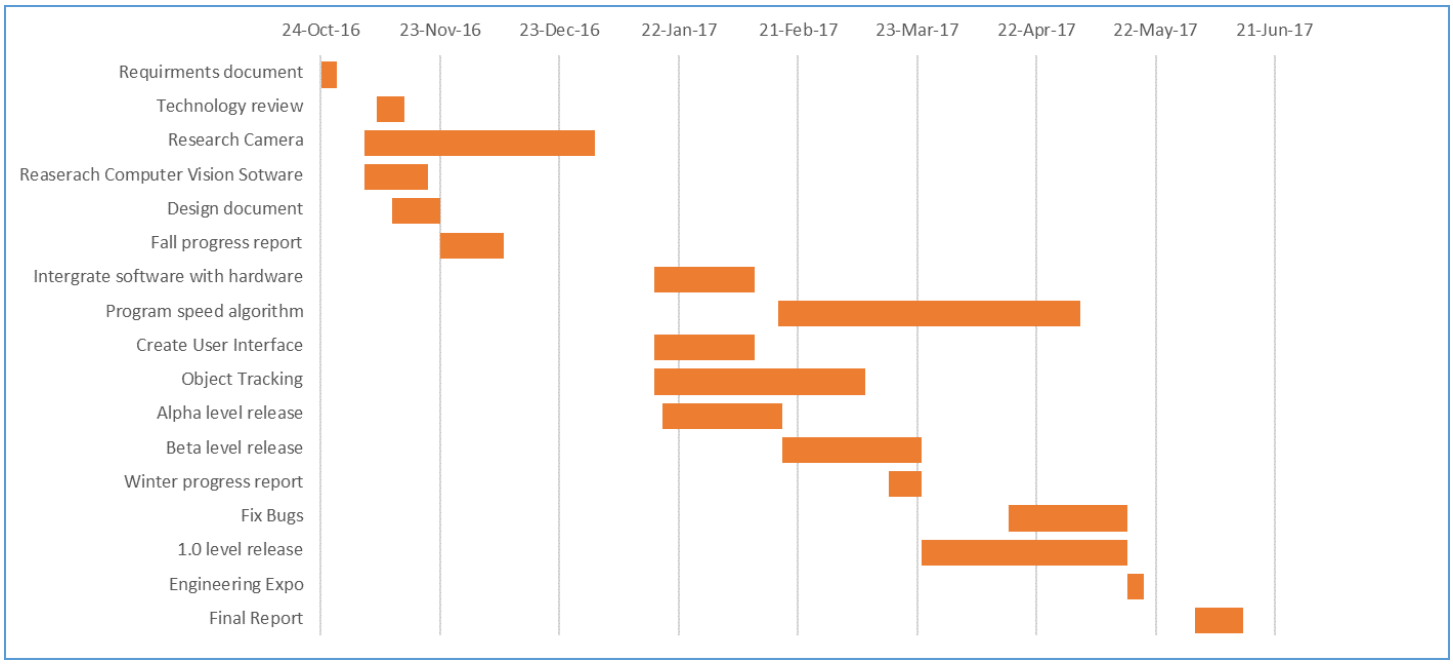
The product will accurately work within 100m of the camera’s placement. Objects within 100m of the camera’s placement, within its view angle and with an unobstructed line of sight, should all be able to be identified by the camera, reflecting the accuracy requirements described above. Objects beyond the maximum effective range can potentially still be identified if the conditions allow, but are not granted the same accuracy guarantee. Our product’s interface will consist of a window where the user can activate the camera and velocity tracking as well as view the camera input live, with squares surrounding each object being tracked and speeds posted below the squares. There will be a menu bar in the application that will allow the user to specify aspects of the product, as shown by Section C in the picture A above. There will be a table on the side of the application, Section D of picture A, that will have a list objects being tracked and their speed. These values will be saved to a comma separated value text document for later analysis. Our products application window will consist of a small button row at the bottom for essential buttons, such as the start and stop buttons, as depicted in Section B of picture A above. The application will have a central section for displaying the video captured by the camera with our information overlay, such as Section A of picture A above.


Requirments document
Technology review
Research Camera
Reaserach Computer Vision Sotware
Design document
Fall progress report
Intergrate software with hardware
Program speed algorithm
Create User Interface
Alpha level release
Beta level release
Winter progress report
1.0 level release
Engineering Expo
Final Report
24-Oct-16
23-Nov-16
23-Dec-16
22-Jan-17
21-Feb-17
23-Mar-17
22-Apr-17
22-May-17
21-Jun-17

2.1 Changes Made

Number	Requirement	What happened to it	Comments
1	Track multiple objects	Changed to single object	Our client informed us that this project was intended on being a proof of concept. We decided to first focus on tracking one object at a time.
2	Work up to 100 meters	Changed to 10 meters	At first we were unaware that depending on our camera we would be very limited. We ended up using the Microsoft Kinect which has a max range of 10 meters.

2.2 Final Gantt Chart



3 Design Document

(Begins on next page.)

Oregon State University
College of Engineering

CS Capstone Design Document

February 18, 2017

Object Velocity Tracking

Prepared for

Alex Neighbors _____ Signature _____ Date

Prepared by

Group 37

Alex Bailey _____ Signature _____ Date

Dylan Washburne _____ Signature _____ Date

Benjamin Wick _____ Signature _____ Date

Abstract

The Object Speed Tracking system incorporates many different technologies to make it possible. This includes computer vision libraries, object tracking methods, speed algorithms, and the user interface. This document shows the intended design for the system and how it will incorporate the technology need to meet the system requirements.

CONTENTS

1 INTRODUCTION

1.1 Purpose

The purpose of this document is to give the necessary and required information to effectively define our systems design and give our team guidance to execute the system throughout the implementation process.

1.2 Scope

This system intends to utilize a live video feed in order to determine the speed of an object. The software will be able to identify a specific type of object, such as a person, or RC car and then calculate the object's speed and display it to the user on a windows application. This offers an alternative method to track speeds of moving objects.

1.3 Context

The Object Speed Tracking system will be a windows application that utilizes a Microsoft Xbox 360 Kinect camera along with a computer vision library to track and calculate the speed of moving objects. This project is intended for a class at Oregon State University. Our goal is to implement the project design. Future development plans will be based on feature needs and success of project determined by the client.

1.4 Glossary

Term	Definition
API (Application Program Interface)	A particular set of rules and specifications that software programs can follow to communicate with each other.
Computer vision	The ability for the computer to extract, analyze and understand from an image or video.
Object	The entity being tracked by the video feed.
Kinect camera	A type of camera that has in infrared camera and color camera as well as a infrared light projector.
User Interface (UI)	The visual part of the application in which the user will interact with.

2 BODY

2.1 Identified Stakeholders

The main stakeholder for our product is Alex Neighbors, as the client of our project. Alex Neighbors' concerns for our product is to create a product that is able to track objects and calculate the speed. This project is a proof of concept for which in the future, more different systems can be used and our project can be enhanced to fit the needs of others.

There are many different applications and users that can use our system who would be identified as stakeholders. We will now discuss possible object types, the stakeholders depending on the type and their concerns. Law enforcement would be a possible stakeholder for this technology as it could be used in place of current speed limit enforcement technology. Their concerns could be how many vehicles it can track at a time or the accuracy of the velocity.

Sports teams and sports analysts could be stakeholders for analyzing how the players are moving. Anyone group trying to study the movement of groups of people, such as someone trying to see how fast people are able to evacuate in an emergency, could be a stakeholder. Their concerns could be the speed of the calculation or speed and consistent of the frame-rate.

2.2 Interaction Viewpoint

2.2.1 Viewpoint Description

The Interaction Viewpoint describes how the different pieces of our product work together and what is sent back and forth between them.

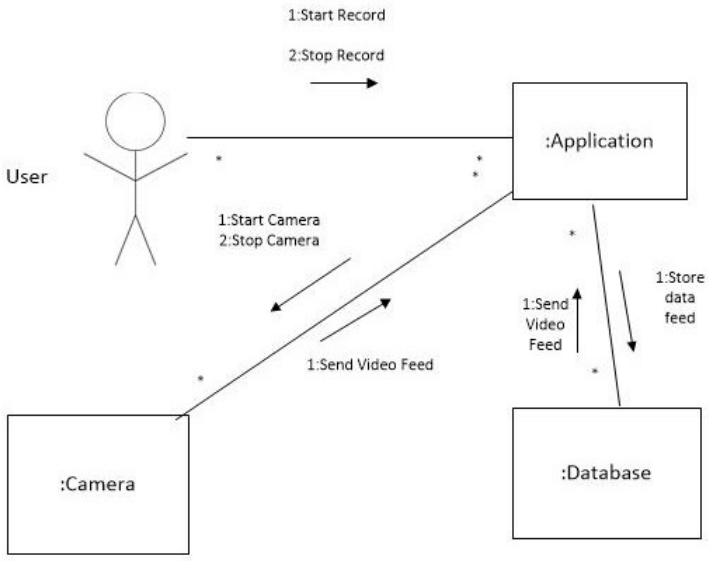


Fig. 1. UML Communication Diagram

2.2.2 Design Concerns

The concerns for this Viewpoint are how the different components of our product will be interacting, which components will be interacting with each other, and what information will be passed between each component.

2.2.3 Design Elements

There are four elements in this viewpoint. The first is the user of the product. To keep the user interface simple the user will be able to start the recording and stop the recording. The second element is the application. This will be run on a computer of the user choice. The application will handle the interactions with the camera and database. It will send the start and stop signals to the camera and send the video for storage to the database. The application will also be able to receive a video stream from the database in order to recalculate the data. The third element is the camera. This will most likely be connected to the computer running the application, but not necessarily. The camera will record video when signaled and send the video feed to the application. The fourth element is the database. This will most likely be run on the same computer running the application, but not necessarily. This element will receive the video for storage from the application and will also send a video to the application for recalculating the data.

2.3 Information Viewpoint

2.3.1 Viewpoint Description

The purpose of the Information viewpoint is to describe how the data will be stored for the video radar software. The goal is to maximize efficiency and accuracy. There are two main entities within this viewpoint which include video

storage and results storage.

2.3.2 Design Concerns

Concerns with information include storing the correct data, data management strategies and data access schemes.

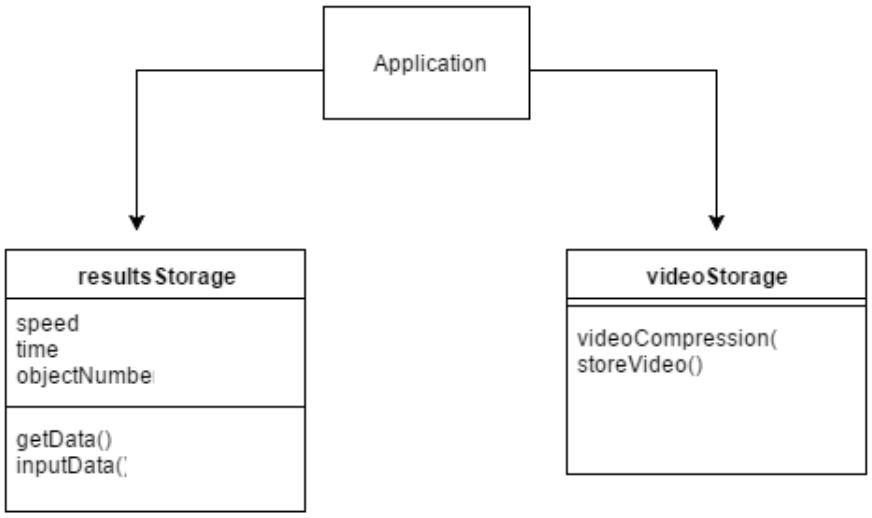


Fig. 2. Flowchart diagram for information storage.

2.3.3 Design Elements

The user will interact directly with the application. Once setting up the system, the Kinect camera will identify the distance to objects and then the application will calculate the speed at which it is moving. This information is displayed to the user on the screen as well as stored in a separate text file. The first element is the video storage. The live video feed will have be captured and stored. To do so we will implement a video capture class that is provided in the OpenCV library. This class will capture the live video feed and store the video in MP4 format into a file. The second element includes the final results being stored. The final results will be stored into a text file. The data that will be stored includes the object number for reference with the video, the speed of the object, and the time the object's speed was calculated. This will be implemented with a class called "storeResults". This class will extract the data needed from the calculation classes and the video feed and write it into a separate text file.

2.4 Context Viewpoint

2.4.1 Viewpoint Description

The Context Viewpoint describes the specific details of our project which are vital to its overall success, while sometimes not applying to any singular other category.

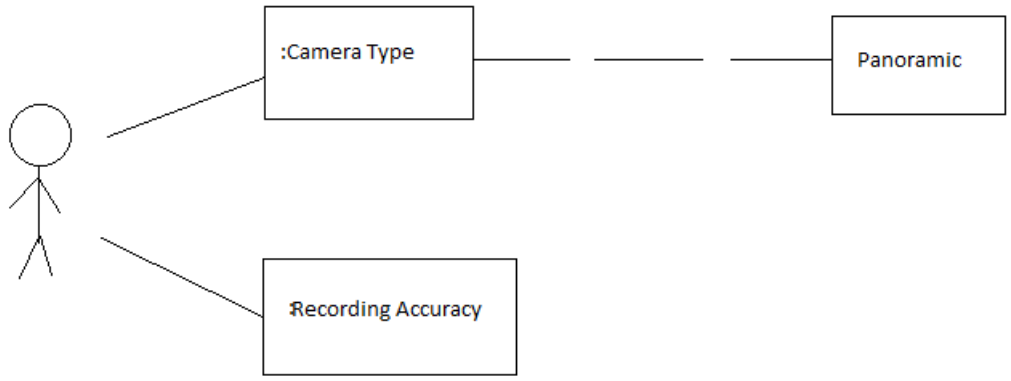


Fig. 3. UML Use Case Diagram of the project's context.

2.4.2 Design Concerns

This Viewpoint is concerned with the specific issues which affect the performance of many other aspects of the project, which are to be adhered to throughout the building and testing of the project.

2.4.3 Design Elements

The first element is that the finished project will be able to return the velocity of specified objects in frame. The returned velocities will be within 90% of their actual velocities. The upcoming elements discussed are general pointers on how to achieve that outcome. The second element is that the project will utilize a Kinect camera to receive the video data. The Kinect will generate information about the depth of objects in the frame. Our client provided us with the Kinect, due to its nature to more synchronize shutter timing and create depth information. The next element is that the cameras must have enough range to accurately determine an object's depth. We believe that the Kinect's 6 meter range will be sufficient for our tracking purposes.

2.5 Algorithm Viewpoint

2.5.1 Viewpoint Description

This viewpoint describes method used for calculating the velocity of the desired object.

2.5.2 Design Concerns

The concerns of this viewpoint are determining the logic for calculating the velocity of the desired object type, determining where data would be coming from and sent to, and when data should be stored for later use and when it should be retrieved.

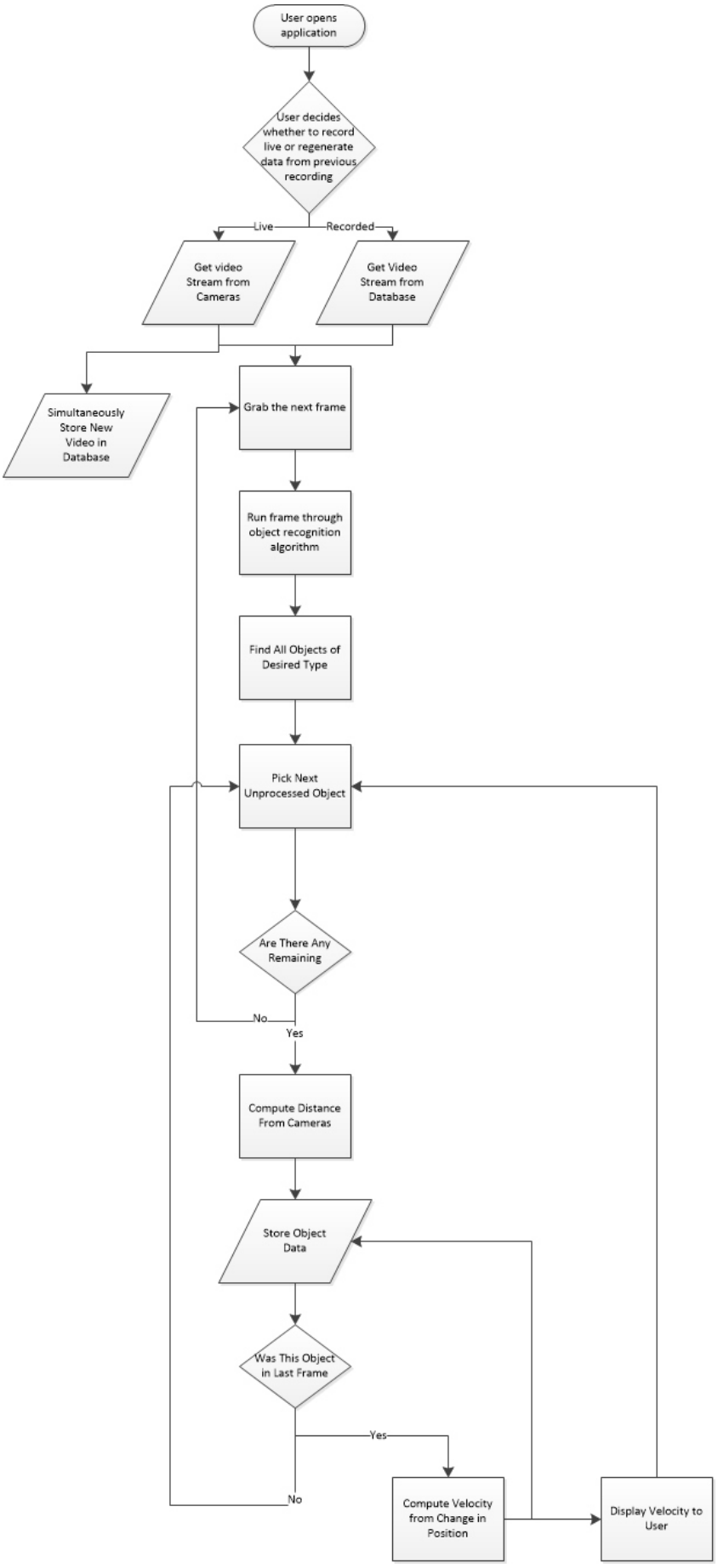


Fig. 4. Flowchart Diagram of Calculating Object Velocity

2.5.3 Design Elements

The data elements represent the database. The element that displays velocity to user will print the information to the application window, with a way of showing the user which object is being referenced by that velocity.

2.5.4 Processing Attribute

The prerequisites for this flowchart is for the application to have been activated, the cameras be plugged in, the database connected to the application. The loop for finding the next object to be processed withing the frame will continue up to the minimum number of objects our product must recognize, but may or may not process the remaining objects depending on the limitation of the system. The priority of the algorithm will be to compute the velocities of the minimum number of vehicles, followed by frame-rate for displaying to velocity to the user.

2.6 Design Rationale

The main rational behind the creation of our product and this document was the concerns supplied by our client Alex Neighbors, making a proof of concept for a product that can replace the current technology for object velocity tracking by making up for the pitfalls of the current technology. So we designed our product around how we could make up for the pitfalls, such as how a camera would have the ability to see and therefore track several object whereas one of the current methods, a radar gun, could not. The choice of our camera depended largely on price, convenience, and the ability to track objects. We decided the Kinect would be a good choice because of the 3D sensors it uses to determine depth. The Kinect also was created to track people which gave it an edge on other camera choices.

3.1 Changes Made

In this document there were some changes from the original design to the final. The recording and replay feature changed a significant amount. Firstly, our final application cannot replay the video. Secondly, instead of saving the recordings as video, the recordings are saved as images. Thirdly, the recordings are not saved in a structured database as originally intended, but are saved to a specified file that they could be view from. Another change from the original is the accuracy of the speed. Originally we planned on 90% accuracy, but this was difficult to verify as the objects we tested rarely had a consistent speed. Therefore the accuracy of the speed is likely lower than 90%. Another difference in this document is that we changed from multiple objects to a single object.

4 Tech Review

(Begins on next page.)

Oregon State University
College of Engineering

CS Capstone Technology Review

February 18, 2017

Object Velocity Tracking

Prepared for

Alex Neighbors _____ _____
Signature *Date*

Prepared by

Group 37

Alex Bailey _____ _____
Signature *Date*

Dylan Washburne _____ _____
Signature *Date*

Benjamin Wick _____ _____
Signature *Date*

Abstract

While we could consider whether we would use a singular camera or multiple, there is really no debate here. Stereoscopic cameras carry such an advantage for a project like this, we are making that decision.

CONTENTS

1 Introduction

This document is a detailed review of multiple technology options considered for our project. Each piece of technology has three options that is being compared. It includes reviews of live data feed, synchronization, and long-term compression written by Alex Bailey. Also reviewed, are options for long-term storage, computer vision libraries, and object tracking algorithms written by Ben Wick. Finally, written by Dylan Washburne, includes live compression, UI overlay, and video to velocity formulas.

2 Technologies

2.1 Live Data Feed

The options for Live Data Feed are processing both images simultaneously, both images plus distance information, and a single image with distance information. The goal for this piece is how the information should be passed into the velocity algorithm. The velocity algorithm will need to know how the object is moving perpendicular to the camera as well as how it is moving parallel to the camera. Because of this, the algorithm will need to know the distance the object is from the camera and whether or not it is changing as the object moves. There are a couple criteria for this piece. The first is simplicity and ease of use, as we have a limited timeframe for this product. Second is the amount of data that is being transmitted. The more data being transmitted, the longer the process will take.

For processing both images simultaneously, the program will pass the images from both cameras to the velocity algorithm and the velocity algorithm will have to compute the distance to the object as before computing the velocity of the object. Computing the distance will add some complexity to the this method, but not a lot, as the distance will have to be calculated at some point, but this may not be the most efficient place for this computation. The amount of data being transferred is significant because 2 video feeds is a lot of information, but the lack of the extra distance information means that it is not the worst it could be.

For processing both images plus distance information, the program will pass the images from both cameras to the velocity algorithm as well as the distance to the object. Having the distance computed before calculating the velocity will reduce the complexity of this piece by a fair amount, though the distance will still have to be calculated at some point. The amount of data being transferred is very significant in this option as two video streams is a lot of data and then the distance information of top is a lot.

For processing a single image with distance information, the program will only pass the video feed from one of the cameras, probably an arbitrary choice of always the left or the right camera, to the velocity algorithm. With the distance already being computed this will have a small complexity as the algorithm will only need to compute the velocity. The amount of data being transmitted will be significantly less than the other two options, having only one video feed with extra data.

Option	Difficulty	amount of data
2 Image	Hard	Medium
2 Image and Data	Easy	High
1 Image and Data	Easy	Low

Because the single image with distance information is tied for the lowest difficult but also has a significantly less data transmission it is the best choice

2.2 Synchronization

The options that we will be looking at for this technology are having two cameras and having no synchronization, having two cameras and using a global shutter to synchronize the camera, and using a stereoscopic camera which will self-synchronize. The goal of this piece is to decide the best method for synchronizing, or not synchronizing, the two cameras we will be using the track the objects. This is important because we will be performing calculations based on the images from both cameras. Since the objects we will be tracking are moving, we will need to know if the two images were taken at the same time. If the cameras are synchronized, then we can perform calculations knowing the object is in the same place. If the cameras are unsynchronized then the objects will be in different locations and we will have to take that into account. The main criteria that our group will be looking at is ease of use. Due to our projects limited timeline, we need a method that will be simple and fast to pick up.

The first method is have to two cameras with no synchronization. If we were to use this method then we would have to take into account the difference in position between cameras. Depending on the velocity of the type of object this could be significant. If we choose people as our object, this difference will be fairly little. However, if we to use vehicles as our object, then the difference will be more significant. This will require a fairly complicated algorithm to account for the difference in either case. The framerate of the camera will also play a role in this. The lower the framerate the greater the amount that the two cameras can be off.

The second method is to have two cameras that have a global shutter, most likely set by the computer controlling the cameras. This could be fairly difficult. Depending on the type of camera, it may not be possible to set the shutter from the computer running the cameras. This would mean that we would need to attempt to start them at the same time and hope that there is no variation in the real shutter speed, otherwise there could still be variation. Alternatively, we would have to find a hardware work around, but as our group has limited knowledge of video camera hardware, this would be very difficult, especially with our limited timeframe.

The third method is to use a stereoscopic camera. This would be a simpler solution as the camera itself will deal with the synchronization of the shutters. However, this would come at the price of not being able to choose how far apart the camera lenses are. Though there could be other benefits, depending on the camera.

Type	**Difficulty**
No Sync	Hard
Global Shutter	Medium
Stereoscopic camera	easy

Due to us needing the easiest solution to implement, the stereoscopic camera is the best choice.

2.3 Long-Term Compression

The options that we will be looking at are Xvid, FFV1, and OpenH264. The goals for this piece of the project is to compress the video after it has been displayed to the user, to be kept for long term, should they be needed at a later date. With our product, it is likely that the user will be leaving our product running for a significant amount of time, possibly hours. When this happens, video files can become rather large. This will become a problem for long term storage if our product is used often. So, the answer to this problem is to use a compression encoder/decoder, codec. This will reduce the size of the video as much as possible. There are several factors to consider when looking at video compression codecs. The first is whether or not it is lossless. When compressing information, especially pictures and videos, the compression codecs will often save space by removing data, often in the form of merging pixels, leading to a lower resolution, these are called lossy codecs. While a lossless codec is obviously preferred, there are not many lossless video compression methods available and they often have limited compression. Lossy codecs generally offer a greater reduction of file size, with the amount of data lost, depending on the codec. This could be an acceptable tradeoff, depending on the size of the file and amount of quality lost. Second is the amount of compression, often expressed as a ratio. This is the ratio of the size of the original video file to the size of the compressed file. This can be difficult to judge as some videos compress better than other depending on what is being recorded. Third, price is almost always a factor, as it is in this case. If there is an open source alternative that is comparable to a paid version, then the open source would be more favorable.

Xvid is a an open source codec alternative to a commercially sold codec, DivX [1]. Xvid claims to be able to ”compress video at a ratio of 200:1 or more” [long2]. While impressive, this is likely only under certain conditions. Xvid is a ”’lossy’ compression but aims at removing just those picture details that are not important for human perception” [2].

FFV1 is a lossless video codec that is a part of FFmpeg a ”leading multimedia framework, able to decode, encode, transcode, mux, demux, steam, filter and play pretty much anything that humans and machines have created” [3]. Two sources have reported FFV1’s compression ratio as roughly 100GB per hour to 45-50GB per hour [4], approximately 2:1, to roughly 1.2:1 to 2.5:1 [5].

H.264 is a video codec that was created by International Telecommunication Union [6], that has become ”an industry standard for video compression” [6]. Cisco has recently decided to release an open source version under a BSD license and cover the royalties for anyone using their binary files [7]. While this would allow our product to use the H.264 codex, it would restrict us in the need to only use their binaries and always keep them up to date. Should there be an issue, this could cause significant legal trouble. H.264 has a lossless version that has a ratio of roughly 2:1 [8].

Codec	**Lossless**	**Compression Ratio**	**Open Source**
Xvid	No	200:1(theoretical)	Yes
FFV1	Yes	2:1 (roughly)	Yes
OpenH264	Yes	2:1 (roughly)	Yes (with caveat)

While Xvid has the best compression ratio, since it is not lossless, it is not the best choice. OpenH264 has the roughly the same abilities as FFV1, but because of the potential legal problems and potential royalty payments this is also not the best choice. This means that FFV1 is the best option for the long term compression codec.

2.4 Long-Term Storage

The three option we have considered for video storage is Audio Video Interleave (AVI), Matroska Multimedia Container (MKV) and MP4. The goal of the storage is to use the best storage container based off of features and for our user needs. The container used is important because we plan on storing the videos so the user is able to go back and view the video feed. This however, creates problems for storage because raw video files can tend to be large in size and storing all of our video would just be impossible. Once the video is compressed with a codec, the container is going to be used to package the video. We are looking for a container that is capable of handing compressed files and storing them.

AVI is an older container created by Microsoft. It features support almost any video format and audio format. Because it is an older container, it does lack a few features. The only supported devices are Microsoft devices. This means it won't be able to play on any apple device. AVI also tends to be larger than most video formats because of the lack of video compression features [28].

MP4 is another container that is developed by the Motion Pictures Expert Group. The videos inside MP4 files are encoded with H.264 [29]. MP4 is compressed using AAC encoding or lossy which allows for great storage [29]. MP4 is compatible with devices like IPad, IPod, Android Phone and many others which gives it an edge on MKV and AVI [29]. This reason alone makes it usually one of the most used containers. MP4 is also capable of handling high quality videos. However MKV tends to win in that category.

MKV is another container that is used that can hold an unlimited number of video, audio, picture or subtitle tracks that is developed by CorCode, Inc [30]. MKV does a great job of handling high quality video [30]. It is very flexible a It can't be played on portable devices including phones and tablets [30]. However, MKV does support H.264/AVC which creates for efficient HD content playback.

	Video formats supported	**Codecs supported**	**Files size**
AVI	Almost anything	DivX, Xvid, Cinepak, Indeo, DV and Motion JPEG	Tend to be larger
MP4	MPEG-2 Part 2, MPEG-4 ASP, H.264/MPEG-4 AVC, H.263, VC-1, Dirac	AVC M	Smaller than both AVI and MKV
MKV	Almost anything	Almost any	Tend to be very large

Choosing a container is a little challenging because many offer different benefits. MKV and AVI tend to have larger file sizes. However, this means it's quality is lot higher. Because we aren't necessarily looking for super high quality video playback, this may not be the best options for us. MP4 is also better supported on devices. This is a huge positive as it creates less chance for issues when playing back video. This means that MP4 is the best option for us.

2.5 Computer Vision Library

The three options for Computer Vision (CV) libraries include OpenCV, LTI-Lib, and VXL. Selecting a good CV library is essential for our project. The goal of the CV library is to provide us with a large array of functions that we are able to utilize. Every CV library we are looking at is open source and is free to use. The library selected must be able to support our needs of being able to identify and track objects in real time through our live video feed. It must have many available tools to simplify the identification and the tracking of objects. The criteria that will be evaluated are languages available, features available, and performance. The languages available is important because choosing a language we are already familiar with will give us an advantage. We are hoping to use C/C++. Feature available is also another very important criteria. Specifically, we are looking for features that are able to simplify object tracking and detection. Performance also plays a huge large factor in our choice. We are looking for library that is as efficient as possible. These libraries tend to be written in C/C++.

OpenCV is one of the most commonly used libraries for computer vision. The OpenCV libraries are written in C/C++. According to their website, they offer over 2500 algorithms [18]. The purpose of OpenCV is to offer a large number of function that the user is able to use to simplify difficult tasks. This includes algorithms that are capable to identify and track objects, recognize faces, follow eye movements and many more [18]. OpenCV is used by many large companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda and Toyota [18].

LTI-Lib is also another Computer vision library that has many algorithms and features. LTI-Lib's main goal is to provide an object oriented library in C++ [19]. It includes libraries for linear algebra, classification and clustering, image processing, as well as visualization and drawing tools [19]. There are a few research projects that utilize LTI-Lib including one at the University of Liege, Belgium where they do many things like machine learning, medical imaging, radar imaging, as well as sports video analysis.

VXL is another great computer vision library utilized by a lot of people. Similar to OpenCV, the VXL libraries are written more in C++ [20]. The main libraries within VXL include numerics (VNL), imaging (VIL), geometry (VGL), and streaming I/O (VSL) [21]. They also have libraries for image processing, camera geometry, stereo, video manipulation, 3D imaging and many more [21]. VXL is written to be light, fast, and portable over many platforms [vision5].

	Languages available	**Features available**	**Performance (rank)**
OpenCV	C, C++, Java, Python	Many	1
LTI-lib	C, C++	Many	2
VXL	C, C++	Numerics, Imaging, geometry, streaming I/O	3

Based on the criteria needed for computer vision library, OpenCV has a large number of algorithms that we will be able to use as well as performs faster than the other libraries. The speed performance test was done by Utkarsh Sinha. The comparisons done were 2D DFT, resizing, optical flow, and neural net [22]. His findings show that OpenCV performs faster than both LTI-Lib and VXL with LTI-Lib coming in second [22]. Also, Based on reviews, OpenCV is just a lot more popular and widely used. This is a huge positive because it means more references and documents will be available for use. For our project I think OpenCV would be a lot more useful based on the criteria tested.

2.6 Object Tracking Methods

The three options for object tracking algorithms are Haar Cascades, background subtraction, and Speeded Up Robust Features (SURF). Each method can be implemented using OpenCV libraries. The goal of each method is to offer an accurate and efficient way of identifying an object. To compare each option we will look at accuracy and speed of the method. Accuracy is a crucial criteria because a method that does not correctly identify an object will fail to calculate the speed. We are looking for the best option that will not produce a large number of false-positives as well as miss objects. The speed of the method is also very important because the method used should be capable of detecting an object that is moving quickly through the video feed.

Haar Cascades is a great object detection method. It is capable of facial recognition, vehicle tracking, animal tracking, and many other objects [24]. Haar cascades is most accurate with facial recognition with a precision rate of 80 percent [24]. However, vehicle tracking precision is sub optimal with a precision rate of just below 50 percent [24]. The accuracy of cars does go up when viewing cars from the side [24].

Background subtraction is also another common method for object detection. According to documentation on OpenCV's website, "Background subtraction calculates the foreground mask performing a subtraction between the current frame and a background model." This means it is able to detect the background and then use that to detect the moving object. This is done in two steps, background initialization, and background update [25]. This method is easy to apply when an image of the background alone is available [26]. Because the method relies on moving objects, objects that are moving more slowly have a higher chance of error. Shadows can also cause error because shadows move as well but can be marked as foreground [26].

Speeded Up Robust Features or SURF is another method capable of facial recognition and car detection. SURF relies on an image to determine key points. These key points are then used to detect the objects on the live feed. This means it will rely on a selection of images to create key points and then compare it to the live video feed to recognize objects, so creating a library with enough images is crucial to increase accuracy.

	Acuracy	**Speed**
Haar cascades	Accurate for facial recognition, less accurate for car detection.	Very fast
Background subtraction	Accurate for fast moving objects	fast
SURF	Accurate (%80)	Slightly slower than both Haar cascades and background subtraction

This option is a little more difficult to choose from. All options are viable and have positives as well as negatives. The accuracy testing was done in a research paper testing for the speed and accuracy of Haar cascades and SURF [27]. The deciding factor is whether accuracy is more important than the speed of the detection. As long as the method is capable of detecting fast moving objects with a low rate of error, then accuracy should be the main deciding factor. Because Haar cascades is more accurate for facial detection, and SURF being less efficient, our best bet is to explore the background subtraction method.

2.7 Live Compression

When the program is running, the camera will be actively generating data. Videos by their basic nature require large amounts of data to be transferred. A method to reduce the load of the video passing through is to compress it at some step of the way. However, compression carries with it potential dips in video quality which may be required to accurately identify objects.

The most basic approach to the problem is not to compress the video at all. This ensures the entire image is received by the computer which is doing the processing. This however is not an ideal solution, because while it is receiving the maximum quality image, it could also has to transmit the entire image to the display port, which often is smaller than the resolution the video was taken at. This also means that while it has the highest visual fidelity, it has to parse that much more imagery before it can spit out a response [15].

A second method to consider is that the computer does video compression in addition to its other duties, every frame. This offers the option to output significantly less imagery to the display port each frame, at no loss to incoming video. Alternatively, the computer can also compress the image before it begins its scans to lessen the amount of work required to scan the entire image. It is fully possible that the image will not degrade in a significant enough manner under compression that it would affect the object recognition [16].

If the image getting compressed would not affect the object recognition, then it would be beneficial to offload as much processing the computer would perform as possible. To this end, it is possible that modern video cameras can compress the frames before sending the images back to the computer. Doing so would offload a decent amount of stress from the computer when processing the image, as well as when receiving the data from the camera in the first place [17].

Name	Processing Strain	Image Quality	Speed
No Compression	High	High	Slow
Computer Compression	Very High	Moderate	Moderate
Camera Compression	Low	Moderate	Fast

No compression, while the ideal of receiving perfect image quality is certainly a respectable endeavor, is unfeasible for this project. In addition, the quality loss from compression does not seem to my human eye to be significant in any measure. To that end, it seems most beneficial to move as much strain off the computer as possible and, therefore, attempt to get a camera which will automatically compress the frames before they're sent for processing.

2.8 UI Overlay

The UI overlay we select can be generated by one of the three following software packages: OBS, Camtasia, or XSplit. The goals for this software is to run with minimal overhead because it must perform its function while the backend is also doing the heavy lifting. A major function we require from the selected software package is the ability to create an overlay on top of the incoming video feed. In addition, depending on how the backend is set up, we will also need the overlay to place boxes around positions specified by the backend. This is an effort to show the user where identified objects are in the image feed.

Open Broadcaster Software, frequently referred to as OBS, is an open source project designed for use in video editing and streaming. OBS is also currently the most used software for live-streaming video feeds over the internet. It is made to be as lightweight as possible while providing all required power to the user for the purposes of editing the video feed [12].

Camtasia is, at its core, a video editing software. Since the rise in popularity of live streaming, Camtasia's creators (TechSmith) have added the capability to support live video streams. They have an extensive toolset to allow you to manipulate what is shown on-screen due to their roots in video editing, but that also comes at the cost of marginally more overhead when running compared to other modern solutions [13].

XSplit is a software designed primarily for use in web streaming a video. As a result, it is designed to be run in the background while more strenuous tasks simultaneously run on the computer. Popular add-on packages allow it to generate pop-in elements on the overlay, primarily for Twitch donation notifications, but it can potentially be retooled for dynamic placement [14].

Name	Overhead	Dynamic Overlay Placement	Open Source
OBS Studio	Low	Yes	Yes
Camtasia	Moderate	Yes	No
XSplit	Low	Yes	No

Based on the capabilities of review technologies, OBS Studio is the selection for this project. Camtasia has too much overhead due to its roots as a video editor. XSplit is much more similar in performance reviews, but OBS has more power available to the user. While XSplit can install packages to match this power, each package has an overhead cost, which places OBS on top.

2.9 Video to Velocity Formulas

In a sense video to velocity formulas already exist, however they exist to view their targets from a known, fixed distance and take advantage of hard-coded values. In that way, we can certainly make use of what has been made, but we will have to heavily modify it for our purposes. We can do so by using one variety of block-matching algorithm. Our options include SDSP, Adaptive Rood, and Phase Correlation. Because we are expecting to receive many frames every second, the computer should be able to calculate the velocities of located objects with minimal overhead and maximum speed.

A Small Diamond Search Pattern (SDSP) is a heavily simplified version of what is referred to as an exhaustive search. In a sense, an exhaustive search is a brute-force application to find the object's motion between frames, in any possible direction. SDSP approximates the boundary conditions of an exhaustive search by searching in 8 directions around an object to where it may have traveled. It repeats this any number of times until it believes the new location is approximately correct. This method has very small variance from an exhaustive search while offering very reliable velocities when set up correctly [15].

The Adaptive Rood Pattern Search is an algorithm that takes an object's previous motion and assumes it will approximately continue to use that motion between frames. This allows it to potentially jump to the answer immediately, though in reality it will self-correct in a large number of cases. This self-correction takes place through the use of a Diamond Search pattern, a variant of which was discussed above, after which it returns a velocity [16].

Phase Correlation is used to determine how far an object has moved on screen by comparing the two images and using the fourier shift transform to locate the change in x and y which the object traveled. This gives highly accurate results when it works, but its reliability comes at the cost of higher processing time required [17].

Name	**Overhead**	**Speed**	**Accuracy**
SDSP	Low	Fast	High
Adaptive Rood	Low	Fast	High
Phase Correlation	Low	Slow	High

Phase Correlation is clearly not an option we will wish to pursue for this project, based on the conditions listed above. The SDSP had the lowest overhead of any method available, however Adaptive Rood was notably faster in many scenarios. It is worth noting Adaptive Rood has parts based around Diamond Search patterns which would make it more advanced than the pattern by itself, however it is questionable whether the objects in frame would move fast enough that we would need to predict their motion. That said, I am selecting the Adaptive Rood method for this project.

3 CONCLUSION

After researching multiple options for each technology, we believe we have a good start starting point of which option we will need to use create our project. The above information shows what we will prioritize when developing the project.

REFERENCES

[1] B. Clark, 'All You Need to Know about Video Codecs, Containers, and Compression', 2015. [Online]. Available: http://www.makeuseof.com/tag/all-you-need-to-know-about-video-codecs-containers-and-compression/ . [Accessed: 15-Nov-2016]

[2] xvid.com, 'Home', 2016. [Online]. Available: https://www.xvid.com/ . [Accessed: 15-Nov-2016]

[3] ffmpeg.org, 'About FFmpeg'. [Online]. Available: http://ffmpeg.org/about.html . [Accessed: 15-Nov-2016]

[4] E. Lorrain, 'A Short Guide to Choosing a Digital Format for Video Archiving Masters', 2014. [Online]. Available: https://www.scart.be/?q=en/content/short-guide-choosing-digital-format-video-archiving-masters . [Accessed: 15-Nov-2016]

[5] CS MSU Graphics & Media Lab, *Lossless Video Codecs Comparison*, Moscow: CS MSU Graphics & Media Lab, 2004, p.14[Online]. Available: http://compression.ru/video/codec_comparison/pdf/lossless_codecs_test_en.pdf . [Accessed: 15-Nov-2016]

[6] Vcodex, 'H.264 Advanced Video Coding'. [Online]. Available: https://www.vcodex.com/h264-resources/ . [Accessed: 15-Nov-2016]

[7] openh264.org, 'FAQ'. [Online]. Available: http://www.openh264.org/faq.html . [Accessed: 15-Nov-2016]

[8] H. Lewetz, et al., 'Comparing Video Codecs and Containers for Archives', 2013. [Online]. Available: http://download.das-werkstatt.com/pb/mthk/info/video/comparison_video_codecs_containers.html#codec_tests . [Accessed: 15-Nov-2016]

[9] compression.ru. [Online]. Available: http://compression.ru/video/codec_comparison/pdf/msu_lossless_codecs_comparison_2007_eng.pdf . [Accessed: 15-Nov-2016]

[10] microsoft.com. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/dd743961(v=vs.85).aspx . [Accessed: 15-Nov-2016]

[11] vcodex.com. [Online]. Available: https://www.vcodex.com/news/how-to-stream-better-quality-video-part-1-basics-and-good-practices/ . [Accessed: 15-Nov-2016]

[12] obsproject.com. [Online]. Available: https://obsproject.com/ . [Accessed: 15-Nov-2016]

[13] techsmith.com. [Online]. Available: https://www.techsmith.com/camtasia.html . [Accessed: 15-Nov-2016]

[14] xsplit.com. [Online]. Available: https://www.xsplit.com/ . [Accessed: 15-Nov-2016]

[15] ncbi.nlm.nih.gov. [Online]. Available: https://www.ncbi.nlm.nih.gov/labs/articles/18255398/ . [Accessed: 15-Nov-2016]

[16] semanticscholar.org. [Online]. Available: https://pdfs.semanticscholar.org/7f78/41b7b9c98b1e1f52282bdc3c2710cf83d3f0.pdf. [Accessed: 15-Nov-2016]

[17] semanticscholar.org. [Online]. Available: https://pdfs.semanticscholar.org/2c6c/6d3c94322e9ff75ff2143f7028bfab2b3c5f.pdf. [Accessed: 15-Nov-2016]

[18] "ABOUT — OpenCV", Opencv.org, 2016. [Online]. Available: http://opencv.org/about.html. [Accessed: 16- Nov- 2016].

[19] "LTI-Lib", Ltilib.sourceforge.net, 2016. [Online]. Available: http://ltilib.sourceforge.net/doc/homepage/index.shtml. [Accessed: 16- Nov- 2016].

[20] "VXL: 1. Introduction", Public.kitware.com, 2016. [Online]. Available: http://public.kitware.com/vxl/doc/release/books/core/book_1.html. [Accessed: 16- Nov- 2016].

[21] "VXL - C++ Libraries for Computer Vision", Vxl.sourceforge.net, 2016. [Online]. Available: http://vxl.sourceforge.net/. [Accessed: 16- Nov- 2016].

[22] "Computer Recognition System For Detecting And Tracking Objects In 3D Environment", Csus-dspace.calstate.edu, 2016. [Online]. Available: http://csus-dspace.calstate.edu/bitstream/handle/10211.9/1249/Introduction%20to%20computer%20vision.pdf?sequence=2. [Accessed: 16- Nov- 2016].

[23] U. Sinha, "OpenCV vs VXL vs LTI: Performance Test - AI Shack - Tutorials for OpenCV, computer vision, deep learning, image processing, neural networks and artificial intelligence.", Aishack.in, 2016. [Online]. Available: http://aishack.in/tutorials/opencv-vs-vxl-vs-lti-performance-test/. [Accessed: 16- Nov- 2016].

[24] "Object Detection Using Haar-like Features" 2016. [Online]. Available: http://www.cs.utexas.edu/ grauman/courses/spring2008/slides/Faces_demo.pdf. [Accessed: 16- Nov- 2016].

[25] "OpenCV: How to Use Background Subtraction Methods", Docs.opencv.org, 2016. [Online]. Available: http://docs.opencv.org/trunk/d1/dc5/tutorial_background_subtraction.html. [Accessed: 16- Nov- 2016].

[26] "OpenCV: Background Subtraction", Docs.opencv.org, 2016. [Online]. Available: http://docs.opencv.org/3.1.0/db/d5c/tutorial_py_bg_subtraction.html. [Accessed: 16- Nov- 2016].

[27] "Learning SURF Cascade for Fast and Accurate Object Detection"2016. [Online]. Available: http://www.cv-foundation.org/openaccess/content_cvpr_2013/papers/Li_Learning_SURF_Cascade_2013_CVPR_paper.pdf. [Accessed: 16- Nov- 2016].

[28] ”AVI vs MP4 , difference between AVI and MP4 format”, Iorgsoft.com, 2016. [Online]. Available: http://www.iorgsoft.com/compare/avi-vs-mp4-comparison.html. [Accessed: 16- Nov- 2016].

[29] ”All You Need to Know about Video Codecs, Containers, and Compression”, MakeUseOf, 2016. [Online]. Available: http://www.makeuseof.com/tag/all-you-need-to-know-about-video-codecs-containers-and-compression/. [Accessed: 16- Nov- 2016].

[30] L. Case, ”All About Video Codecs and Containers”, TechHive, 2016. [Online]. Available: http://www.techhive.com/article/213612/all_about_video_codecs_and_containers.html?page=2. [Accessed: 16- Nov- 2016].

4.1 Changes Made

For object tracking we listed several different methods. We ended up choosing to use a simpler approach for tracking because the object was changed to a red ball. Our other options included haar cascades, background subtraction, and SURF. Looking further into using haar cascades, a lot of sources online discouraged the use of it for simple objects like a red ball. The method would have involved us to track certain features of the object but because the red ball is simple enough it would have been essentially a color tracking method. This is also the same reasoning against using the SURF detection method. The last option, background subtraction, was also considered. However, we wanted our project to work without a pre-determined background which is why we decided not to use this.

Another Change made was not using any of the long term storage methods talked about which includes AVI, MP4 and MKV video storage. In the initial design we intended it on being saved as a video. In our final completion, however, we used images and decided to save the images instead.

We also altered the velocity formula we were to include from the tech review document. In the tech review, we looked at several sophisticated methods for extracting velocity from an object on-screen, however in the end it turned out that we didn't need a method that was sophisticated so much as a method that worked. Given our resources at hand, it made more sense to simply take the translation of the object's position as a vector and use that vector as the velocity.

5 Weekly Blog Posts

5.1 Alexander Bailey

5.1.1 Fall Term

Week 3

Our progress so far is that we have talk to our client and discussed what he wanted for the project and made a problem statement that he and our group agree on.

For next week, we will begin looking into the different types of cameras that are available and their APIs to determine which type of camera and API will best suit our needs. We will also research the various programs that currently exist that are related to our problem and how much our group will need to do from scratch. Currently there have been no problems.

Week 4

This last week I gave an update to our client about the need to redo the problem statement and the impending requirements document. I emailed our TA asking for comments on our problem statement.

For this coming week, we are going to get our problem statement revised. We are meeting on Monday to work on the requirements document.

We haven't had any problems

Week 6

This week, we polished up our requirements document with feedback from our TA Jon Dodge and Professor Kirsten Winters. We are sending our client the requirements document for his input and signature. Next week we will be writing out tech reviews. There were no problems.

Week 7

This week we got approval from Alex Neighbors on our Requirements Document and submitted it. We also discussed

our Tech Review and did some research.

Next week, we will finish up the Tech review and begin working on our Design Document.

One problem with this week is that we had not figured out what 9 things we were going to do for our Tech Review.

Week 8

Last week we wrote our tech review and got started on the design document

This week we will be continue to work our design document.

There were no problems this week.

Week 9

This week we began work on the design document.

Next week we are going to finish up with the design document and begin writing our progress report.

There were no problems this week.

Week 10

This last week we wrote the Design document.

Next week we are going to write the progress report and record it.

There were no problems.

Week 11

Due to Finals we did not do much work, but we did get our paper approved by Alex Neighbors.

In the next few weeks we intend to choose a camera.

We had no problems this week.

5.1.2 Winter Term

Week 2

This week we are having Alex Neighbors send us the Zed camera. I also began learning C#.

Next week we will continue to learn C# and making a GUI.

No problems to report.

Week 3

This week we continued learning C#.

Next week the camera should have arrived and we can begin testing it out.

No problems this week

Week 4

This Week we received the Xbox 360 kinect camera and are waiting on the usb adapter to begin testing with it. We also are making the skeleton for the GUI.

Next week we should have the adapter and can begin working with the kinect. Next week we plan to be able to view a camera within our GUI.

No problems.

Week 5

This week we received the USB adapter for the Kinect camera, so we can begin using it. I looked further into using image subtraction for tracking.

Next week I'll look more into using the Kinect with C#.

No problems.

Week 6

This week we worked on updating the documents for the alpha level release.

Next week I'll continue looking into Kinect.

no problems.

Week 7

This week I worked on getting the depth map to display on the screen.

Next week, I'll continue working on the depth map.

No problems this week.

Week 8

This week I got the depth map to display as a black and white screen, however the screen oscillates for some reason. More specifically as an object moves towards and away from the camera the object should move from black to white with white being closest and black being far way, but instead it the object will go black then white then black then white, etc.

Next week I will try and find out why the depth map is doing this.

No problems this week.

Week 9

This week I worked on fixing the depth map so that it doesn't cycle between black and white.

Next week I will continue to work with the depth map.

No problems this week.

Week 10

This week we mainly focused on the poster and progress report paperwork, so little was done on the actual project. We got confirmation to use colored balls for tracking and to scale back the recording and replaying feature.

Next week we will continue working on the paperwork.

No problems.

5.1.3 Spring Term

Week 2

This week we were able to get the object tracking working with the Kinect camera and began working on getting the depth data at a specified point.

Next week we will finish adding the finding depth of the center of the object.

One problem that we aren't entirely sure about is that if too much red enters the frame then the camera will not update, but as soon as the red leaves the frame, the camera is fine. Luckily, it works fine when only one ball of red is in the frame, so this will work for our needs, but we are also looking into Haar cascades, which would hopefully remove this problem.

Week 3

This week we mainly worked on a surprising error where only Ben's computer was able to run the Kinect. We also had the poster review session with professor Kirsten and were able to get some good feedback.

Next week we will probably abandon trying to fix the error due to the limited amount of time till the code freeze. Instead we will focus on the speed algorithm.

Currently we have the problem that we fear that we may not have enough time to complete all the things in our

requirements document, such as the recording an play back, so we will talk to our client in order to identify the most important aspects of the project.

Week 4

This week we were able to get speed tracking working and displaying in a text box. We also made a final draft of the poster and had it approved by our client.

Next week we're going to get recording and the text output file working, should be in time for the Code freeze.

The only problem we had is that for some reason my computer has trouble running the recording software, whereas my teammates computers don't seem to have a problem. Mine will show the text of the balls position but won't show the video. However, if I remove enough code from our speed tracking function, it will show up.

Week 5

This week we tested the speed function. We also had the program output a text file of the data. We also started the recording process, but for some reason this caused problems with the live feed.

Next week we are going to work on the midterm report and recording feature.

No problems.

Week 6

This week we began the midterm report paper and video.

Next week we are going to finish our report and then we are going to make sure that our project is ready for the expo display and record a video in case we have trouble doing a live demo during expo.

No problems.

Week 7

This week we finished the paperwork for the progress report, made a demo for expo, and went to expo.

Next week we are going to start the 3 writing assignments.

No problems.

Week 8

This week we contacted our client about the final hand off and returning the kinect.

Next week we'll begin work on the final documents.

No Problems.

Extra Questions

If I were to redo this project from the fall term, I would tell myself to not be so ambitious in the design of the document.

Getting experience with C# is definitely big. Secondly would be working with a openCV.

Mainly the C# for GUIs, but possible openCV if I get into some kind of computer vision.

I liked the interactive nature of the project, it was fun to see the program recognizing objects as they were in motion.

My teammates taught me about openCV and finding the speed of the object in 3D space, as that was the pieces they were in charge of.

I might possibly be disappointment given the original design, but knowing more about object recognition and tracking I think I would be okay knowing that the students had no prior experience with it.

The next step would probably be multiple object tracking.

5.2 Dylan Washburne

5.2.1 Fall Term

Week 3

Since the start of the term, we have now been assigned into this project group, spoken with our client, and determined what we are going to be doing in the project. Personally, I attribute almost all of this to preparing the Project Statement, though honestly it had to be done eventually.

In the upcoming week, we will be researching the available varieties of cameras that would be applicable to this project. This involves researching which cameras can be most effectively used to capture real-time data, as well as determining if we want mono- or bi-focal cameras.

We are also looking into the apis used in the programming of these cameras. While not determined as of yet, we have discussed the point and believe looking into what cameras already have the capability to recognize objects in the frame is a serious consideration to work with.

Week 4

In the last week, I've mostly been looking up some camera options in my free time. The issue with doing this without dedicating serious time to it is that it's a little difficult to locate all the information pertaining to the coding side, and whether or not it has built-in capability to locate objects in frame. Granted, certain cameras like Kinect heavily emphasize that point, but I'm going to need to look a lot more at the api side of things to see if it will behave as needed for this endeavor.

In the coming week, I'm planning on getting the problem statement fully revised, as well as typing out the requirements document. I also hope that the group can sit down and definitively decide what camera we're going to use, because that's going to be needed soon when we are able to begin the actual work.

Week 5

This last week was kind of hectic, but due to a lot of factors not directly influenced by this class. Apart from us redoing the problem statement and making a requirements document, I also had 2 midterms on the same day, so you know... everything suffered just a bit.

On a happy note, now that we've got all this out of the way we can finally start actually doing hard work on the project. And yes, this means actually selecting a camera to use, just as I said 3 weeks ago. That has been our client's number one request to us and we haven't been able to do it very well due to mass overlap in time where we can be together, and documents that need to get done.

Week 6

This week was terrible for me because of non-capstone reasons. As far as capstone itself goes, this week was fairly uneventful, as we mostly did edits to the requirements document. I did personally look a little into cameras and object tracking, but I still have no concrete idea what would be best overall for our purposes.

We really need to get the type of camera locked down. Whatever work we get next week is hopefully trivial so we can, as a group, come together and finally research that.

Week 8

This week was a nice slap in the face to me for our writing. A lot was done between Sunday and Wednesday to make the tech review actually happen. For me, though, I most hustled in redoing our citations, and even that I know wasn't properly done, since /cite didn't want to work for me.

Upcoming, more documents to be done. Yay, more writing! Sorry that was sarcastic, but it's how I deal with being mad at myself over the last week. Writing isn't particularly hard, and I see the applications to all the assignments, so I don't really mind the writing all that much.

Week 9

This last week was Thanksgiving week, so not much time was devoted to this work. Nonetheless, we now have divided up the work for our upcoming document.

Going forward, we will complete that document for real.

Seriously, not much to say this week.

Week 10

The previous week was mostly us completing the requirements document. This took less time than anticipated, but only because it required us to understand everything before we could actually start, which means the combined time was much longer. We also did a little work towards the progress report, but there's still a long way to go on that.

Coming up, we must finish the progress report. After that, we can actually begin on the project itself over winter break. I anticipate we will group a little, but much of what we have to do requires all of us here. Our first steps to get setup complete I anticipate will take a few weeks.

5.2.2 Winter Term

Week 1

Winter break didn't really do anything for our project's completion, since we didn't meet up. We all seem to remember everything with good detail though, so there was no loss either.

We have researched stereoscopic cameras, and found that the market is somewhat cornered to models which have a maximum range of 20m. In fact, most only reach about 3m away, because the major market is peoples' living rooms. Our initial goal of 100m is not going to happen unless we construct it ourselves. That said, we have reevaluated how much we need to track every object in a football stadium, and while a bit more range would be welcome, 20m would satisfy our needs well. We are hoping we can update our requirements to reflect that.

Week 4

We have received the camera to work with. It was not the Zed as we had proposed, but rather a Kinect. I am cautiously optimistic about this change in scope, because narrowing it down is actually a good thing, so we are constrained by force to not over-deliver what we can't possible create.

I kind of wish we had heard the context for this selection from Alex before he sent it to us, but he's explained it now and we all understand. We just sort of wish we had heard this context before we got the package and were rather confused.

We need to do a loot in the coming weeks to get our alpha product done, but I'm hopeful we can get a compact starting product prepared. Unpolished maybe, but I think it can be done.

5.2.3 Spring Term

Week 2

In the previous week, we had a good number of new developments. We were able to make the object tracking perform correctly for objects with the color red, and we have begun implementing that alongside the depthmap. The implementation to make it coincide with the depthmap is not yet complete, but quite frankly at this point the progress we have made is good enough for me.

The main difficulty here is that we are still moving slower than we should, for our point in the term. We have accepted that and been working hard to try and approach where we need to be.

Coming up, we are preparing to update our poster for expo (and the assignment where we need to update it), as well as once again sending a message to our client on how we are modifying the project requirements.

Week 3

This week we received some small red stress balls to work with as objects for the video to focus on. They appear to work well enough, although there are still some issues inherent in the program. We are hoping to resolve them well enough for our expo presentation to be workable.

We also did poster reviews for extra credit in Kirsten's office. To be frank, we knew our poster was a bit terrible going in. That said, we got extremely good feedback from the other group on how specifically to improve the poster. I feel we can make it look good by expo, so I am happy with the meeting.

I am still worried that we may not be able to fix everything by expo. I don't exactly have a plan for us being unable to do this, but it is my largest current concern for this project.

Week 4

This week, we completed a method to take the xy coordinates in the color image, the image which does the object recognition and tracking, and use those coordinates to look up the z distance away from the camera through the corresponding coordinates in the depthmap. This allows us to automatically locate the center of an object, and determine its precise point in space every frame.

From there, I was able to integrate some formulas to calculate velocity, half through knowing the Pythagorean theorem, and half through fiddling with data of a tracked ball at various depths to understand how many pixels it translates horizontally, based on its distance from the camera. Naturally, an object farther away travels fewer pixels horizontally than an object up close, when translating the same distance in the real world.

Coming up, we still need to hammer out a few issues before the code freeze. We are also actively working on touching up the poster before it gets turned in. We have already gained client approval, but we are looking for more minor isues at this point like visual spacing of text boxes (and only to a minor degree, nothing that would change the poster in any significant manner).

Week 5

Code freeze hit on Monday. We were working nearly until the deadline to submit what we felt could be a "complete" product. Granted, we completed most everything before this point, but the last-minute things were more of the small-scale issues which we felt should be featured in a "release".

We also submitted the poster and completed all the individual paperwork required to present at the expo. It was really just busywork, so it was completed in good time.

Since the code freeze hit and we have essentially "submitted" our project, we have actually let ourselves not meet up as often this week. I believe this to be a temporary measure, partially because there are also midterms, but we will be working again to complete the grading assessment paperwork before that is due. We plan to meet over the weekend to bang most of that out.

Week 6

No major breakthroughs have happened this week. To be brutally concise, we have not worked on the actual project in any significant capacity. This is mostly because of other events in each of our lives, notably midterms. It is also notably

because we have been working on the "midterm" progress report.

The midterm still has to be completed. We are catching back up on the paperwork to reflect our actual state of affairs, relative to the previous progress report last quarter. Unlike the last 6 months, we actually have a product to show off for this. This will likely lead to more time spent making the video, to represent what has been completed.

Week 7

Expo happens this week. I think we are as prepared as we will ever be, and I'm honestly just waiting for it to be over. I am sure within 90% tolerance that it will go smoothly, with probably a couple of people causing minor trouble, but nothing show stopping.

Week 8

Expo was last week. Overall I would say it went about as well as expected, though I do wish in retrospect I had packed some different shoes, more aligned to standing in place. Since then, we have not met up in any major capacity regarding capstone, even though we still meet for other class projects. I don't know about Alex and Ben, but I think we are all fairly satisfied with how the expo went.

Looking back, there are a number of things I would have done differently this term. For whoever is reading this, this is where I am beginning the retrospective questions.

If I were to redo this from the fall term, I wouldn't change much from the fall term until we hit the requirements document. There, I would say that we should have promised way less. Of course, we slashed it down to a moderate size as time passed, which I would also say is the biggest thing I learned. I never before realized how often people in the industry would have to go back and tell the client they were cutting features.

In the future, I can see myself using a number of the delegation skills I honed over this term. My programming skills weren't really the largest factor over the project, though I did learn a few new things. Rather, the ability to work with the team and figure out the work loads was the most important skill I had to use nearly every time we met. I suppose you can also say this is what I learned from my teammates, because I can confidently say that all of my notable gains in skill from this project came from interacting with the team.

I liked that the project was truly us building an application from the ground up, with no preexisting code base, though that's a double edged sword because I know any job I acquire will want me to implement on an existing system instead of building my own, and I would have preferred to work on something existing so I could get more real experience.

If I were the client for the project, I am not sure if I would be satisfied with the work done. On the one hand, we delivered a working proof-of-concept at the quality you might expect from recruiting random college undergrads who have never had a job, but simultaneously we could have created something of higher quality that runs better with better market applications. Though, I also think the project itself wasn't exactly that aimed at top-end market applications anyway.

Were this project to be continued next year, I think they could get good progress by optimizing the object recognition algorithm because that's where all the processing is going, and from there expanding the objects it can recognize. They could even scale up to a higher quality camera with higher range to expand the applications the product would be viable in.

5.3 Benjamin Wick

5.3.1 Fall Term

Week 3

This week we learned a lot more about our project. We did some research on cameras that we might use for our project. We also worked on our abstract and problem statement. Next week, we plan on continuing our research and hope to come up with a list of possible cameras we can use.

Week 4

This week we got our camera and we began working on the user interface. We have to update our requirements and tech review. We plan on having a skeleton on the user interface done and able to receive video input by next week.

Week 5

This week we completed our final draft of our problems statement. We also worked on our rough draft of our requirements document. This was my first time writing a requirements document so I think there's going to be a lot of updating as we get feedback on it. After we turn in our final draft of our requirements document, I think it would be a good time to get into researching possible cameras we want to use for our project. Overall, proud of my team and the work we've done so far and excited to keep moving forward.

Week 6

This week we went and got feedback from Kirsten and our TA, Jon to improve our requirements document. Overall we got some really good feedback to improve the document. We all sat down and worked more on the document. We were unable to finish the final signed draft by Friday. However, we did finish our final copy and we just need to send it over to Alex to confirm and sign it. Once we hear back from him we'll be able to sign and turn in our hard copy. So far we haven't really ran into any major issues. Our biggest issue is probably finding meeting times where were all available to meet because of our busy schedules. Other than that our team is making good project and I'm proud of the work we've done so far.

Week 7 and 8

We did a lot more research this week about cameras and what we'll need to do our project as well as our tech review. We made a lot of progress with researching things. After this week I have a lot better of an idea of what we will need and how we will finish our project. We started working on our tech review and though we did start later than I would hope because we struggled finding out 9 different topics we managed to finish on time.

Week 9

This week we mainly just worked on the design document. The document is due in two weeks so hopefully we're able to get a good start and not have to work on everything last minute. Our goal because the document needs to be signed, is to have it done I think by Wednesday next week.

Week 10

This week not much has happened besides wrapping up our project design. We focused on the design this week and we plan on doing our project report starting Monday. We still need to talk about what we plan on accomplishing over winter break, which a head start would be very beneficial.

5.3.2 Winter Term

Week 1

This week our main accomplishment was choosing a camera for our project. We decided to go with the Zed camera

because of multiple benefits. It had the farther range as far as stereoscopic cameras go and the fact that it is stereoscopic will make depth finding a lot easier.

Week 2

We have emailed our client and are currently waiting to hear back from him on the approval of the camera we chose. No problems have occurred otherwise.

Week 3

We still are waiting on our camera, we started to learn more C# and looked further into the design. Other than that, we won't have much to do until the camera arrives.

Week 4

This week we got the Xbox 360 Kinect. At this point all we can do is more research and work on the GUI as we wait for the USB adapter to come in. Next week we should get the adapter and can start implementation.

Week 5

This week we got the USB adapter for the Kinect. We plan on working on the input using the Kinect camera. We need to look at how we can use OpenCV with the Kinect as well.

Week 6

This week we mainly worked on the assignments and worked on our alpha release product. We still haven't been able to track objects yet. My main concern is using EmguCV instead of OpenCV because we plan on using C# as opposed to C++. Finding a good resource to help implement tracking with EmguCV are a little more difficult than OpenCV.

Week 7

Started to work on EmguCV and playing around with the functions in C#. Making slower progress due to loads in other classes.

Week 8

This week I started trying to track a red ball. We all figured a solid color might be easier to track than detecting people themselves. Still unsure if this will be the method of tracking we implement in our final release though.

Week 9

Ran into an issue where the camera to my program wasn't able to connect. I think if I can get it to connect the code for tracking a red object should work.

Week 10

This week I continued to work on the issue with the camera. Not much progress there and may pivot in tracking method because of discussion with team. Going to try and work on tracking using background subtraction. This will detect any moving object thought. Hopefully can finish before final progress report.

5.3.3 Spring Term

Week 1

This week we're able to start tracking red objects and recording their X and Y coordinates using a webcam. We don't really know the units of these coordinates. We will need to do further research on this so we can figure out how to do the speed calculations. We are now trying to implement a speed algorithm that will help us calculate the velocity of the red ball we are tracking.

Week 2

So far this week we combined our depth map and our tracking into one main program. We ran into some issues

where we needed to convert the bitmap image to a EmguCV MAT object so we can apply the tracking onto the image being captured from the Kinect. We figured this out and now we have Dylan doing some more research on the speed calculations. We split up roles to revamp the tracking, maybe using haar cascades, and we also have Alex and Dylan working on the speed algorithm.

Week 3

This week I worked on trying to use haar cascades but after more research I read that for simple objects, like a ball that is it not the best method. So for the lagging issue I'm going to try and draw rectangles instead of circles because the problem was with drawing circles. We also ran into problems where the development environment isn't working for Alex and Dylan and only my machine. As far as requirements go we are thinking we might not be able to finish the file saving and are shooting to just get the speed calculation working.

Week 4

This week we made a lot of progress. We finished up our poster as well as almost completed our speed algorithm. We are really close to wrapping up before the code freeze. We still have a few more features to work on including text file output. We decided that video recorded playback isn't really feasible right now because we are comparing images instead of recording a video. One problem we still are encountering is creating the development environment on systems other than mine. This is something we plan on fixing and we will also create a step by step compiling instruction so that ideally any system can compile our program. We also will work vigorously to finish outputting to a text file.

Week 5

This week we made some good progress on the project entering the code freeze. We finished up saving the information to the text file. The only problem I would say is the tech review. I think there was miss understanding about having elements from the tech review as actual requirements for our project. I didn't anticipate that things in the tech review were going to be graded in our final project, only thought it was the requirements document, so that was a little strange. Other than that We plan on getting started with our midterm project report.

Week 6

This week we mainly focused on documentation and our midterm progress report. We started working on our written document and mostly finished it with the exception of adding in a few images. We haven't ran into any problems this week as we haven't done much to the program itself. We plan on doing the video in the next few days.

Week 7

This week all we did was wrap our midterm presentation as well as prep for the engineering expo. We had to create a video of our demo. We also haven't ran into any issues yet. Next we simply await instructions on our next assignment from Kevin.

Week 8

This week we didn't do any work regarding the project itself. All we did was start creating a template for our final written document as well as e-mail our client about the expo. No problems so far. We plan on just chipping away at the final report to try and get it done early.

If you were to redo the project from Fall term, what would you tell yourself? I would first tell myself to set realistic expectations in the beginning. At first I feel as if didn't really have an idea of what we would be capable of or not capable of and we ended up overshooting on requirements. We ended up having to scale down.

What's the biggest skill you've learned? I think the biggest skill I learned from this is the ability to go out and find

certain information I need to make a project with no previous knowledge. I also learned how to make a program using .NET and visual studios which I think is a great skill to know.

What skills do you see yourself using in the future? Depending on where I work I can see myself using visual studios, ability to use external libraries, and creating things with .NET. No matter where I work, this project helped me develop my soft skills. It helped teach me more about working in a team, writing documents, and just the process of development in general.

What did you like about the project, and what did you not? I liked how our project was really flexible and more so a proof of concept instead of dealing with the stress of creating a marketable product. I did not like how the project relied so heavily on hardware. Certain computers had different performance rates and we weren't able to solve it.

What did you learn from your teammates? My teammates were awesome and I enjoyed working with them. I learned how to communicate and also how to manage time within a team environment. We all were so busy with other classes as well that we had to utilize time we did have to work on the project together.

If you were the client for this project, would you be satisfied with the work done? Yes I would be.

If your project were to be continued next year, what do you think needs to be working on? I think the tracking algorithm can be worked on to further track different objects and track objects more accurately.

6 Project Poster

(Included on next page.)

Video Radar

Calculating the speed of an object in real time using a video camera.

Considered approaches

To build our system and application, we started off knowing we were going to need a camera, computer vision library, and:

- OpenCV
- ZED Stereo Camera: Two cameras side-by-side that parse depth by differences between the two images.

The Kinect

The Kinect is a camera device that was made by Microsoft for use with video games for their Xbox 360 games console. The Kinect has 2 cameras and a infrared laser projector. The first camera is a color camera for normal color view. The second camera is an infrared camera. The infrared projector projects a pattern of dots in front of it that the infrared camera then picks up. The Kinect then determines the distance depending on the pattern and spread of the dots.

Possible Applications

Our project could have several application in real life, tracking various subject, here are af few:

- Cars
- Athletes
- Sporting objects
- Analyzing crowds
- Animals

The Problem

The main method for people to track the speed of an object as they go by is to use a radar gun or laser gun, but unfortunately these methods have issues, they can only be used to check the speed of one object, the user can not tell exactly which object is being measured if there are multiple close together, and they must have an active user.

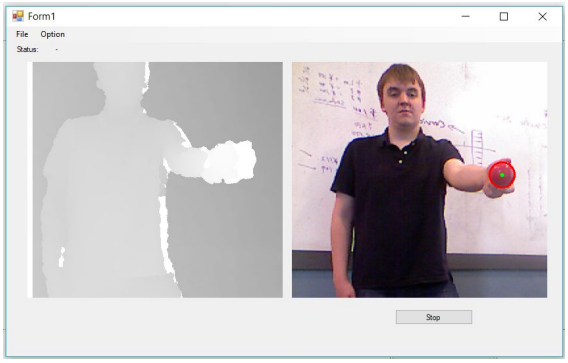
Our Process

First the Kinect camera calculates the depth map based on the infrared projections. Then it passes both the depth map image and the color image to our program. We then run an Emgu CV detection algorithm which identifies the red ball and its center. We then look up the distance to the center of the ball on the depth map. Next, we calculate the real world coordinates of the ball based on its distance from the camera. Finally, we calculate velocity based on change in position vs change in time.

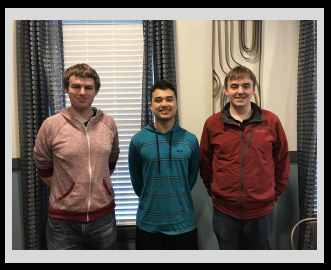
The Solution

Our solution to this problem is to use a camera in order to overcome these issues. For our proof of concept, we are using a Kinect camera. The use of the kinect allows us to measure the distance from the camera to an object so that we can track the velocity of objects. Since the video is being displayed the user can also see which object is going what velocity.

Our solution consists of a Graphical User Interface (GUI) application, which is made very simple for ease of use, that has a box that displays the live camera feed, where the objects that are being tracked are surrounded by a colored box. The velocity of the object is displayed in the text box to the side of the GUI. There is only one button that starts the camera and object tracking and stops it when clicked again. Our solution will also generate a comma separated text file so the user can review the data at a later date.



Ball identification with depth map



Team members:

Alexander Bailey (baileyal@oregonstate.edu)
Benjamin Wick (wickbe@oregonstate.edu)
Dylan Washburne (washburd@oregonstate.edu)

Project Advisors:
Professor Kevin McGrath
Professor Kirsten Winters
Jon Dodge, Teaching Assistant

Sponser:
Alex Neighbors

Conclusion

- **There are many different applications for this kind of system**
- **Efficient tracking methods are difficult to create**
- **To track different types of objects within one application there would have to be a different algorithm for each object which would be difficult**
- **Computer vision has endless possibilities**

Oregon State UNIVERSITY

7 Project Documentation

7.1 How Our Project Works

Our project is a Windows application. Our application is developed in C#, specifically a windows form application. Our application has a GUI developed using C#. In this GUI, we have two image boxes. The image on the left displays the depth map, while the right box displays the camera image with an overlay to identify the tracked object. Next there is the text box, which displays the position and speed of the tracked object. The record button starts and stops the Kinect camera. For the code, when the window is created, the necessary Kinect variables are initialized. Then when the start button is pushed, the Kinect is activated. When the Kinect prepares both the color image and the depth map, the the allFramesReady event is called. In this event the depth map and color image are transfered to a global variable for later use, calling functions to change the type of image. Then the processframe function is called. In this function, we are tracking the red ball. This is done by finding all the red pixels on the screen by using a range of hue values, which indicate red pixels, that exist for each image. Once we have all the red pixels we are creating a thresholded image that just shows only the red pixels on the image. Next we run a few EmguCV calls to filter out outlying pixels to make the thresholded image more accurate and then we are able to draw a circle around the object. After the object is identified, the speed calculation is inside an if statement that is only entered at a rate less than the camera's speed in order to prevent lag. Inside of this if statement, the z disposition is determined by reading the depth map and subtracting from the previous frames z value. The x and y displacements are then calculated by subtracting from the previous frames values and multiplying by the z position in order to compensate for the fact that the further away the object is the smaller it appears. The speed is then calculated as a displacement of the x, y, z divided by the frame-rate of the camera divided by the tick rate, the rate of reducing the speed calculation. Finally the depth map and the color image with overlay are copied to the screen and the speed is written to the text box.

7.2 Setting up our project

The following is a list of software and the location to download it. All of the following is required to run the project. Included is also the steps to make sure the program compiles correctly.

Required Software:

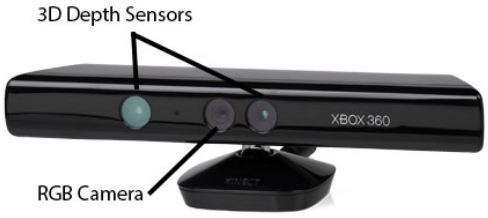
1. Kinect for Windows SDK v1.8 - https://www.microsoft.com/en-us/download/details.aspx?id=40278
2. EmguCV library 3.1.0.2504 - https://sourceforge.net/projects/emgucv/
3. Used for development: Visual studio 2015 community version (.dll is specified for this version)

Steps to compile:

1. Install Kinect SDK v1.8
2. Install EmguCV library v 3.1.0.2504 and set up environment
   (a) Set system environment variable path to C:\Emgu\emgucv-windesktop 3.1.0.2504\bin\x86
3. Open the project solution with Visual Studio and run

7.3 Hardware Required

We chose to use the Microsoft Xbox 360 Kinect. The Kinect has an RGB camera as well as two 3D depth sensors. These depth sensors should help us create a depth map that can be used to determine the distance of a person on the screen, which will be vital in determining speed.



The Kinect was made for the Xbox 360 which meant we had to buy an adapter that will allow us to plug it into a female USB port. Below is an image of the adapter that will be used to do so.

Both these hardware components aren't necessary for every system to be implemented. Theoretically, other cameras may be used depending on the object being tracked. We chose to use the Kinect because of price, convenience, and the ability to detect people. The combination of these two hardware pieces will allow us to create a live video for our program to detect objects which will help us obtain our goal.

8 Learning New Technology

Our team knew very little about the technology that we were going to be using coming into the project. This meant we had to do tons of research about the Kinect, differnt tracking algorithms using computer vision, as well as the different variables and challenges when trying to calculate the speed of an object using a camera. This meant our primary resource was Google. We spent a lot of time researching everything we needed. The following is a list of main resources we used when researching information.

- EmguCV documentation - http://www.emgu.com/wiki/index.php/Main_Page
- EmguCV code examples - http://www.emgu.com/wiki/index.php/Code_Gallery
- StackOverflow forum - https://stackoverflow.com/
- MSDN Kinect documentation - https://msdn.microsoft.com/en-us

9 What We Learned

This section will go through each individual member of our team and discuss what we learned over the course of the year. Each member will answer specific questions regarding this year.

9.1 Alexander Bailey

9.1.1 What technical information did you learn?

During this capstone project I learned about using the Kinect, about the various ways that it could be used, how to use its depth map, about the various functions that is came with. I also learned about C# and its similarity and difference to C++. I learned about using even driven programming. Up until this point in my computer science career, my programs have been almost entirely programs which followed a set order, first one thing then another then another. However, with this project, the flow of the program depended largely on human interaction and the flow of the Kinect, as I did not know when the Kinect's data would be ready. Finally, with this project I gained experience working with a program GUI, designing it to be simple and appealing, as well as the code to handle the button clicks.

9.1.2 What non-technical information did you learn?

Working on this project has taught me a lot about how the flow of a project can go. As this is the first time that I have had to be a part of a large scale project from start to finish.

9.1.3 What have you learned about project work?

This project has shown me how much of project is planning and paperwork. The importance of planning out a project so that future problems can be minimized. Documentations is also important because there will almost always come a time when the project will be used for someone else and if it is no properly documented, then it will be almost entirely useless or the other party will need to spend a large amount of time digging into the program just to understand how to use it.

9.1.4 What have you learned about project management?

During this project I have realized that it is important not to overreach when planning a project, because this can lead to many problems down the read. It is best to do enough research so that it can be determined how difficult each goal is and how much can be done given the limited time attached to the project. It is a good idea to set more reasonable goals for a given project and then when those are achieved, to take the project a step further.

9.1.5 What have you learned about working in teams?

I have learned that is important to meet as a group on a regular basis in order to make sure that everyone is on the same page. Meeting regularly also helps to keep everyone on task and motivated. It is also a good idea to set group goals, giving everyone something to strive for in the immediate future.

9.1.6 If you could do it all over, what would you do differently?

If I could do this project over, the main thing I would do is to have a more realistic goal, as our original design document was too much. I would also make sure that we met at least twice a week, possible more as well as setting very specific weekly goals in order to make sure that we were making good progress.

9.2 Dylan Washburne

9.2.1 What technical information did you learn?

This project taught me about the software development cycle in general, as though I have held programming positions in the past, I have never been a core part of the design teams, nor have I even had nearly as much power in the building

process. While historically I have been assigned to maintain and implement additional features, the act of creating a piece of software from nothing is an entirely new and unique challenge to me. I learned about computer vision, C#, repo management, and to a lesser degree, a number of features of Visual Studio.

9.2.2 What non-technical information did you learn?

On the non-technical side, I have learned about some more basic management techniques, though this was never expressly taught and the act of managing was shared among everyone on the team. I also learned about the act of writing documents which lay out the expectations of the project, and how to properly interact with the client throughout the entire process.

9.2.3 What have you learned about project work?

I learned how much work is actually done in the planning and preparation steps. I have never before had to so precisely document all possible outcomes associated with building a project before building any part of it, but I must say that having so precisely mapped out what I wanted to be done, it certainly streamlined steps later. That said, I also learned how to properly assess work loads associated with individual tasks, as my estimates were way off.

9.2.4 What have you learned about project management?

I have learned that projects don't necessarily need to have a team leader, however it is beneficial to delegate actions to people so there is a sense of direction to pursue. In fact, I'd say that simply having some objectives available to be completed is the most important part, because then people can simply pick an objective and pursue it.

9.2.5 What have you learned about working in teams?

Teams are really just a group of people who should (at least in theory) be pursuing a common goal between all of themselves. For longer-term projects, this process is made more convenient when the people can get along and effectively use each other at any given time to optimize the performance of everyone involved. I don't think we hit that point, but I do think that we have a good rhythm going where we can work efficiently until we see a decrease in performance, where we are able to cut it off and regroup at a later time. That later time can be in a few minutes or the next day, but the important part is that we got what we needed complete.

9.2.6 If you could do it all over, what would you do differently?

I would absolutely have started on every project at least a couple days earlier, and I would have aimed way lower in the requirements documents. The starting earlier is somewhat self-explanatory, to even out the workloads, but even in the cases where we were fine holding off, I would rather have done a couple pieces per day than create a time crunch. As for the requirements, that actually stems more from me not wanting to tell the client "we messed up our predictions" more than anything else. Our client was remarkably accomodating, but I still did not like walking up to him and saying we could not complete what was promised, albeit a bit foolishly promised.

9.3 Benjamin Wick

9.3.1 What technical information did you learn?

This project has taught me a lot about UI development in visual studios, computer vision, and the Microsoft Kinect. This was my first project where we designed a user interface. Visual studios ended up being a very useful tool when

doing this. It made things incredibly easy to design and had the exact tools we needed to create our simple application. I also learned how powerful computer vision can be. When doing research on OpenCV, there were many different example projects ranging from facial recognition to changing and distorting images. I learned how to research and use the OpenCV library to function the way we needed it to. Last but not least, I also learned how the Kinect can be a powerful development tool. The Kinect was capable of things we did not expect including creating a 3D depth image to detect how far objects are in relation to the camera.

9.3.2 What non-technical information did you learn?

One valuable thing I learned from this project is the ability to go out and research information on my own. This is a valuable skill that can be easily used in the future on the job. I also learned about the process of project development as well as the client and developer interaction aspect. I learned more about how to write well-written documents that are important to project development.

9.3.3 What have you learned about project work?

Specifically I learned that it is really important to plan out and design your project. Having a good plan and design can make the project flow a lot easier during the implementation phase. This means putting in the research needed for your project as well as creating a timeline of what you plan to accomplish.

9.3.4 What have you learned about project management?

I learned that creating specific weekly goals that the team intends on accomplishing is incredibly useful. This way it makes the project more manageable and having a set goal gives sense of accomplishment to the team.

9.3.5 What have you learned about working in teams?

One thing I learned about working in a team is that dividing and conquering is the best method to accomplishing tasks. We ended up dividing work between our team members which allowed us to easily write the documents and implement things at a faster pace.

9.3.6 If you could do it all over, what would you do differently?

If I could go back I would set the requirements lower at the beginning. When we started the project, we kind of had the freedom to set the requirements ourselves as our client told us this was mainly our project. We all had an idea in mind but we did not really know how difficult it was. We ended up having to scale the project back. If I could go back and re-do it, I would set the requirements a little lower and set more difficult sounding tasks as stretch goals or add them later in the project.

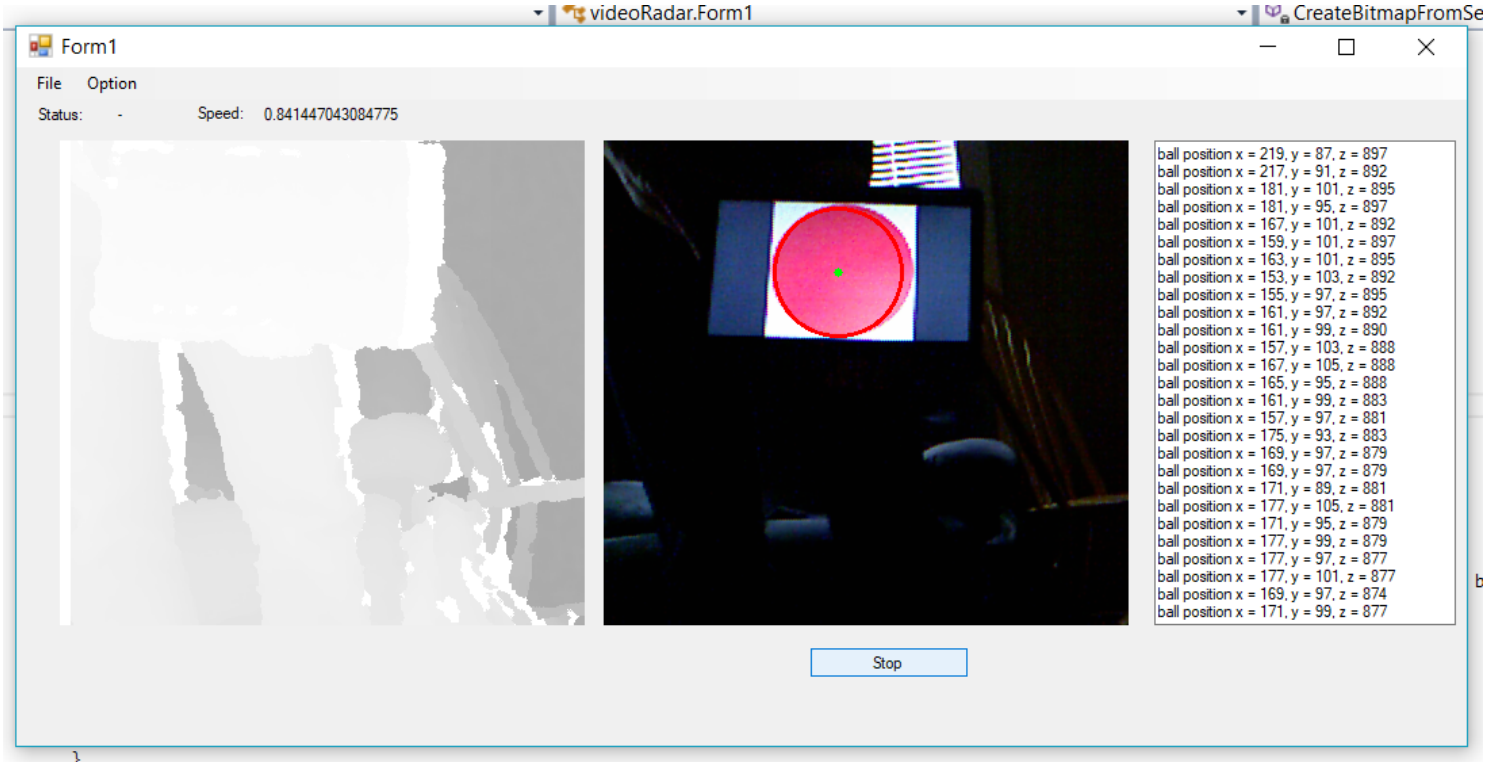
10 APPENDIX 1

```
void ProcessFrame()
        {
            // Create camera instance
            // originalImage = captureCamera.QueryFrame();

            // Create instances
            hsvImage = new Mat(originalImage.Size, DepthType.Cv8U, 3);
```

```
upper_red_hue_range = new Mat(originalImage.Size, DepthType.Cv8U, 1);
lower_red_hue_range = new Mat(originalImage.Size, DepthType.Cv8U, 1);
processedImage = new Mat(originalImage.Size, DepthType.Cv8U, 1);

// Convert to HSV image
CvInvoke.CvtColor(originalImage, hsvImage, ColorConversion.Bgr2Hsv);

// Create range of hue value for red and then add both to the processedImage
CvInvoke.InRange(hsvImage, new ScalarArray(new MCvScalar(0, 155, 155)), new ScalarArray(new
    MCvScalar(20, 255, 255)), lower_red_hue_range);
CvInvoke.InRange(hsvImage, new ScalarArray(new MCvScalar(160, 155, 155)), new
    ScalarArray(new MCvScalar(179, 255, 255)), upper_red_hue_range);

CvInvoke.Add(lower_red_hue_range, upper_red_hue_range, processedImage);
CvInvoke.GaussianBlur(processedImage, processedImage, new Size(3, 3), 0);

Mat structuringElement = CvInvoke.GetStructuringElement(ElementShape.Rectangle, new Size(3,
    3), new Point(-1, -1));

//CvInvoke.Threshold(processedImage, processedImage, 10, 255, ThresholdType.Binary);
//CvInvoke.BitwiseAnd(mask, s, mask, null);

CvInvoke.Dilate(processedImage, processedImage, structuringElement, new Point(-1, -1), 1,
    BorderType.Default, new MCvScalar(0, 0, 0));
CvInvoke.Erode(processedImage, processedImage, structuringElement, new Point(-1, -1), 1,
    BorderType.Default, new MCvScalar(0, 0, 0));

CircleF[] circles = CvInvoke.HoughCircles(processedImage, HoughType.Gradient, 2.0,
    processedImage.Rows / 4, 100, 50, 0, 0);

// Drawing circles around objects
foreach (CircleF circle in circles)
{
    CvInvoke.Circle(originalImage, new Point((int)circle.Center.X, (int)circle.Center.Y),
        (int)circle.Radius, new MCvScalar(0, 0, 255), 2);
    CvInvoke.Circle(originalImage, new Point((int)circle.Center.X, (int)circle.Center.Y),
        3, new MCvScalar(0, 255, 0), -1);
}

tick--;
if (circles != null && circles.Length != 0)
{
    //this.speed_val_label.Text = ((ushort)depth_data[(depth_data_width *
        (int)circles[0].Center.Y) + ((int)circles[0].Center.X)]>>3).ToString();

    x_pos = (int)circles[0].Center.X;
    y_pos = (int)circles[0].Center.Y;
    z_pos = (ushort)depth_data[(depth_data_width * (int)circles[0].Center.Y) +
        ((int)circles[0].Center.X)] >> 3;
```

```
            double x_disp, y_disp, z_disp;

            if (tick <= 0)
            {

                if (textBox1.Text != "")
                {                           // if we are not on the first line in the text box
                    textBox1.AppendText(Environment.NewLine);         // then insert a new line char
                }

                textBox1.AppendText("ball position x = " + x_pos.ToString() + ", y = " +
                    y_pos.ToString() + ", z = " + z_pos.ToString());
                textBox1.ScrollToCaret();             // scroll down in text box so most recent
                    line added (at the bottom) will be shown


                tick = tick_rate;

                x_disp = (x_pos - x_prev) * z_pos;  //147*2250 = 310*1097
                y_disp = (y_pos - y_prev) * z_pos;
                z_disp = Math.Pow(z_pos - z_prev, 2);  //1 ft = 304.8mm
                double speed = ((Math.Sqrt(Math.Pow(x_disp, 2) + Math.Pow(y_disp, 2) +
                    Math.Pow(z_disp, 2))) / Math.Pow(304.8, 2) * (30/tick_rate));
                                //sqrt(x^2+y^2+z^2) * pixels-to-feet

                this.speed_val_label.Text = speed.ToString();

                x_prev = x_pos;
                y_prev = y_pos;
                z_prev = z_pos;

                // write lines of text to the file
                datafile.WriteLine(x_pos.ToString() + "," + y_pos.ToString() + "," +
                    z_pos.ToString() + "," + speed.ToString());
            }
        }

        imageBox1.Image = originalImage;
        //imageBox2.Image = upper_red_hue_range;
    }
```

11 APPENDIX 2



Here is the final version of our GUI.