

# **Active Surveillance as a Service (ASaaS)**

Nicholas White  
Andrew Fisher  
Robert Marshall  
Mark Heller

*Department of Electrical and Computer Engineering, Cleveland State University*

Titus Lungu  
Brahm Powell

*Department of Mechanical Engineering, Cleveland State University*

Submitted to—

## **Faculty Advisors**

Dr. Pong P. Chu

*Department of Electrical and Computer Engineering, Cleveland State University*

Dr. Majid Rashidi

*Department of Mechanical Engineering, Cleveland State University*

## **Company Advisor**

Dr. Joseph Kovach

*Parker Hannifin Corporation*

This page intentionally left blank...

## **Contents**

Executive Summary	4
Problem Statement	5
Background	5
Detailed Design	6
Future Work	25
Professional Awareness	26
Project Management	35
Updated Gantt Chart	26
References	27
Appendix A: Relevant Portions of Fall Semester Report	x
Appendix B: Market Analysis and Business Strategies for Full Scale Implementation of ASaaS	x
Appendix C: Résumés of Team Members	x

## **Executive Summary**

The purpose of this document is to present a detailed description of the Intelligent Visual Surveillance software being produced for use by law enforcement to more effectively combat crime. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders of, as well as the developers of the system and is being proposed to the project's faculty advisors, as well as the project's company advisor, Dr. Joseph Kovach, for its approval.

This software system is a multi-component system designed for local law enforcement to use throughout an urban city in order to detect crime and track suspects in real-time<sup>1</sup>. This system is designed to maximize the efficiency of law enforcement by providing an intelligent<sup>2</sup> security system that eliminates many of the issues that exist with the current model of police work. By allowing law enforcement to track suspects as they flee a scene, a more focused response from law enforcement will result. This will, in turn, aid law enforcement in yielding more arrests, which leads to a lesser number of repeat offenders, and a safer community.

More specifically, this system is designed to allow police officers to view, manage, and respond to events (dubbed 'chases') as they happen in real-time, as opposed to the current model which forces police to respond only after an event has been reported (which can be quite some time after). The software facilitates communication between the already existing security cameras located throughout a city and a multi-user, multi-camera tracking algorithm. By linking these two with a web, mobile, and desktop facing client, the full utility of the algorithm can be reached since users will be able to interact with the system, adapting its decisions in real-time and adding human common-sense to the decision-making process.

---

<sup>1</sup> Although the system was initially designed for campus police, it has been shown that it may be scaled up, or down to meet the needs of many different security applications.

<sup>2</sup> Intelligent in this sense means the system has the ability to make decisions based on the data it receives, and is not passive (does not simply pass video through to a hard drive).

In addition to real-time tracking, the system contains a relational database for storage and retrieval of any and all relevant information regarding a chase, as well as general activity<sup>3</sup>. Videos, suspect profiles, and other related information will be stored here as well. This information can be loaded into a data warehouse, where exploratory analytics can be performed on the data. This sort of analysis will lead to a knowledge base regarding the location, frequency, and severity of crimes that cannot be generated to the same granularity with the current model of policing and data collection.

Solutions such as VSaaS (Video Surveillance as a Service) exist today, but are limited in scope and effectiveness. VSaaS provides video surveillance, but nothing more than mere logging of video with timestamps. ASaaS, in contrast, is an active system, meaning it tracks individuals, and communicates with a tracking algorithm, and operational database in real time. The objective of this project is to encompass all of these goals into the world's first Active Surveillance as a Service solution.

## **Problem Statement and Background**

Based on a poll of students at Cleveland State University, the consensus is that security (in this case on an urban college campus) is inadequate. A preliminary study showed that of the thirty students consulted in the engineering department, only three felt as if they were truly safe when traveling on campus (it is important to note that all three of these students are commuters, and are not in the city for more than a few hours at a time). This uneasiness is primarily attributed to the simple fact that law enforcement cannot be everywhere, and they surely cannot be everywhere at all times. This leads to heightened criminal activity, and a feeling of insecurity. While having law enforcement officers being physically present is key to providing a united and visible front against crime, the methods and procedures employed by these officers are not sufficient for achieving a high enough rate of capture to deter future crime.

The current methods in place for detecting a crime are manual. A crime is reported when someone generates stimuli in the form of a phone call to 911, the push of a campus

---

<sup>3</sup> General activity might be the number of individuals in an area, average traffic in an area, etc.

police emergency beacon button<sup>4</sup>, or the use of the Viking Shield<sup>5</sup> mobile application. All of these stimuli take time to generate, and are most often generated after a crime has occurred. From that point on, the police are directed to the area in which the crime *happened*, but, where is the criminal *currently*? By this time, the criminal is usually far away, either hidden away or having escaped completely. This is a major issue with the current method of policing. Law enforcement arrive where a crime happened sometime after it happened. With this model, it is clear to see that law enforcement are running a relay race in which the criminals are given a five-minute head-start.

Another issue with the current method of policing is suspect/victim identification. If a crime occurs in an area in which five people were present, how do the police know upon arrival who is innocent, and who should be investigated further? The current method is to ask questions of everybody, view physical evidence (broken objects, injuries, etc.). While this method often produces accurate results, the chance that an innocent person might be processed wrongfully is not a chance that is acceptable. The time law enforcement spends piecing back together the scene of a crime is time that is not being spent tracking the suspect(s). The best manner of describing the current model is to call it the *reactive* model. In the reactive model, everything is a response to stimuli, as opposed to a *proactive* model<sup>6</sup>, or *concurrent* model<sup>7</sup>.

## Detailed Design

Many changes have been made to the project since its inception in an effort to increase the performance of our system as well as to simplify it for scalability and faster response time. These changes include a redesign of the system architecture, changing the look and feel of the front-end application, creating a simpler computer vision algorithm,

---

<sup>4</sup> Emergency beacons are placed around campus. These beacons have buttons that call local law enforcement. However, the placement of the beacons is static, and is therefore predictable to criminals.

<sup>5</sup> Android: <https://play.google.com/store/apps/details?id=com.viking.shield&hl=en>, iOS: <https://itunes.apple.com/us/app/viking-shield/id698105969?mt=8>

<sup>6</sup> Proactive model – a model of police work in which actions are taken to actively reduce crime given a certain set of criteria. E.g. law enforcement chooses to question a suspicious looking individual because they are loitering around an “off-limit” area.

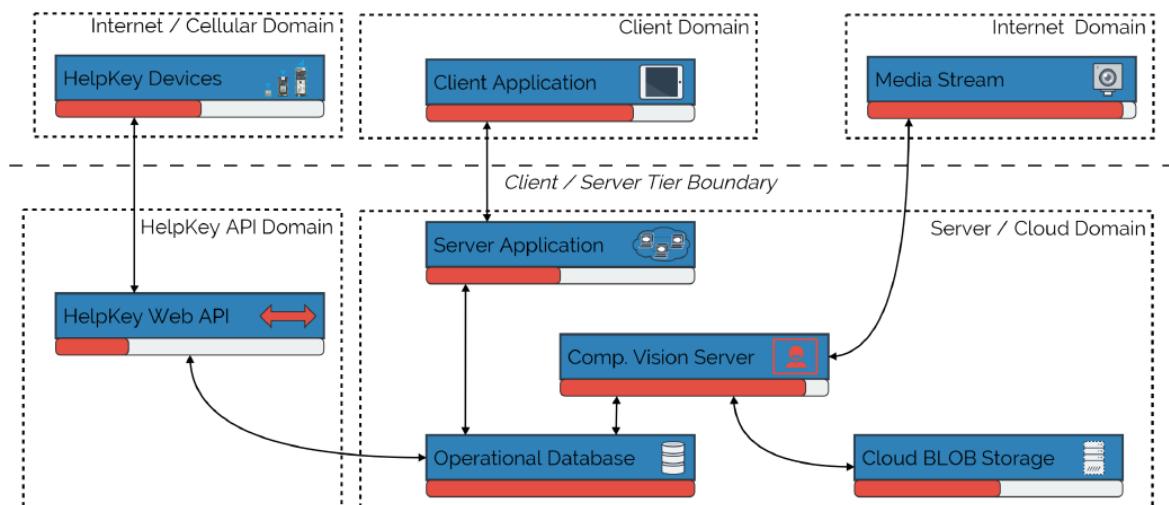
<sup>7</sup> Concurrent model – a model of police work in which actions are taken as a crime is happening. Stimuli is not needed as law enforcement is already aware of the crime in real-time, and can act in-step with the criminals (not achievable without active surveillance systems).

and designing the HelpKey, a distress signaling device to trigger a chase in our application. The design of this project has changed with time, as many engineering projects do. The major components within the design, however, have remained the same. These three components are: the system's architecture, or how the system will be implemented and supported, the computer vision algorithm, and the hardware component. These three major parts make up the full stack security suite.

## Data Flow

The architecture of the system is the component that facilitates the application. Included in this architecture is the client application, the server applications, the database design, and the many other components. These parts work together to form a cohesive data flow that allows the application to function as specified. The data flow diagram for this is as follows:

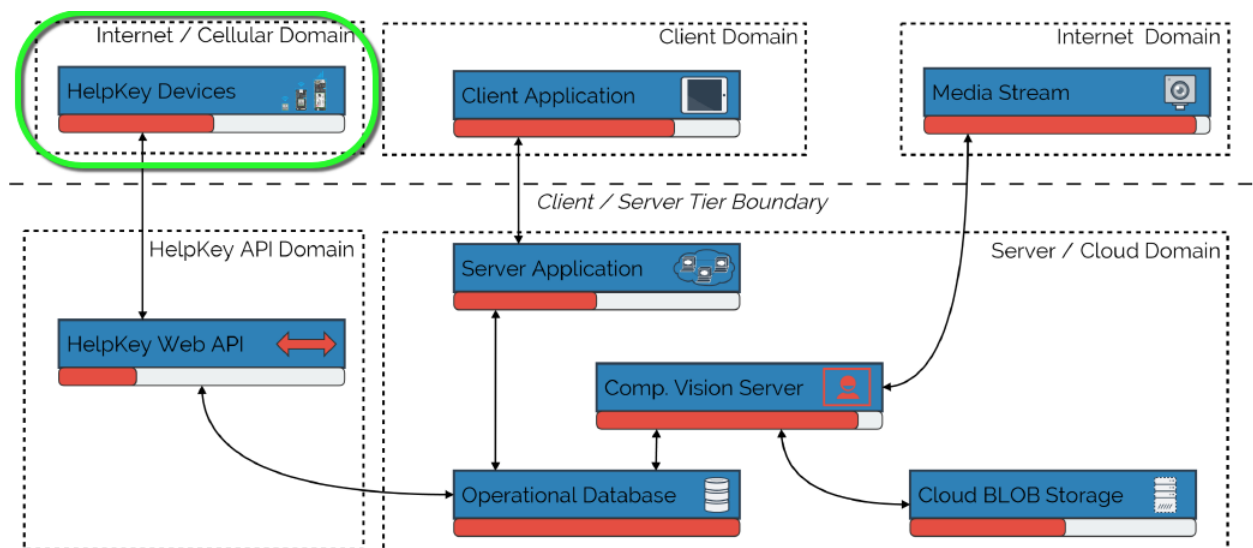
This diagram shows how data flows through the system, and between servers. Each of these components will be described in detail, but first, a general overview of the data flow will be provided here.



To begin, we will start with the database, shown in the bottom of the diagram. The operational database is the persistence layer in this application, and is what allows the

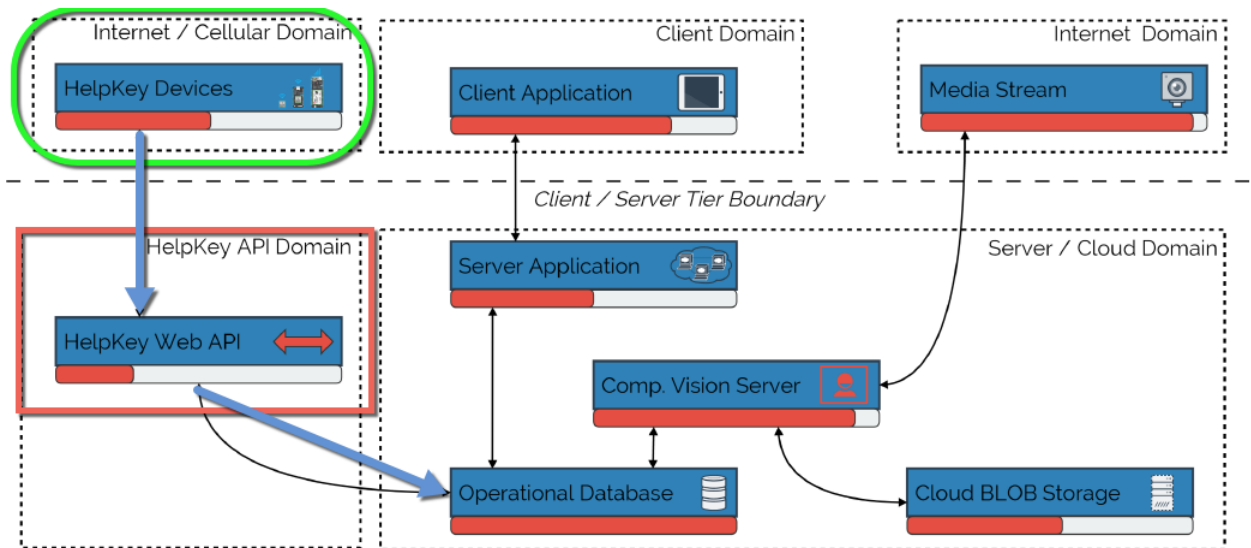
application to scale horizontally across multiple servers. This data store is used for persisting data from several sources, including: the computer vision server, the hardware JSON API, and the web application server. The operational database is fed by the computer vision server, which is fed by media streams from the internet (IP Cameras). The database is also fed by the web application server, which is in turn fed by the client application, which is modified by user input. Lastly, the database is fed by the hardware when it sends messages using a JSON API.

To best illustrate the flow of data through this diagram, we will follow a simple scenario. Let's imagine that a crime was just committed, and the victim managed to press their HelpKey device. This device is shown in the image below as having a green box around it, this will be our starting point.

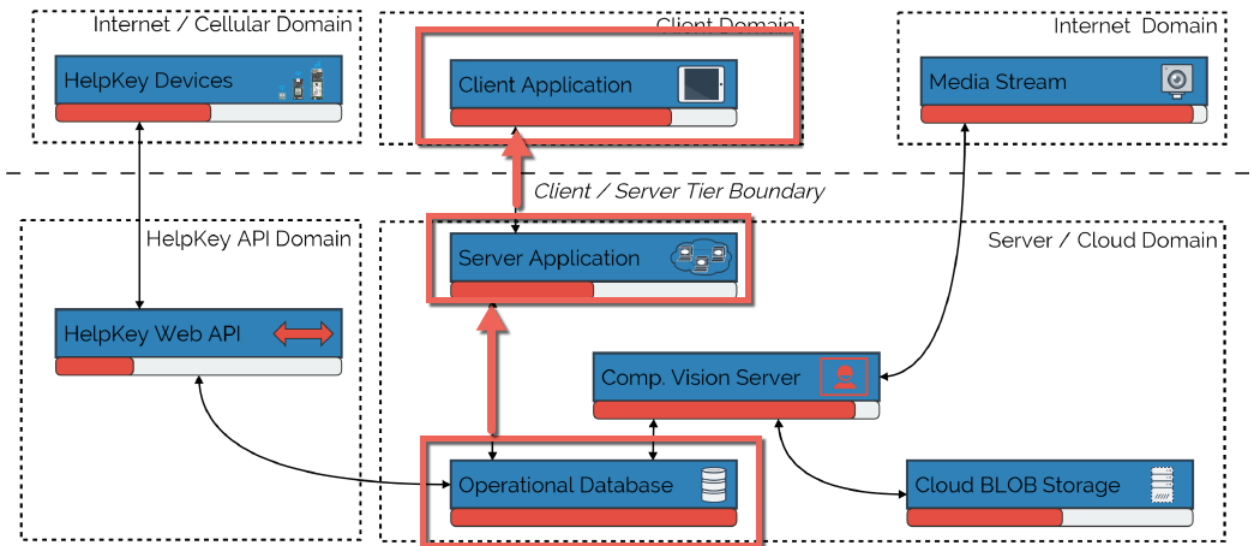


Once this happens, the data from the embedded system is sent using a JSON API (shown in the picture below in red) to the database. Since the format of the JSON data is known, it is simply inserted into the database. Once this happens, a stored procedure in the database cleanses and formats the data if it was not already formatted.





Once the data is placed into the database, the server application (which is listening for database changes) kicks off a process which updates all client applications as to the start of a crime, as seen in the diagrams below. This process usually takes about three seconds in total.



The other half of the application is the computer vision part. The details of the algorithm are defined in the following sections, but the infrastructure this runs on is defined here.

First, metadata about sensors (in this case IP cameras) is retrieved from the database and stored on the computer vision server. This server then spins up and instance of the

computer vision algorithm for this stream of video. The algorithm processes the video, extracts metadata to send back to the database, then stores the stream onto the BLOB storage server. This process happens continuously, without the client application being aware of it. Once the data is available in the BLOB storage, the client can then begin streaming this data in the form of video. This completes our process from video stream to client.

## Components

### *Client application*

The client application for this project was a web application. This application runs in the browser, and is the only way law enforcement can interact with the system. The client application was built using popular web technologies as described in the following table:

Technology	Purpose
HTML5	Client side markup and structure
CSS3	Styling and positioning
JavaScript	Used for DOM manipulation, as well as asynchronously loading data when needed. Used to communicate to the back-end server without needing to refresh the page and interrupt the user.
Angular.js	Also used for asynchronous communication. Built on top of Javascript.
Node.js	Used for real-time updating of charts, and the map. Also used for the chat feature within the client application.

These technologies were deployed to a Microsoft Azure hosted web application, managed by the team.

### *Server Application*

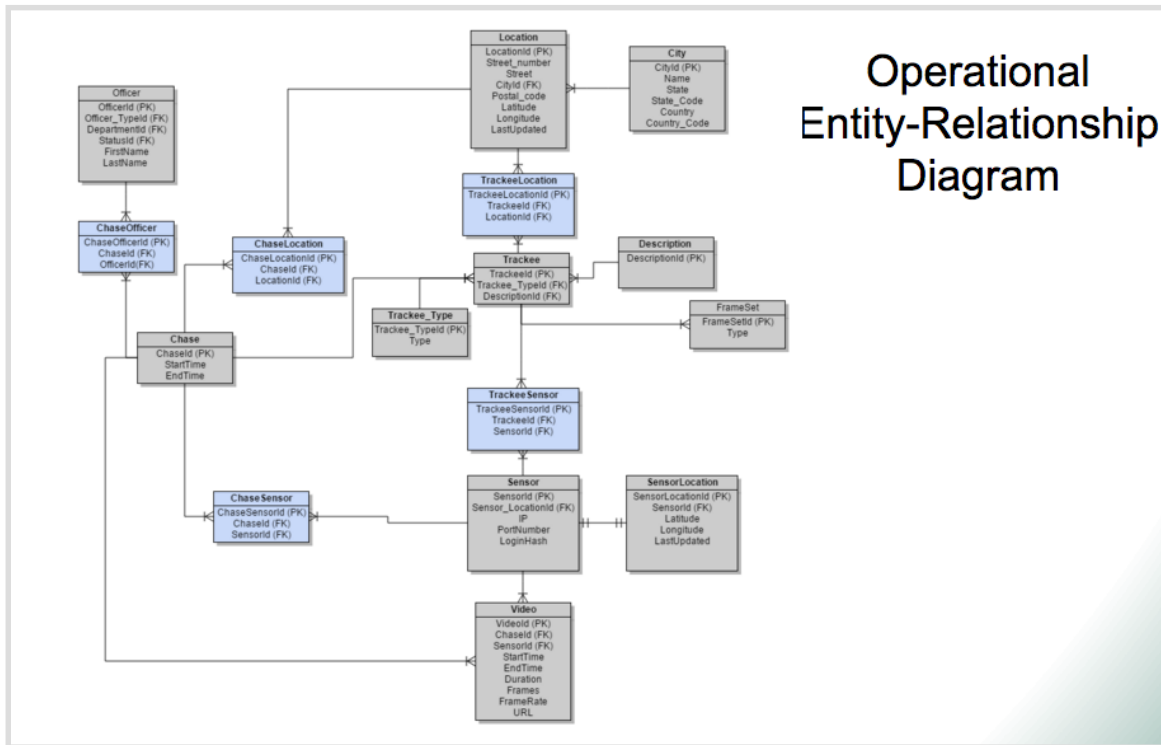
The server application was built using Microsoft Azure and the .NET Framework. We used ASP.NET MVC 4 specifically because it uses a Model View Controller design pattern. This allowed us to implement the UI in a way we saw fit, while leveraging the tools that existed for rapid prototyping. The job of the sever is to handle all incoming HTTP requests. This includes requests for resources, video feeds, map updates, etc. This server communicates with the back-end database to provide valid data to the browser's views (HTML) and models (JSON).

### *Database*

The database for this application was built using Microsoft SQL Server 2014, hosted on MS Azure. This allowed us to have our RDBMS in the cloud, and so it was redundant, reliable, and accessible from anywhere. The model for the database is too large to display here, but it basically breaks down as follows:

Table	Purpose
Officers	Holds all the user information for law enforcement officers utilizing the application
Location	Wrapper for real-world model of a specific location, contains latitude, longitude, as well as dates when changed, etc..
Trackee	Table of people who are being tracked using the app. This table provides metadata about the individual so they can be found again on another camera.
Sensor	This table contains all the data needed to connect to a remote IP camera. IP, port number, username, password are all stored here.
Video	Keeps track of all the video streams and videos stored in BLOB storage. Is capable

	of retrieving all videos associated with a specific chase.
Etc..	The list continues with irrelevant helper tables, these are the essential ones.



## Computer Vision Algorithm

The computer vision algorithm has seen much progress over the past months. Several changes in the algorithm are discussed below, including the ability to stream live video from IP cameras across the globe, accurate detection of humans within videos, and a high-fidelity method for recognizing and tracking the same person throughout a video and across multiple cameras. While the algorithm has room for improvement, as discussed in the *Future Work* section, it is currently a working prototype which demonstrates the feasibility of our system for law enforcement use.

### *Live Video Analysis*

Our algorithm has been tested on numerous forms of video feeds, including pre-recorded videos, videos streamed live from IP cameras, videos of different qualities, and videos with a large diversity of backgrounds and lighting conditions. Streaming from IP cameras was tested with cameras from New York, Atlanta, and even Moscow. Human detection could be performed in real-time as the video was coming in. The tested scenarios ranged from snowy ski slopes to crowded shopping malls, all of which showed a high ratio of accurate to false detections.

### *Human Detection*

Our algorithm can effectively detect humans (Figure 1) in either pre-recorded videos or live feeds using a Histogram of Oriented Gradients and a Support Vector Machine, which is trained on several thousand images of humans. Currently, detection performs best in controlled environments or by appropriate tuning of “search parameters” based on lighting conditions, background interference, clarity of the image, and size of humans in the image. With appropriate tuning, 80% to 100% of humans in standing position can be accurately detected on a consistent basis within the video feed of a stationary camera. False detections sometimes occur due to tall standing trees whose branches may cause resemblance to the physical form of a human body.



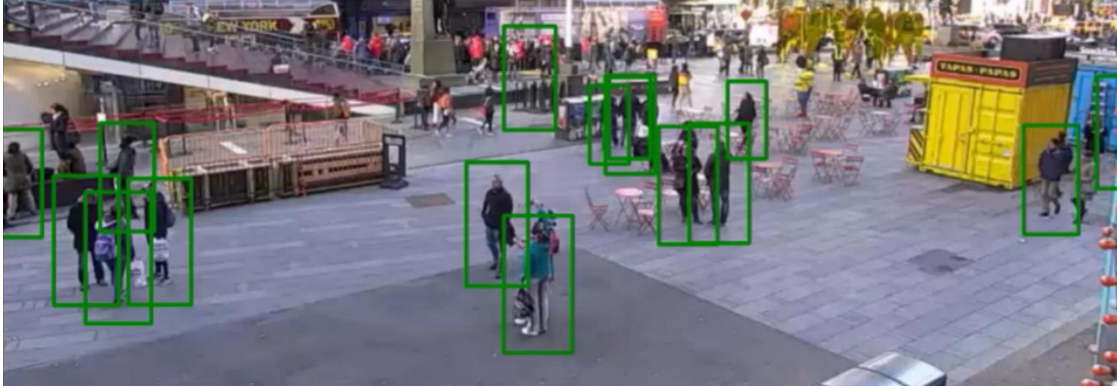


Figure 1: As depicted, human detection performs well in many different locations, though with different accuracy depending on the surrounding conditions. Colors have no significance here and the algorithm is simply looking for humans at this stage. It is important to note that with the current training data for human detection, only standing humans can be identified. There are plans to improve this, as described in subsequent sections of the paper.

### *Recognition and Tracking*

Every person detected has several types of information associated with them in our database. This includes a timestamp of last detection, cameras in which they have been seen at each detection, physical location within the frame of the video, and their image from each detection. This allows for the ability to accurately track a person by recognizing that they are the same person as previously detected based on key data.

Several methods for tracking people in video feeds were initially proposed. As proved through extensive testing, these techniques are insufficient to gather appropriate, distinct, and consistent data about a person. When using RGB color patterns to differentiate between people in a video, it was found that a person's color pattern changes significantly based on lighting conditions and orientation of their body. The average RGB of different people was also not distinct enough where a person could be differentiated from another solely based on this metric.

The size of a person is also inadequate to track different people due to the difficulties in interpreting depth in flat images. While ratios can be found between the known sizes of objects and their size in an image, this method requires that a known image exists in every video feed. This is not a method we wished to use since our system is aimed at implementation across a large number of cameras, in which case we would not be able to ensure that known objects exist in every image to perform this comparative analysis.

Instead, metrics for tracking were found which were more adaptable to different environments and non-stationary people while providing robust data to find consistent traits attributable to a specific person. The first method that was tested for tracking involved learning a person's key identifying features based on visual data from all of the collected images of that person. This was done by training an artificial neural network but was found to be problematic. When a person is initially detected, only one image of them exists for the algorithm to use as training data. This provides insufficient data from which to extract feature vectors that are distinct for every person.

The entire infrastructure required for gathering image data and preparing to train a neural network is set up, and with some future modifications it will likely prove to be a very accurate tracking method. Each detected person's image is rearranged with all of its pixels in one row and labeled as positive training data. Negative training data is then added (using the images of all the other detected people) to create the training set of image for the neural network. After training for each person in the image, the neural networks are queried in the next frame and the person deemed most similar to a previously detected person is labeled as that person. His/her image then is added to the positive training set of images and the neural network is updated with the new data.

While several methods for recognizing and tracking people proved inaccurate, one method was found to work very well. This consisted of tracking people based on nothing but their location in the image, as seen in Figures 2 and 3. When a person is detected, the x and y position in the frame is recorded. In the next frame, all the x and y positions of detected people are compared to previous detections. In order for a person to be identified as being the same person as was previously detected, two conditions must be met. First, the current person must be the closest person to his previous location. Second, his previous location must be the closest previous location to his current location. This dual-condition requirement ensures that multiple people are not labeled as the same person, which may occur if two people are both closest to the same previous location of one of them.

Distance here is measured in pixels as an absolute distance, combining x and y distances into one. A list of positions is kept for every person. Some confusion may occur when individuals cross paths, but this is mainly due to the fact that we cannot currently detect a partially occluded person. Implementing a Gaussian Process Regression was tried



in order to predict the location of a person in the next frame in order to improve the performance of the algorithm. Due to the irregularity of human motion, however, a traditional machine learning regression cannot be implemented as is.

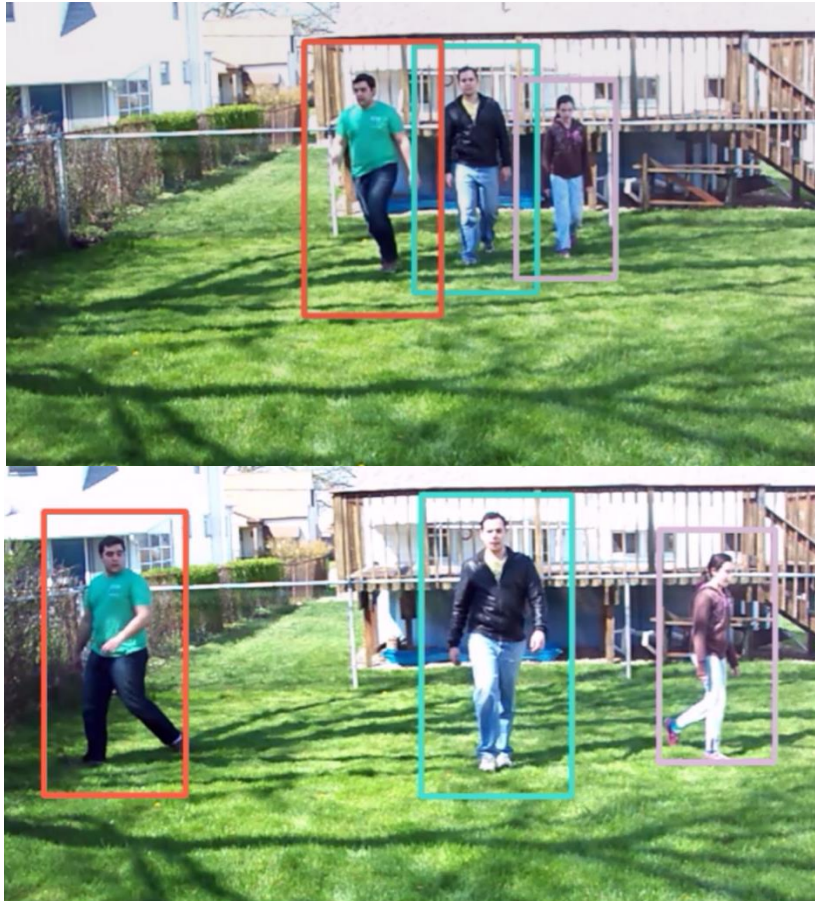


Figure 2: *In controlled environments, humans can be detected accurately and can be tracked as they move throughout the frame of a video, as seen here. Each specific color of the bounding boxes for each person signifies that the algorithms identifies that person as the same person as was previously seen.*



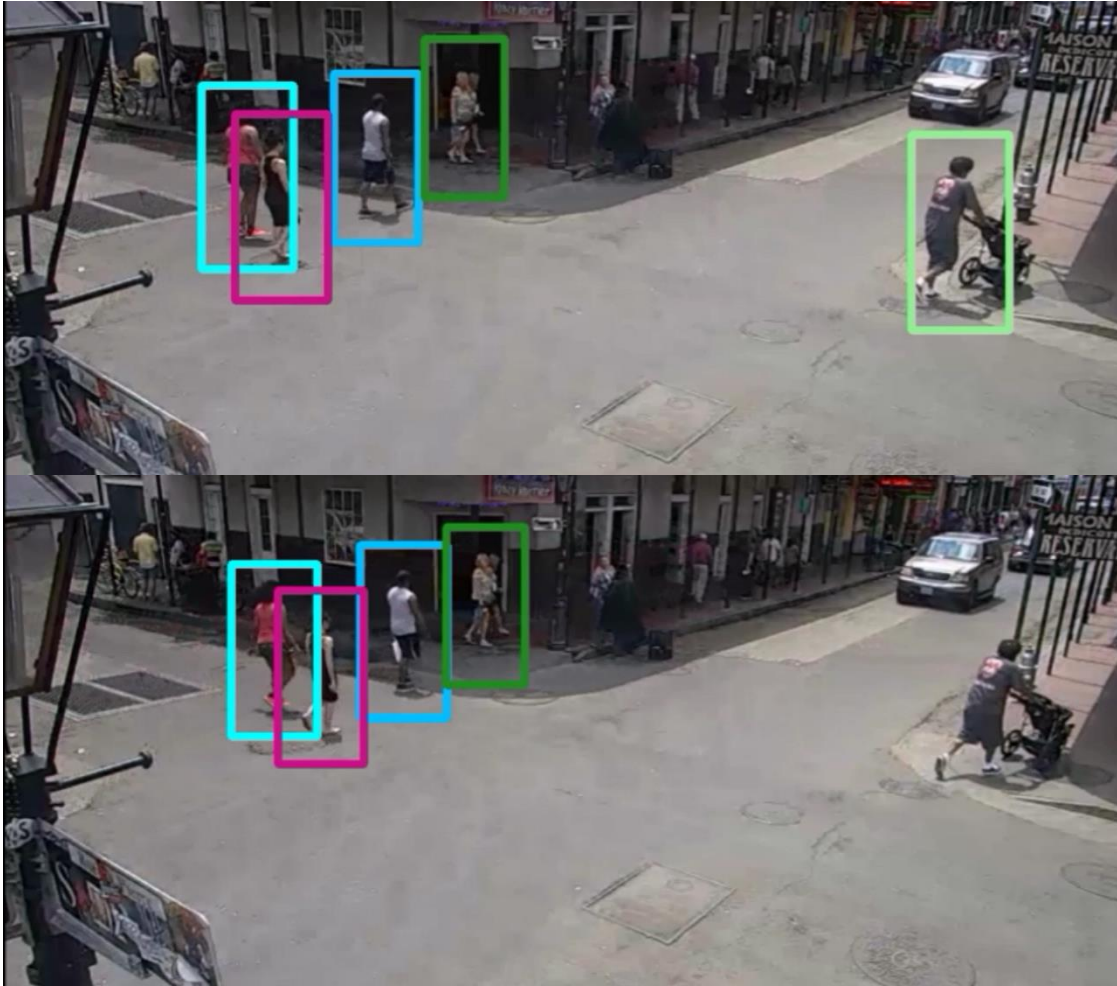


Figure 3: In slightly less controlled environments with many shadows, human detection still works relatively well, and tracking performs with high quality. In this case, some imperfections exist, such as the man with the stroller no longer being identified as a human in certain frames.

## Hardware

### HelpKey Introduction

In addition to the application we set out to develop, we also developed a hardware interface for the system. The hardware interface that we have developed resembles a key fob. However, this is no ordinary key fob. This hardware interface is known as the HelpKey. The HelpKey is a device that is designed to be a system actuator, which sends distress calls to the System. The HelpKey an internet-of-things device used to transmit a distress signal when a victim is in need of help. The distress signal comes into the database as the GPS location of the person calling for help, which gives law enforcement the

information they need catch a suspect when a crime is committed. We will get into further details about how this hardware is designed and implemented into our system.

### **HelpKey Development Process**

In order to achieve the obstacle of getting data to the database of the system, the HelpKey was designed to be connected to the cloud. To begin the implementation of this Internet-of-Things device we selected to use hardware from Particle. Particle provides a platform for developing Internet-of-Things products. The advantage of using Particle as our Internet-of-Things solution is the Particle + Microsoft integration which provides developers with the ability to connect to the cloud with Particle hardware, and then collect and analyze data using the Azure platform.



Courtesy of: <https://www.particle.io/microsoft>

We began development of the HelpKey using the Particle Photon; a Wi-Fi transceiver with a built in STM32F205RGY6 120Mhz ARM Cortex M3 microprocessor. The frequency band of the Particle Photon is 2.412GHz – 2.462GHz in the United States, uses sub channels 1-11. The Particle Photon comes with several peripherals which give it the ability to acquire several types of data to be transmitted to the cloud.

After learning what the Particle Photon can perform, we approached the problem of sending GPS data to our system. This was not difficult given that the Particle Platform is

open source. There is plenty of open source code that is available for developers to use for all sorts of projects. We were able to find a GPS library that supports Particle Products. The Library we used is “TinyGPS” which was developed by Mikal Hart. Then the library was adapted for the particle platform by Krishnaraj Varma. With this code available, we were able to quickly modify it for the use in our system. The TinyGPS library is compatible with any NMEA standard GPS hardware. For the HelpKey we used the MediaTek 3329 GPS module.

The MediaTek 3329 is a 66 channel GPS module with 10 Hz updates. Originally we used a MediaTek 3329 made by an unknown manufacturer. The module was purchased on Amazon. To start off, this GPS module was difficult to implement with the Library, and code example. After several test we were unable to get the GPS to connect, and obtain a valid GPS coordinate. After doing some research we found that the Baud Rate that was manually set in our firmware was incorrect. The MediaTek 3329 that we were using has a default baud rate of **38000 bits/sec**. The baud rate that we had manually set in the firmware was **9600 bits/sec**. The Particle Photon communicates with the MediaTek 3329 through UART serial communication using the TX and RX pins in each module. The TX pin of the MediaTek 3329 was connected to the RX Pin of the Particle Photon MCU, and the RX pin from the GPS to the TX pin of the MCU. In the Device Breakdown section of the paper you will be able to see a schematic for each device.

The firmware we originally used to test the Particle Photon and the MediaTek 3329 GPS module was the code example given with the TinyGPS library. This firmware grabbed the GPS location from the GPS module every fifteen minutes. Once we were able to prove the functionality of the GPS module, we then moved forward to customizing the firmware for the Particle Photon. In the Firmware section of this paper we will explain the process we designed for the HelpKey to meet our system requirements.

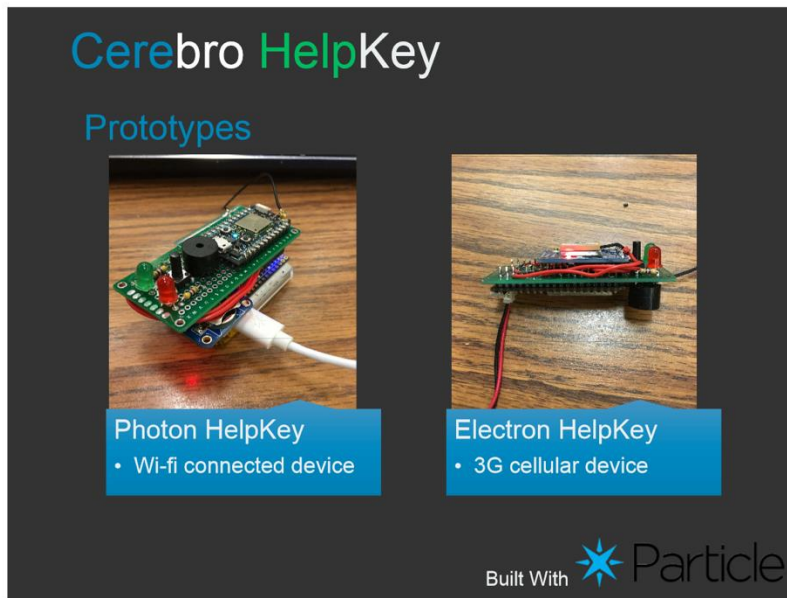
After completing our first Photon HelpKey prototype, we had a defined list of hardware needed for each HelpKey. At this point a Bill of Materials was drafted and sent to Paul to get the equipment ordered, and shipped to Cleveland State. Our Bill of materials is as shows in the figure below.

Item Description	Manufacturer	Product ID	Quantity Needed	Price Per Unit	Total
Photon Wifi Module with headers	Particle	N/A	2	\$ 19.00	\$ 38.00
Power Shield with LiPo Battery	Particle	N/A	3	\$ 25.00	\$ 75.00
Wifi flex antenna	Particle	N/A	3	\$ 10.00	\$ 30.00
Electron Data kit with headers	Particle	N/A	2	\$ 69.00	\$ 138.00
Adafruit Ultimate GPS (mt3329)	MediaTek/Adafruit	746	4	\$ 39.95	\$ 159.80
Piezoelectric Buzzer	Adafruit electronics	1536	1	\$ 0.95	\$ 1.00
Button (short)	Adafruit electronics	1119	one 10 pack	\$ 2.50	\$ 2.50
Button (tall)	adafruit electronics	1490	one 10 pack	\$ 2.50	\$ 2.50
LEDS (green)	Adafruit electronics	298	one 25 pack	\$ 4.00	\$ 4.00
LEDS (red)	Adafruit electronics	299	one 25 pack	\$ 4.00	\$ 4.00
Resistors (470 ohm)	Adafruit electronics	2781	one 25 pack	\$ 0.75	\$ 0.75
				Total from Particle	\$ 281.00
				Total from Adafruit	\$ 174.55

Two additional Particle Photons were ordered to continue development of the Photon HelpKey. We decided to also order three Particle Electrons to add to the HelpKey family. After doing much testing with the Particle Photon, we determined that the connection to the cloud with the Photon was unreliable. First, the connection to Wi-fi was not a strong connection due to the unreliable csuguest network. Second, we were not able to obtain a Wi-fi connection at a long range from campus buildings. This was a big issue, because a majority of the crimes that are reported to us are committed in areas that are secluded. The Particle Electron uses cellular data to send data to the cloud. With the Particle Electron we were able to obtain connections all over campus, and beyond campus. The only limiting factor with the Electron Help Key is that the GPS module does not obtain connection in buildings where it does not have a line-of-sight with the GPS antennas.

Also the Electron HelpKey improved on the size of the device. This is because the Electron has a built in power circuit, and a battery connector on board. Also the Electron can be programmed to monitor the battery circuit to determine the battery capacity on the device. The battery is a Lithium Polymer battery. The Electron HelpKey was programmed with the same firmware as the Photon HelpKey, but with the additional battery monitor function.

Prototypes of the Photon HelpKey and the Electron HelpKey can be seen in the figure below.

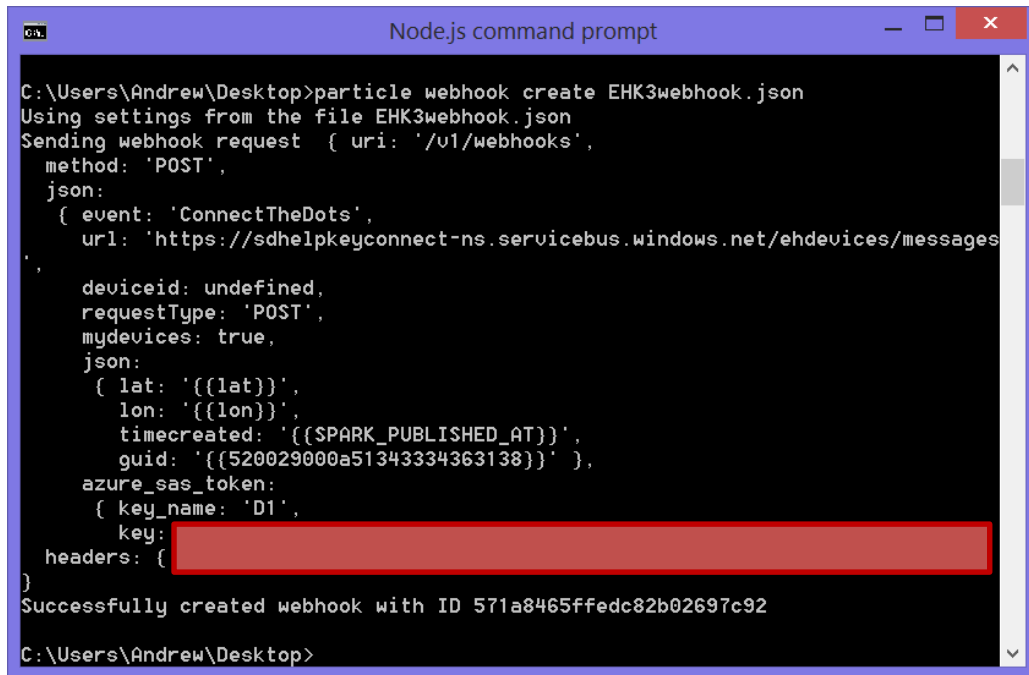


### Connecting HelpKeys to the Database

In order to send the data obtained from the HelpKey devices, they had to be linked to the Azure database using an Azure Event Hub. The Azure Event Hub is monitored by a query that obtains the data, and is then designed to pinpoint the location of the HelpKey on a google maps application. The first step to obtaining a connection between the HelpKeys and the Azure Event Hub is to create a webhook for each device. A webhook provides a path to the Event Hub using the URL of the Event Hub being used for our devices. Each webhook is unique to each devices using the ID # of the device. A webhook from one of our devices is shown below. We learned to link the HelpKeys to the database using a hands on lab in the [hackster.io](https://hackster.io) website.

```
webhook
{
  "event": "ConnectTheDots",
  "url": "https://sdhelpkeyconnect-ns.servicebus.windows.net/ehdevices/messages",
  "requestType": "POST",
  "json": {
    "lat": "{{lat}}",
    "lon": "{{lon}}",
    "timecreated": "{{SPARK_PUBLISHED_AT}}",
    "guid": "{{370044001447343432313031}}"
  },
  "azure_sas_token": {
    "key_name": "D1",
    "key": [REDACTED]
  },
  "mydevices": true
}
```

Each webhook was created using the Particle CLI command prompt. This can be seen in the Figure below.

A screenshot of a Windows command prompt window titled "Node.js command prompt". The window has a blue title bar with standard minimize, maximize, and close buttons. The command prompt shows the following text:

```
C:\Users\Andrew\Desktop>particle webhook create EHK3webhook.json
Using settings from the file EHK3webhook.json
Sending webhook request { uri: '/v1/webhooks',
  method: 'POST',
  json:
    { event: 'ConnectTheDots',
      url: 'https://sdhelpkeyconnect-ns.servicebus.windows.net/ehdevices/messages',
      deviceid: undefined,
      requestType: 'POST',
      mydevices: true,
      json:
        { lat: '{{lat}}',
          lon: '{{lon}}',
          timecreated: '{{SPARK_PUBLISHED_AT}}',
          guid: '{{520029000a51343334363138}}' },
      azure_sas_token:
        { key_name: 'D1',
          key: [REDACTED] },
      headers: { }
    }
  }
Successfully created webhook with ID 571a8465ffedc82b02697c92
C:\Users\Andrew\Desktop>
```

The "key" field in the "azure\_sas\_token" object is redacted with a solid red rectangle.

## Device Breakdown

The HelpKey devices were designed to be small devices that can be fit in a user's pocket. The device does not require many hardware devices. The center of the device that does all of the processing is the Microcontroller + Wireless Module. We have two HelpKey prototypes, one device uses the Particle Photon wi-fi module, and the other is the Particle Electron 3G cellular module. As mentioned before the MCU is the STM32F205RGY6 120Mhz ARM Cortex M3 microprocessor.

The sensor in the HelpKey device is the MediaTek 3329 GPS. The Particle Photon, and Electron communicate with the MediaTek 3329 through UART serial communication using the TX and RX pins in each module. The TX pin of the MediaTek 3329 was connected to the RX Pin of the Particle Photon MCU, and the RX pin from the GPS to the TX pin of the MCU. In the Device Breakdown section of the paper you will be able to see a schematic for each device.

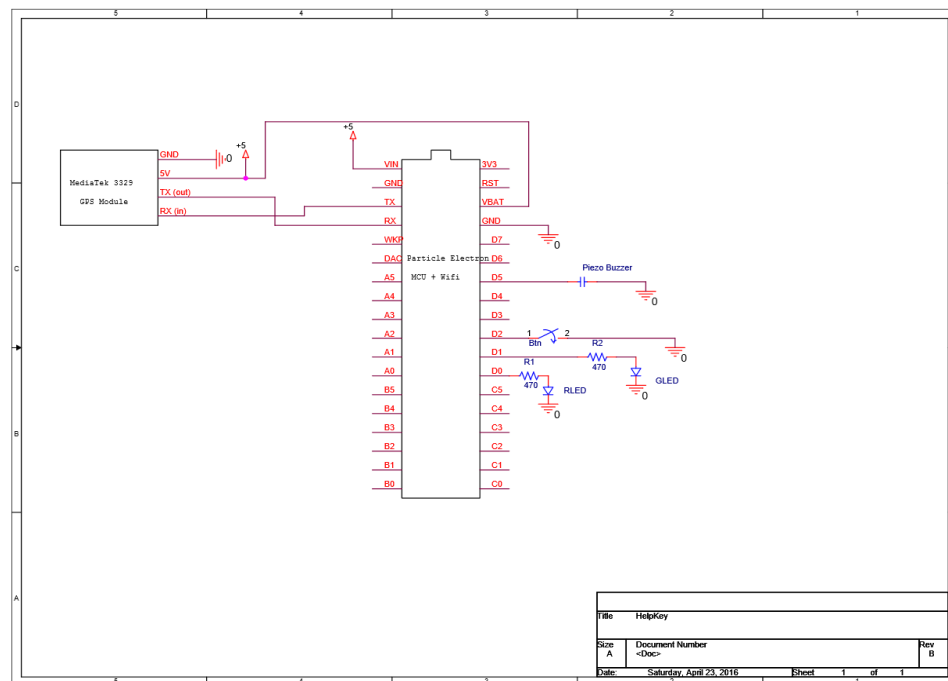
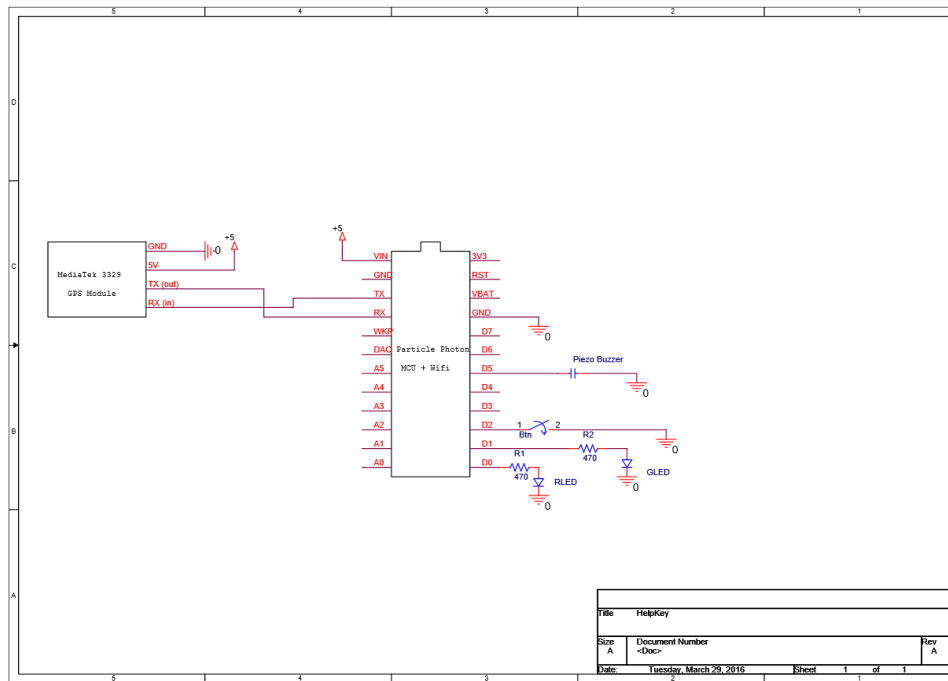
The one input to the HelpKey device is the Button which initiate the Firmware process of obtaining the GPS location and sending it to the cloud. Our hardware device outputs are an audible speaker used to initiate an alarm when the button is pressed. There are two notification LEDs used in the device. The green LED notifies the user when a valid GPS location was sent to the database to initiate a chase. The red LED notifies the user when a non-valid GPS location was sent to the database. In the future we plan to add a capability for the device to continue to try to sample the GPS module until a valid location is obtained. The final device output comes in the form of JSON data which contains the GPS raw data, and this goes to the database to be processed by streaming analytics. This data is used to pin-point a location on the google maps portion of the web application.

The device inputs, and inputs are connected to the MCU in the following matter:

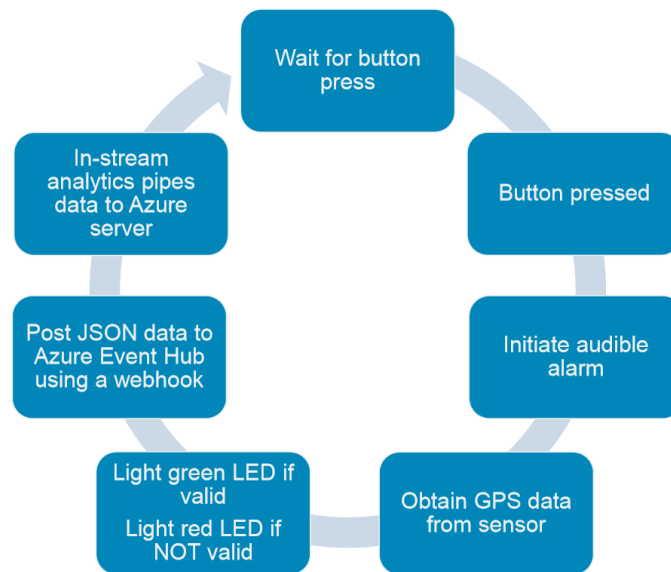
- Red LED => D0 digital input
- Green LED => D1 digital input
- Speaker => D5 digital input
- Button => D2 digital input
- GPS TX => RX of MCU + Wireless module
- GPS RX => TX of MCU + Wireless module
- GPS GND => GND of MCU + Wireless module
- GPS VIN => VIN on Photon, VIN & VBAT on Electron in order to operate on battery power

**All connections can be seen in Schematics below**





## Firmware



The firmware on both the Photon and Electron devices were nearly identical. It follows the simple flow chart shown above. Currently the photon waits for a change in voltage level on the D2 input. Once this change is detected the D5 pin output is written to a high logic voltage value activating the attached speaker. While the speaker emits sound the serial pins are used to communicate with the GPS device. This was done using the TinyGPS library. If valid coordinates are received the D1 pin output is written to a high logic voltage value. This received data is then stored temporarily. We then build a JSON packet and insert these received values into it. Once the JSON packet is built and filled we send it to our webhook using the built in publish method from particle. If non valid GPS data is received the D0 pin output is written to a high logic voltage value. When non valid GPS data is received the same process is followed but blank GPS data is sent in the JSON packet.

On top of the main firmware function a power monitoring function was also included in our firmware. While waiting for the button to be pressed it also checks the

battery state of charge which returns a floating point percentage of the state of the battery. If the battery has less than 15% the red LED flashes while it is low. This battery monitoring function uses a counter that increments by 1 for each check of the button state. This allows both functions to run simultaneously without the battery monitoring blocking the main help key function.

This power monitoring function was fully implemented on the Electron help key but the Photon was unable to use this function. The photon requires the power shield to implement any power monitoring functions. The functions for the Photon were different then the Electron and they did not behave the same way.

```

HelpKey_gps v3
// Built in TinyGPS library which contains pre built functions
// for receiving GPS data
#include "TinyGPS.h"

// Pin Outputs
int pled = D1;           // Pass LED (Green)
int fled = D0;           // Fail LED (Red)
int spkr = D5;           // Speaker

// Pin Inputs
int btn = D2;            // Pushbutton

// Global Variables
TinyGPS gps;             // From pre-built GPS library,
                        // holds latitude and longitude.

char szInfo[64];         // Holds GPS coordinates only for sending.

int sleep = 1 * 60 * 1000; // 1 Minute increment conversion.

int btnState;            // Holds the state of button to a true
                        // or false which is easier to understand.

float lat, lon;          // Floating point variables to hold latitude
                        // and longitude and values.

bool isValidGPS;         // Determines if a valid GPS coordinate was
                        // received.

int batteryCounter = 500; // Counter used for manually incrementing by
                        // one second during low power.

bool batteryLowLight = false; // Counter for flipping on and off during
                        // low power.

FuelGauge fuel;          // Used for finding the current charge of
                        // a battery plugged into the board.

float batterySOC;        // Variable to hold the state of charge as
                        // a percentage.

//*****
// setup Function: Sets up all the Pins for sending or receiving. The LED's and
//                 speaker are outputs so they can have voltages written to
//                 them. Button (btn) is set to be an input with a pull up
//                 resistor. So pressing the button will lower the input to low
//                 logic level. Serial starts serial reception for the GPS
//                 sensor with a given baud rate. Our GPS sensors have a baud

```

```

                                HelpKey_gps v3
//          rate of 38400 and 9600 for the other. The batterySoC is
//          gotten initially for the battery monitoring circuit.
//*****

void setup(){
    Serial1.begin(9600);
    pinMode(pled, OUTPUT);
    pinMode(fled, OUTPUT);
    pinMode(spkr, OUTPUT);
    pinMode(btn, INPUT_PULLUP);
    batterySOC = fuel.getSoC();
}

//*****
// loop Function:  Constant loop function that acts as our main for execution
//                  Contains all logic that will run continuously. The loop waits
//                  for a button press. When the button press occurs GPS data is
//                  received and if the GPS data is valid it will send it,
//                  otherwise it will send blank GPS data.
//*****

void loop(){
    batteryMonitor();
    btnRead();
    if (btnState == LOW){
        spkrHigh();
        gpsReceive();
        if (isValidGPS) {
            validGPS();
        }
        else {
            notValidGPS();
        }
        sendGPS();
    }
}

//*****
// btnRead Function:Sets the isValidGPS flag to false to start. Reads the input
//                  from the button as On or Off and uses it to control logic
//                  inside of loop function.
//*****

void btnRead(){
    isValidGPS = false;
    btnState = digitalRead(btn);
}

```

```

HelpKey_gps v3
//*****
// gpsReceive Function: Function from the GPS sample code. Receives GPS data
//                        from the serial input and parses it to check that it is
//                        a valid GPS location. It sets the isValid flag to true
//                        or false.
//*****

void gpsReceive() {
    for (unsigned long start = millis(); millis() - start < 1000;) {
        while (Serial1.available()) {
            char c = Serial1.read();
            if (gps.encode(c))
                isValidGPS = true;
        }
    }
}

//*****
// validGPS Function:   Function from the GPS sample code. Turns the Green LED
//                       on alerting the user that a valid GPS coordinate was
//                       sent. It then records the time the GPS coordinate was
//                       received. These three items are stored into the
//                       variables lat, lon, and age. Also builds these values
//                       into a career.
//*****

void validGPS() {
    digitalWrite(pled, HIGH);
    unsigned long age;
    gps.f_get_position(&lat, &lon, &age);
    sprintf(szInfo, "%.6f,%.6f", (lat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : lat),
(lon == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : lon));
}

//*****
// notValidGPS Function: Turns the red LED on letting the user know an invalid
//                        GPS was received and that no exact GPS coordinate was
//                        received. Builds a message with Blank GPS data to send
//                        to the webhook.
//*****

void notValidGPS() {
    digitalWrite(fled, HIGH);
    sprintf(szInfo, "0.0,0.0");
}

//*****
// notValidGPS Function: Uses the char arrays built in the valid and notValid GPS

```

```

                                HelpKey_gps v3
//                                functions and sends them to the particle dashboard and
//                                Azure webhook using the built in Publish function. Waits
//                                fifteen seconds then turns the LED's and speaker off.
//*****

void sendGPS() {
    // Storage container to build the json data into.
    char payload[255];

    // Builds JSON to send for the webhook
    snprintf(payload, sizeof(payload), "{ \"lat\": \"%f\", \"lon\": \"%f\"}", lat,
lon);

    // Sends the data to serial so it can then be sent over wireless
    Serial1.println(payload);

    //built in publish method for Particle, sends it to both particle dashboard and
webhook
    Particle.publish("ConnectTheDots", payload);
    Particle.publish("gpsloc", szInfo);

    // Waits 15 seconds before turning everything off and being allowed to send
information again
    delay(15000);

    // Turn them off
    digitalWrite(spkr, LOW);
    digitalWrite(pled, LOW);
    digitalWrite(fled, LOW);
}

//*****
// spkrHigh Function: Turns the speaker on and turns the Red LED off in case
//                    the power montior is also running. Made the code more
//                    readable.
//*****

void spkrHigh() {
    //digitalWrite(spkr, HIGH);
    digitalWrite(fled, LOW);
}

//*****
// batteryMonitor Function: Turns on the battery montioring circuit and turns
//                            the red LED on and off every minute. Uses a built in
//                            function called get state of charge. Uses this
//                            floating point value to wait for the battery to be
//                            below 15%. When it is it starts subtracting from a

```

```

//                                HelpKey_gps v3
//                                counter that will deincrement once every cycle of
//                                the loop function. When that counter reaches 0 the
//                                LED is switched from its current state to the next
//                                state either on or off.
//*****

void batteryMonitor() {
    batterySOC = fuel.getSoC();
    if(batterySOC < 15) {
        batteryCounter--;
        if (batteryCounter == 0) {
            batteryLowLight = !batteryLowLight;
            batteryCounter = 500;
            if(batteryLowLight) {
                digitalWrite(fled, HIGH);
            }
            else {
                digitalWrite(fled, LOW);
            }
        }
    }
}

```



### *References used*

Hart, Mikal *Tiny GPS*. <http://arduinoiana.org/libraries/tinygps/>

Varma, Krishnaraj. *Particle Tiny GPS* [https://github.com/krvarma/TinyGPS\\_SparkCore](https://github.com/krvarma/TinyGPS_SparkCore)

Particle Datasheets <https://www.particle.io/>

GPS Basics <https://learn.sparkfun.com/tutorials/gps-basics>

DeCarlo, Paul *Hands-on-Lab Particle Photon Weather Station in Azure*

<https://www.hackster.io/pjdecarlo/hands-on-lab-particle-photon-weather-station-in-azure-d89a03>

### **Future Work**

While the semester may be coming to a close, our enthusiasm for this project and our desire to see it fully functional and implemented has no end in sight. We plan to continue working on improving our system over the summer and thereafter, not as part of an academic exercise but as our own project. Proposed improvements to several parts of our system are presented below.

#### *Computer Vision Algorithm*

Two key components make up the computer vision algorithm, namely human detections and tracking. Human detection is the most important part of the algorithm to improve, because once we can detect humans in all circumstances and avoid false detections, tracking becomes much more reliable. Performing better human detection is mainly contingent on having better (and much more) training data. We plan to retrain a Support Vector Machine for human detection on a much larger set of human images. This should significantly increase the ability of our algorithm to detect humans.

In order to increase the accuracy of tracking people, we must couple our method of using location with the ability to extract specific and unique visual features from each person detected. This will require better machine learning algorithms such that our ability to track a person without losing them will become state-of-the-art.

## **Professional Awareness**

As professional engineers we must understand the importance of accepting moral responsibility for any outcomes involving the design, and implementation of our system. Our knowledge of the system gives us the foreseeability of actions that the system will implement. The design method we use gives us causal contribution to any effects the system makes on the public. Since we accept moral responsibility for the actions of our system, we must exercise a standard of care. The standard of care is the set of methods that are accepted by competent engineers in our field. As engineers we agree to ensure the safety, health, welfare of consumers and general public.

This active security surveillance system will use a machine learning algorithm so that it can actively learn repeating patterns. As time passes the system will become smarter, and will become more accurate in the detecting, and tracking, of suspects when a crime is committed. This system has much room to grow and develop. This system will need to learn in order to adapt with the environment that it is detecting, and tracking. As engineers we will want to continue to learn about all concepts involved with the systems operation in order continue to optimize the system.

One concept in the IEEE Code of Ethics that is a major concern for this project is “to treat fairly all persons regardless of such factors as race, religion, gender, disability, age, or national origin”. The solution to this issue is to limit the information being stored in the database to information that does not discriminate any person. The system will automatically assign a person an ID # that will be transferred to the database, and will then track that person based on their identification number, RGB model color, and size. The use of the RGB color model will avoid the representation of individuals using discriminatory color, or racial profiling.

As we develop this active security surveillance system we must consider all ethical issues, and try to make responsible engineering decisions that will solve ethical problems being faced. As the implementation of the system expands to clients around the world, we will have to adapt methods of data collection, and profiling, in order to satisfy social needs.

## **Project Management**

### **Project Tasks**

The attached table details who worked on which part of the project, and in what percentages.

Please see the table attached to this document.

### **Self-Descriptions**

The following are short descriptions of what each student did for the project.

*The lack of mention of a subject should be interpreted as a lack of involvement with that subject.*

#### **Brahm Powell & Titus Lungu**

Brahm and Titus were the two team members responsible for the creation of the computer vision algorithm.

They each performed about half of the work required to complete the algorithm.

Brahm created the ability to identify humans in the videos while Titus used machine learning to track individuals between multiple frames in a video.

Both developed a video processing algorithm for integration with the client application.

#### **Nick White**

Nick White conceptualized the project's initial idea, except the specifics of the machine learning algorithm and the hardware.

Had absolutely no involvement in the creation of the machine learning algorithm.

Had absolutely no involvement in the creation of the hardware or firmware for the HelpKey.

Built the data model for the design, as well as the architecture.

Deployed the server application, the client application, and database to the cloud environment and maintained them.

Developed the integration between the data sent from the hardware and the database server.

Developed the client-side interface, including the map, video feed (but not the video in it), and officer-application interface.

Developed the server code for retrieving data from the database and sending it to the client.

Developed the data model, and database server's functionality.

Authored the architecture and implementation sections of each paper.

### **Andrew Fisher**

Andrew's contributions to the product include:

Writing professional awareness for final project proposal in December.

Wrote hardware section in Final paper.

Worked on 2-3 slides on final presentation for hardware section.

Completed facial recognition using openCV and emguCV using the RGB approach.

Completed research on particle platform.

Performed wifi signal tests on the Particle photon.

Researched gps basics to operate gps module, picked hardware for device, drafted bill of materials for hardware, assembled helpkey prototypes (soldering), worked with mark on helpkey firmware, created webhooks for helpkey devices for particle and azure integration, participated in discussions on system architecture, and participated in project use case discussion.

### **Mark Heller**

My main contribution this semester was with the help key firmware and hardware, specifically the Electron, and some contributions to presentations.

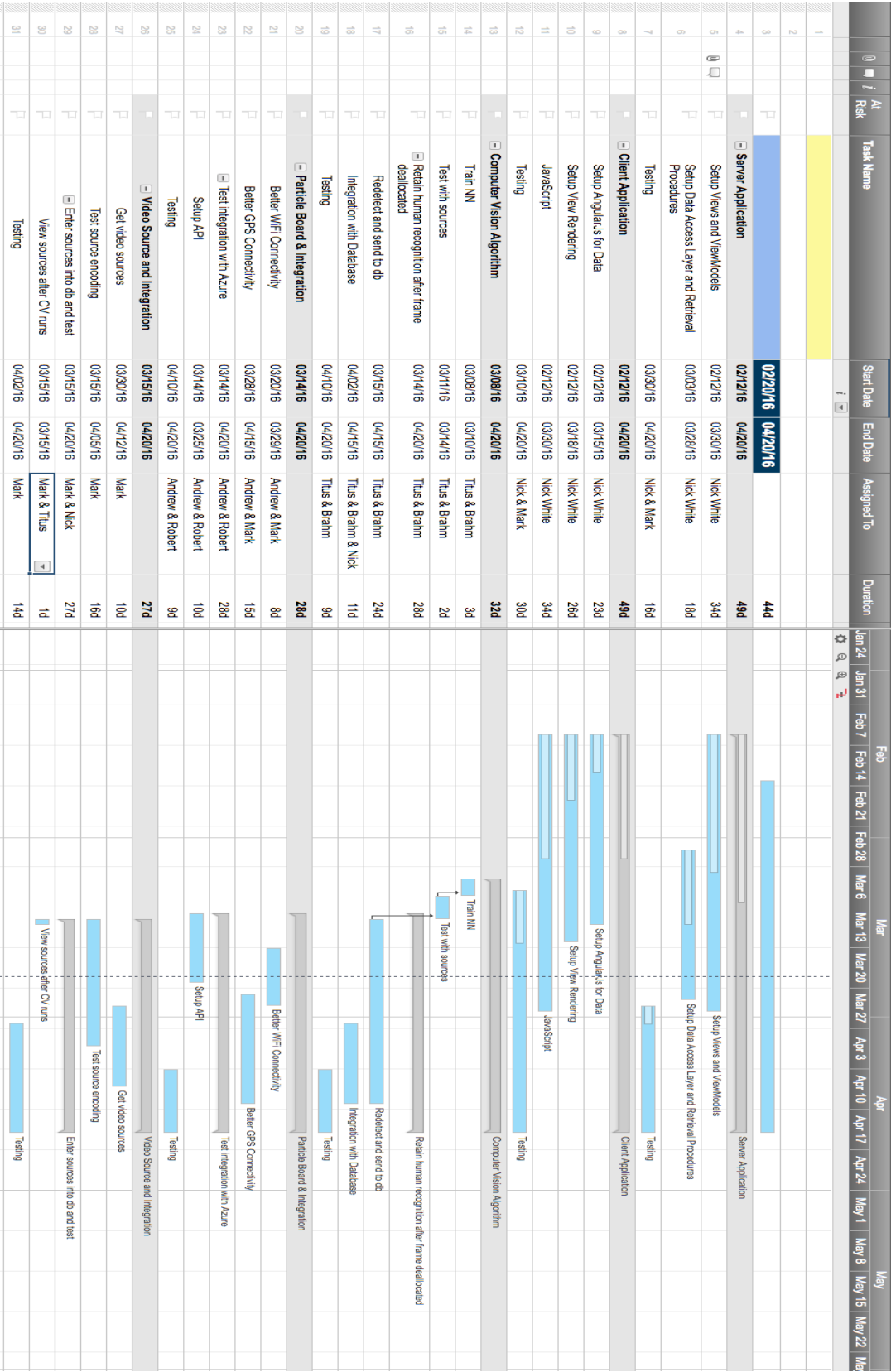
Worked on presentations for each presentation, was involved in the design process.

Was instrumental in designing and helping sort through ideas for the project (written by Nick White).

### **Robert Marshall**

Robert Marshall

Updated Gantt Chart



## References

- [1] Securitysales.com, 'Report: Video Surveillance Market to Reach \$42B by 2019', 2015. [Online]. Available: [http://www.securitysales.com/article/report\\_video\\_surveillance\\_market\\_to\\_reach\\_42b\\_by\\_2019](http://www.securitysales.com/article/report_video_surveillance_market_to_reach_42b_by_2019). [Accessed: 10- Dec- 2015].
- [2] N. Jenkins, '245 million video surveillance cameras installed globally in 2014', IHS Technology, 2015. [Online]. Available: <https://technology.ihs.com/532501/245-million-video-surveillance-cameras-installed-globally-in-2014>. [Accessed: 10- Dec- 2015].
- [3] Office of Postsecondary Education, 'Criminal Offenses - Public Property (Reporting Year: 2014)', 2015.
- [4] J. Castellano, 'Push For Campus Safety Means More Guns, Officers, Security Spending', Forbes, 2015. [Online]. Available: <http://www.forbes.com/sites/jillcastellano/2015/06/11/push-for-campus-safety-means-more-guns-officers-security-spending/>. [Accessed: 10- Dec- 2015].
- [5] [www.opencv.org](http://www.opencv.org)
- [6] [www.emgu.com](http://www.emgu.com)

## **Appendix A:** **Relevant Portions of Fall Semester Report**

### **Design Objectives**

The objective of the proposed project is to create a system that resembles the concurrent model of police work as much as possible while also lending itself to inevitable expansion into a model of proactive nature when the technology to do so matures. The major problems facing the current law enforcement model fall into two categories, crime detection/reporting, and crime resolution (apprehending suspects). Both of these problems, along with many other smaller issues with modern policing procedures<sup>8</sup> can be solved by implementing an active surveillance system.

### **Functional and Non-Functional Objectives**

The objectives of such a system break into two categories, each with sub-categories (as seen in Figure 0). The first category is that of functional objectives, or the objectives that will describe the desired behavior of the system. It is asserted here that the system must solve all of the issues described above. Systems which do not propose a viable solution to even one of the issues are not to be considered.

The second category is that of non-functional objectives, or those that elaborate on the performance of the system. These objectives are broken into categories such as Accessibility, Scalability, Response time, etc. (see footnote for full list)<sup>9</sup>.

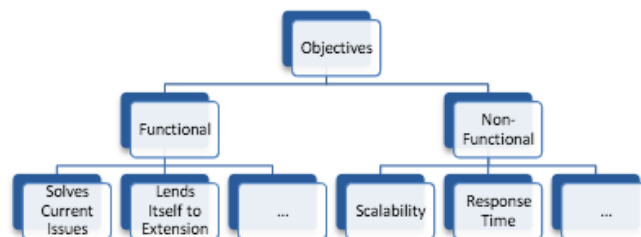


Figure 0.

### ***Functional Objectives***

---

<sup>8</sup> Other issues that can be solved include reviewing what took place at the scene of a crime in real-time, or shortly after, instead of relying on witness testimony to sort out a scene, and real-time reporting on which areas of campus are safe, busy, etc.

<sup>9</sup> Non-functional objective categories: Accessibility, Capacity, Compliance, Documentation, Disaster recovery, Efficiency, Effectiveness, Extensibility, Fault tolerance, Interoperability, Maintainability, Privacy, Portability, Quality, Reliability, Resilience, Response time, Robustness, Scalability, Security, Stability, Supportability, Testability

The functional objectives associated with this project are primarily concerned with covering all of the issues that currently exist with the policing model. To be considered complete, the system must perform a few fundamental tasks.

These objectives are to facilitate the following:

- A. User Interface
  - i. Live Video Feed
  - ii. Interactive Campus Map
  - iii. Region, zone, area selection and reporting menus
- B. Computer Vision Algorithm
  - i. New Human<sup>10</sup> Handler
  - ii. Existing Human<sup>11</sup> Handler
  - iii. Human Detection Handler
  - iv. Tracking Handler
- C. Data & Algorithm Persistence
  - i. Temporary Database
  - ii. Operational Database
  - iii. Data Warehouse

The objectives listed previously are those which are fundamental to the operation of the system. Having these objectives facilitated alone would provide a viable product. It is important to note, however, that expansions and elaborations on these foundational services and technologies are in queue, and are being designed.

### ***Non-Functional Objectives***

The nonfunctional objectives of this project are those which define *how well* the system must operate. These objectives are crucial since having a great design without having the logistics in place is simply a design, and not a viable product. The non-functional objectives are broken down into multiple categories, each of equal and utmost importance.

#### ***Efficiency***

Paramount to the implementation of such a system is its efficiency. Efficiency in this case is the performance of the system, as well as the stability, security, resilience, portability, reliability, and quality.

#### ***Performance***

The performance of the solution is of importance simply due to the real-time nature of the system. Performance in this regard is the time it takes for operations to execute. Updating the client applications, saving to the database, detecting civilians in a camera

---

<sup>10</sup> New Human - A person that is new to the system, meaning they have not been tracked before. Must be handled differently than existing users.

<sup>11</sup> Existing Human - A person that the system recognizes. They are updated in the system as opposed to being added as a new person.



feed, determining if they are part of a chase, etc. All of these actions take time, and with the goal of the application being real-time performance, careful attention must be paid to each component and time spent on optimizations. Optimizations in the system include using a fast, lightweight database management system, using cloud computing technologies to do the image processing, and many more.

### *Stability*

The stability of the system is crucial to its utility, or worth. For a system of this nature to be valuable, its downtime must be minimized. To do this, several redundancies are built into the architecture. These redundancies include having multiple application and algorithm servers and multiple database servers. By following the rule-of-three<sup>12</sup>, we can ensure complete stability and ensure that any failures are not catastrophic.

### *Security*

Access to campus and city security cameras is not a permission to be taken lightly. This sort of access requires high security measures to be taken. To facilitate this, an encrypted messaging format will be used for all client-server communication. No plain-text messages, or database calls will be made. Also, a white-listing<sup>13</sup> approach will be taken to user-authentication. With this, only registered law-enforcement officials will be able to access the system. Servers will be hosted and run from behind university or law enforcement grade firewalls, adding an additional level of security.

## **Technical Approach**

The technical approach to fulfilling these requirements, and completing the objectives we have set is a multi-part process. The first step in this approach will be to identify the unmet needs that currently exist in this industry, and in the realm of surveillance as a whole. The second step in our technical approach is to define the technical specifications of the system. The purpose of this step in the process is to break our specifications into two categories, the system's algorithm and the architecture. Lastly, the two remaining steps in the process concern themselves with enumerating, and lastly choosing a design concept from a list of viable concepts. By doing so, the best design can be chosen, while clearly explaining why other designs would not work.

### **Identifying the Unmet Needs**

The unmet needs this project seeks to fulfill are numerous. The first need we seek to fulfill is a need for a more robust and useful system than mere video surveillance. Currently, video is taken by cameras, and saved to a DVR system, only to be viewed when a crime has happened, *and is known about*. This type of system is obsolete, and considering

---

<sup>12</sup> The rule-of-three states that having three copies of any item ensures complete reliability since if one copy fails, it can be replaced by one of the existing copies. One copy is the main work-horse, while the other two exist for support and redundancy.

<sup>13</sup> White-listing: deny all access unless a user is on a white-list. Black-listing: allow all access unless a user is on a black-list.

how far technology has traveled since their deployment, an improvement is not only possible, but rather simple.

## **Defining Technical Specifications**

### ***Algorithm***

The core of our project is the system's ability to detect individuals, assign an ID to each person, and track that person across cameras (i.e. across a city). Once an individual is detected, the system can use his or her identifying features to assign the same ID to them when they go from one camera to another. In order to achieve these objectives, we will be using and enhancing two open source computer vision libraries: OpenCV and EmguCV. OpenCV is a library comprised of C++ functions for computer vision applications such as detecting color, shapes, motion, trajectories, and specific objects like humans, firearms, etc. EmguCV simply wraps OpenCV's functions in C# classes, which makes them easier to work with and implement in our application.

The initial detection stage in our system will consist of the algorithm identifying all humans individually in a video feed. Once a person is detected, they will be assigned an ID. Further, our algorithm will utilize OpenCV's detection capabilities to determine certain identifying features for each individual, which will be tagged to their ID for tracking. The main features that we will detect are RGB color patterns based on the individual's attire, the physical dimensions of the individual, their motion, their gait pattern, and suspicious objects such as firearms. Combined, all of these detection options allows for efficient tracking and unique identification of enough individuals to ensure there is no confusion in classifying a person in the video feed. Utilizing pre-coded and custom machine learning algorithms will allow us to intelligently detect and identify individuals across cameras based on the mentioned features, and ensure that a person can be tagged with the same ID across all cameras that he or she is seen on.

### ***RGB Color Pattern***

The RGB color pattern will be based on the color of clothing, skin tone, and hair color of a person. This combination of features are generally unique to each individual, but this is not sufficient for ensuring a unique set of identifying features for each person. In circumstances such as poor lighting (at night or on cloudy days) and in highly professional areas (where most people will be dressed in dark suits), more factors must be taken into account than the RGB pattern.

### ***Physical Dimensions***

The physical dimensions of an individual allows for increased robustness of our algorithm. Physical dimensions of a person can include one's height, width, and lengths of the head, torso, legs, and arms. Unlike with the RGB color pattern, identifying a person's physical size is independent of the lighting of the area. OpenCV can easily detect not only size but specific shapes as well, giving this method increased versatility.

### ***Motion and Gait***

An essential feature for identifying individuals in our system is their motion. The most efficient way to find and track an evading suspect likely involves identifying someone

who is sporadically running. Using OpenCV's ability to track motion in real time, we will identify the speed and direction of each person in the video feed. A person's direction of motion will allow us to predict on which cameras he is likely to appear after leaving the current camera's view. His speed will help us identify him with the same ID as was used on the previous camera, as well as alert us to possible suspicious activity.

A person's gait is also a key identifying factor related to their motion. Each person walks slightly differently and takes different size steps. Gait patterns will not only add a unique feature to each person's ID, but can also alert law enforcement of suspicious activity. For example, a person that is limping may have been injured, while someone who exhibits a sudden, aggressive change in motion may be committing a crime.

### *Suspicious Objects*

In an effort to not only detect crime, but prevent it as well, it is pertinent to identify key markers that indicate a crime is about to occur. While a person's motion and gait can help detect suspicious activity, they are not sufficient to predict a crime with high certainty. However, motion and gait, when coupled with object detection of weapons, greatly increases the certainty with which our system can predict a crime. Using OpenCV, we can detect objects such as firearms, suspicious bags, and other possible weapons.

An example of predicting a crime would be detecting a person would is running or moving sporadically, and is also carrying a firearm. While such behavior makes it obvious to the people in the surrounding area that the person in question may have some mal-intent, it generally takes some time to contact law enforcement to report it. However, when our system detects a situation similar to the one described, it will immediately alert law enforcement, allowing faster response time and even the prevention of certain crimes. Once a crime is observed and the suspect(s) identified, our system will use the tracking algorithm to report the location of the suspect(s) to law enforcement in real time.

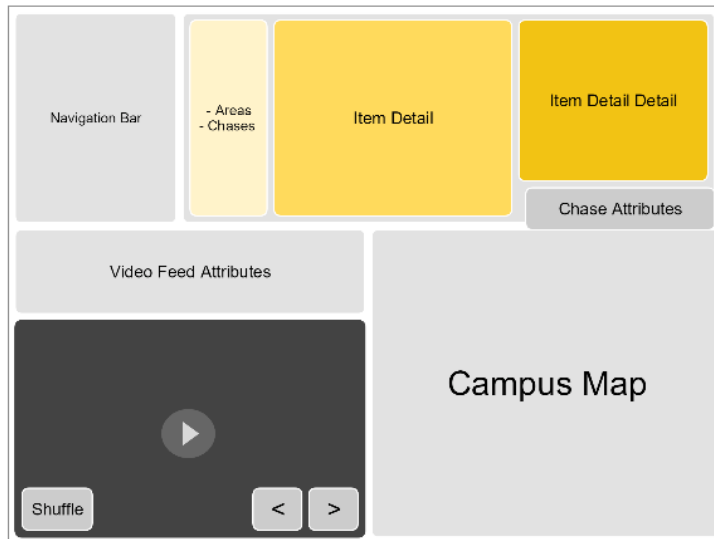
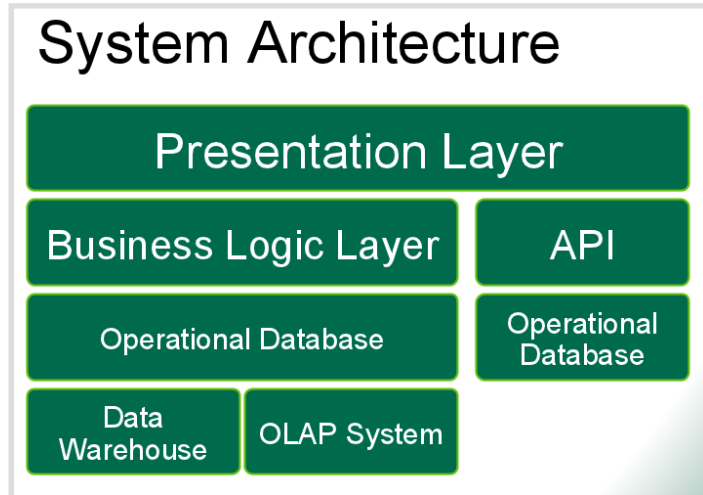
### *Architecture*

The architecture of the system follows a layered architecture design. In this way, the design can be broken up into independent layers, each operating and performing without knowledge of the other layers directly. This system involves designing a complex enterprise application that is composed of a large number of components across multiple levels of abstraction. Separation of concerns among components (for example, separating the user interface from the business logic, and separating the business logic from the database) increases flexibility, maintainability, and scalability.

The first and closest to the user in our layered design is the presentation layer (as seen in the figure on the next page).

### *Presentation Layer*

The presentation layer is the view from the perspective of the user. This layer provides the user interface to the system, and is one of the most important components. The presentation layer can be implemented on a mobile device, desktop, laptop, and even a web application without having to change the underlying logic. This layer will consist of a few major parts, including a campus map, real-time video feed, and views of the area that law enforcement can select and view. A cursory view of this is seen in the image below. In this system, this layer is realized by using Windows' WPF framework inside a .NET application.



### *Business Logic Layer*

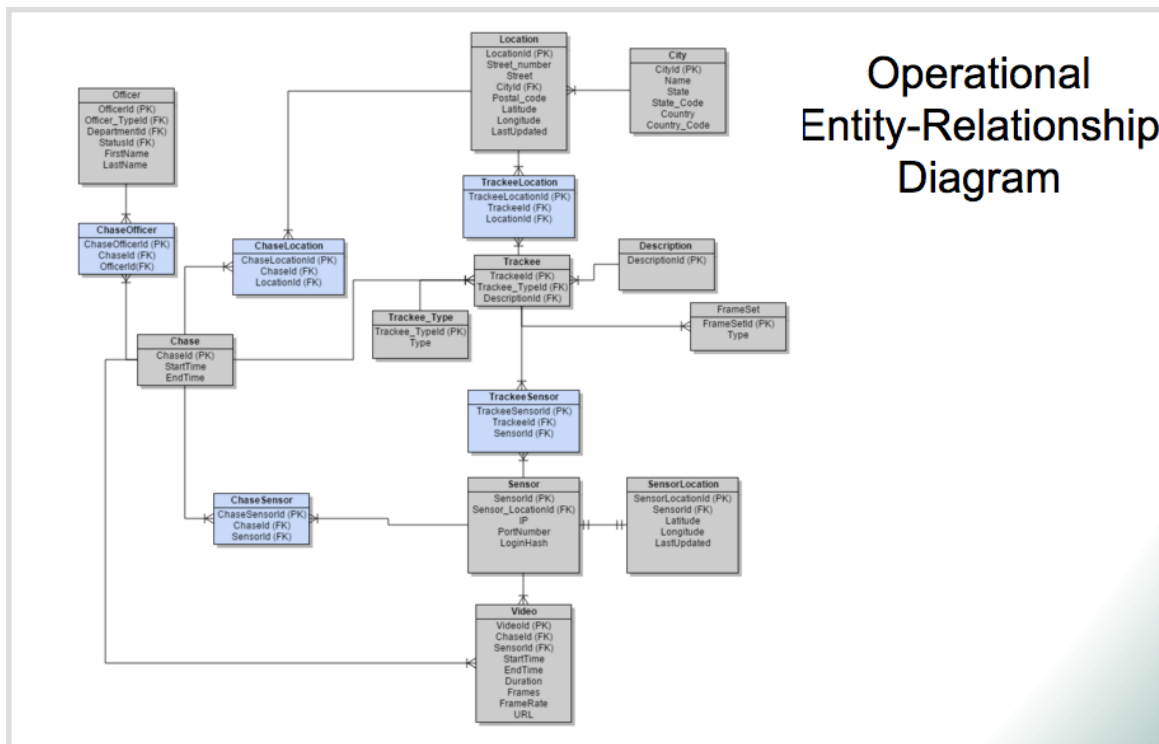
The business logic layer is where the algorithm, as well as data-access logic will reside. This layer concerns itself with the incoming video streams, and data from the database in order to analyze video and make decisions which are then persisted back into the database. This layer is the main entry point of the application, and the true heart of what makes ASaaS a truly innovative solution. In this

system, this layer is realized by using a .NET library as a wrapper to the OpenCV framework.

### *Database Layer*

The database layer of the application is where all of the decisions, video, and law enforcement input are stored. Information about locations, sensors, trackees, videos, cities, officers, and much more are persisted and accessed by the logic layer. The Entity Relationship diagram of the database can be found below in it's simplest form. In this

system, this layer is implemented using Microsoft SQL Server, and Microsoft's Entity



Framework for integration with the .NET application.

### Enumerating Design Concepts

With these objectives in mind, the utility and relevance of a design can be measured by placing it on the chart shown in figure 1, and comparing it to the other designs. Figure 1 shows a plot of a designs effectiveness, efficiency, and complexity. An optimal design would maximize effectiveness, and efficiency, while minimizing complexity (i.e. be a small dot in the upper right corner of the graph). However, designs that maximize effectiveness and efficiency often have higher complexity, so careful attention is being paid to all three of these details. One design we tested was that which is marker 1 in the graph. This design was simple, but

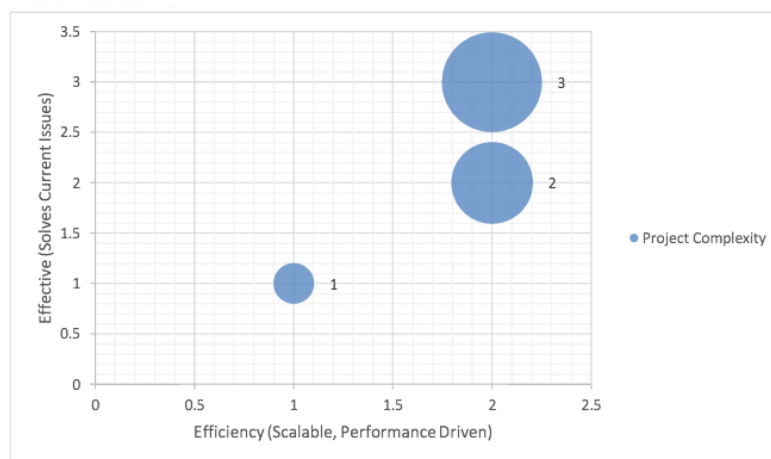


Figure 1

did not meet the efficiency and effectiveness requirements sought by the requirements of this project. The other two designs, 2 and 3, are a desktop application, and a web application respectively. The web application, although more effective, is a lot more complex. Due to the timeline of this project, it is more realistic and time effective to go with the desktop application approach.

### **Selecting Design Concept**

To select the best design, four criteria must not only be met. These criteria ensure the solution is both a good short-term solution by making sure all requirements are met, as well as a good long-term solution by ensuring that it can be applied in a larger setting.

#### *Scalable*

The first is that the system/solution must be scalable. If the system works solely on the campus, but not in a bigger setting, it is not a viable system. The system must be scalable, and so to facilitate this goal the architecture must be able to be deployed across a cluster of servers and not a single server.

#### *Cost Effective*

Solutions that are effective, efficient, not complex, but expensive are not to be considered. To maintain cost effectiveness, only open source and free software is used to implement the system. Any licensed software is used under a free license. In this way, the cost of implementing the solution is only the cost of spinning up a server or two, which is not significant compared to the cost of current surveillance systems.

#### *Secure*

Security is of utmost importance in this system. In this way, the security of the system in a design is a measure of design itself. Designs that place security as a first class citizen are the most important, and no matter the complexity, they should always be chosen over designs with holes in security.

#### *Flexible*

Flexibility of design is something which would allow the design to expand in the future, as stated previously, the optimal design is one which is proactive. To facilitate this goal, further developments in tracking algorithms must be made in addition to the ones proposed in this system. Flexibility in this sense would mean designing the system to be adapted from the very start, employing software engineering tactics such as refactoring, test driven development, and agile development methodologies.

The design that meets all of these criteria is the ASaaS design that is implemented entirely on the Microsoft stack. This stack contains a WPF and .NET front-end, .NET logic

library for the business logic, and the database backend is made up of a SQL Server cluster, and SQL databases.

### **Market Potential**

Video surveillance is one of the fastest growing markets in modern security. In 2013 the video surveillance market had an estimated value of \$14.98 billion. Current projections estimate it will have a value of \$48.32 billion in the year 2020 – a 400% increase from 2013 [1]. Video surveillance as a service (VSaaS) represents 13 percent of this market with a current estimated value of \$1.94 billion. Assuming the market distribution remains consistent, VSaaS will have a market value of \$6.28 billion by 2020. This increase in market size prompts a direct correlation wherein the number of video surveillance products increases simultaneously. In 2014 there were an estimated 245 million security cameras worldwide. Of these 245 million it is estimated that 20 percent are network cameras [2]. The majority of these cameras are designed to identify suspects and aid in solving violent and nonviolent crimes.

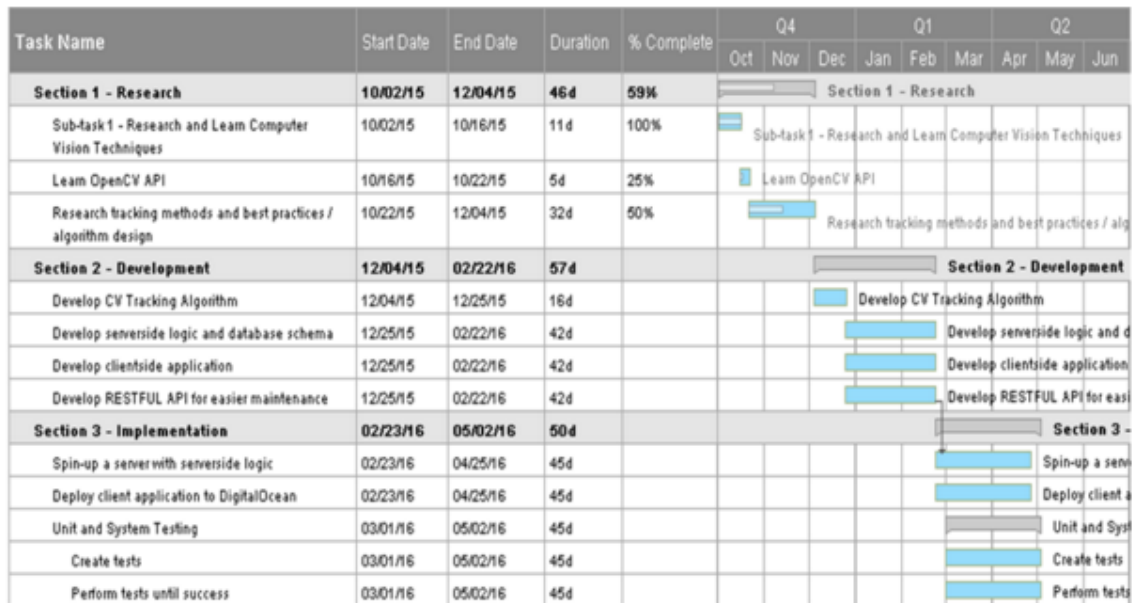
According to The Campus Safety and Security Data Analysis Cutting Tool, there were a total of 530 robberies on four-year public college campuses in 2014 [3]. This averages out to 1.45 robberies per day. To combat crime rates, colleges nationwide are increasing public safety budgets. As of 2015, Temple University in Philadelphia has a \$19.3 million dollar public safety budget, representing a 45 percent increase since 2010 [4]. Colleges – especially urban colleges – are constantly looking for ways to improve student safety and are increasing security budgets to meet this need. Additionally, these campuses are already in possession of network camera systems. College campus police forces in the United States have been identified as the target audience for this service. ASaaS can be offered to these organizations as an effective method of reducing crime and improving campus safety.

Though campus police forces have been identified as the primary target market, the ASaaS system can theoretically be implemented across any network camera infrastructure. It is estimated the system will be implemented across 10 million security cameras in the United States. The cost of implementing the system on a camera is \$10 per month, including processing, hosting, and maintenance costs. It is estimated the ASaaS system would bring in an estimated revenue of \$100 million per month, or \$1.2 billion per year.

### **Project Management**

Our team consists of two students from each of three engineering departments at CSU: mechanical, computer, and electrical. We are managing our team and our individual responsibilities using Visual Studio Online, where we assign tasks and deadlines for the different aspects of the project. Visual Studio is also our current repository for all of the code that will make up our system, including the main tracking algorithm, the WPF application, and the database structure. This allows for seamless version control and as

well as an agile development environment for our product. Below is a Gantt chart representation of our timeline and the breakdown of our project.



## Deliverables

Our final product will have four major deliverables that will make up our system. First, we will provide a WPF application as the presentation layer of our project that allows intuitive user interaction with the tracking system by viewing analyzed video feeds and suspect location on an interactive map, all in real-time. Second, the logic layer of our system will consist of the actual tracking functionality, allowing our system to analyze video feeds as they are being recorded, for efficient and effective suspect tracking. Third, our system will have an operational database, giving it the ability to persist and make intelligent decisions based on stored data and video analysis. Fourth, our system will include a data warehouse, where we will store all of our video feeds and analyses in order to perform exploratory analysis using big data analytics. This will give us the ability to generate reports on crime trends, locations, and other pertinent information using saved historical data.

## Budget

The budget for this project is minimal due to the resources that we are utilizing for the design and implementation of this service. We have access to our own physical server that will be used for all of the cloud processing needed from data transmission, and storage. Since we have access to our own server we will not need to invest money into the cloud processing sector of this project. Currently we are using our webcams built into our own devices in order to develop the algorithm for our service. When we begin to implement our developed service, we plan to use pre-existing cameras that are used by Cleveland State



University. The use of pre-existing cameras gives us the ability to implement our service without spending cash on hardware. This concept will also benefit our service in the future when we begin to implement our service for various applications. For all of our code generation, and software development, we do not have any projected costs as well. For code generation we are using Visual Studio Online which is free to students using a DreamSpark account given to students by Cleveland State University. Using Visual Studio we will develop our algorithm, and we will develop our web application using C# language. The use of Microsoft tools we will be able to optimize our product at no cost using resources given to us as students.

### **Communication and Coordination with Sponsor**

The team has communicated with the industry sponsor on a regular basis over the fall semester via email, phone, and several one-on-one meetings. We plan to continue to meet with our sponsor on campus at least on a monthly basis in order to discuss our progress and to ensure that the project is taking the appropriate direction, as desired by the sponsoring corporation. We will also send regular emails to the industry sponsor to establish meetings, report on our progress, ask for any clarifications that may be pertinent to the success of this project.

### **Team Qualifications**

Our team is well-suited for this project due to our experience with solving real-world engineering problems and our cumulative experience using different coding languages and designing software. Members of our design have had relevant internships at engineering companies, participated in research at the university level, have designed numerous software solutions for corporations, and have designed and implemented different machine learning techniques. We have also had extensive experience collaborating on teams and communicating project goals and progress to advisors/employers.

## **Appendix B:** **Market Analysis and Business Strategies for Full Scale Implementation of ASaaS**

### Active Surveillance as a Service, Market Analysis (Intelligent Visual Surveillance Software)

#### **1. Introduction**

The market being analyzed is a newly emerging market that has been deemed “Active Surveillance as a Service”. This market is composed of intelligent visual surveillance software that is able to detect and track people in live security camera video feeds. The motivation for analyzing this market is that we believe certain technological innovations of recent years will allow such a system to be very scalable. The goal is to implement a connected and comprehensive software system in major cities throughout the world for the purpose of monitoring and even preventing crime, mainly for law enforcement to use.

Our proposed system is a software system in the form of an app, to be run either on a laptop or tablet device, first and foremost in police cruisers, but also in police departments and city surveillance centers. This software provides the client side front-end with which the user can interface with a much larger system, which will run on a cloud platform. This proposed system connects thousands of IP cameras throughout an urban city, streams the live videos, and identifies all humans in real-time. In addition, every person is uniquely identified and tracked such that a police officer can get real-time location of a suspect in a recent crime.

The system does not attempt to find the actual identity of any of the people it is tracking, but instead tracks people based on their unique visual features and movements in space. This allows anonymity of everyone being tracked while also allowing law enforcement to identify a criminal from a video feed and then receive their real-time location and motion on a map interface in the app. The goal is also for the system to be able to identify crimes as they are occurring and even predict them based on known patterns and object recognition. These potential threats would be made known to the officers nearest to the location of the threat via a notification system on their device (similar to how Uber notifies the closest available drivers when a person requests a ride). Officers would be able to confirm or deny the legitimacy of the threat by viewing the video feed on their device, or can tag an event they witnessed as being a threat and initiate a chase of a suspect. In either case, the system will continue to learn based on this user feedback and become increasingly intelligent and capable of predicting crimes. This is similar to how the Waze app uses its large user base to confirm traffic incidents and other irregularities on the road and so learn how to better route all Waze users in the area.

#### **2. Analysis**

##### *Market Description*

The product boundary of this market includes any software system that detects and tracks objects of interest in live video feeds. The product boundary is only concerned with the processing of the video and outputting useful location data for law enforcement. The

boundary does not include the physical hardware used to do this (cameras, data processing centers, police computers, response centers, etc.). All of these pieces of hardware are already in mainstream use. The power of this software will be that it can connect all these pre-existing devices to achieve safer cities.

The geographical boundary is international as the software could be seamlessly implemented throughout the world. The sellers in this market are surveillance companies that produce software similar to that which we are proposing. The buyers are entities such as cities, schools, private companies, and residential customers. Currently, the size of this market is still small, with only a few sellers and buyers.

There seems to be no intermediary parts of this market structure separating the surveillance companies from the buyers. Virtually all of the currently existing companies sell their software packaged with hardware (cameras, motion sensors, etc.). These companies sell directly to their customers. Our proposed product is entirely software and would have all the compatibility necessary to connect to pre-existing cameras via standardized internet identification and communication methods, namely IP (Internet Protocol).

The major customers that we wish to attract are urban cities, where we could implement our system across thousands of pre-existing cameras. The major obstacle here is acquiring access to cameras that are not owned by a city but rather by private entities. Such access is necessary in order to have a broad enough video coverage of city streets for our system to be most effective. Two principal underpinnings of this obstacle are legal boundaries and public sentiment towards heavy surveillance. Alternatively, we could start implementation of our system on a much smaller scale, selling to universities, private companies, and government sectors and buildings. Our concern with this is that there may be certain economies of scale required for implementation of this system in order to be profitable.

The market worth is about \$1.2 billion annually for the video analytics market, which is the closest comparison to our market. Some of the current sellers in this market include companies such as NEC, Envera Systems, and Super Circuits. These companies' market focus range from selling to small private entities as well as government and law enforcement agencies.

### *Competition*

NEC is a publicly traded, international, multi-billion dollar technology and IT company with a state-of-the-art biometrics department. Its yearly technology revenue is upwards of \$30 billion. Within the biometrics department, NEC owns a proprietary software known as NeoFace, which finds the identity of people passing by a camera based on facial recognition. NeoFace has a similar architecture to our proposed system in that it is implemented live on customers' pre-existing IP cameras. In contrast to our system, however, NEC sells computer servers required to connect to the IP cameras and run the software. NeoFace only requires minimal resolution (as little as twelve pixels between the eyes) to identify an individual, making it very versatile and useful even with older, lower quality security cameras. At a recent NIST (National Institute of Standards and Technology) competition, NeoFace greatly outperformed its competition, with eight times fewer errors than the next highest performing algorithm.

While NEC sells this software to some private corporations, its main market focus is law enforcement and government agencies, such as homeland security and defense agencies. The software is generally implemented in cameras positioned in high security places in order to keep track of who enters and identify any possible intrusions or threats. However, the software does not seem to be implemented over many overlapping cameras throughout a vast area, nor does it feature any sort of tracking capability. This is a major difference between NeoFace and our proposed system, as NeoFace is meant to be implemented at specific points of interest while our system would be implemented throughout a larger area.

Super Circuits is a security company that sells cameras, sensors, and tracking software to cities for law enforcement to use. As with NEC, Super Circuits' systems are generally meant for implementation in specific locations in a city, not widespread throughout the entire city. The software is installed in every single camera, which is sold for as much as \$800 a piece, which is up to ten times as expensive as buying the same camera without the software. While Super Circuits' system can perform human detection in real-time, it does not seem to be able to track people.

Envera is a smaller, private company which focuses solely on residential security systems. It sells both the hardware and software for their system, which includes security cameras, human detection software, and an audible system for alerting intruders to leave the premises. On occasion, Envera may implement its software into pre-existing cameras, if they are compatible. Envera's system is very specifically tailored to meet the needs of residential customers and is therefore unlikely to be scalable to the extent which our system aims towards. Similarly to NEC, Envera does not track people and is generally implemented in only a few key locations at a customer's place of residence.

Important to note is that Envera's systems are installed, serviced, and monitored by company personnel. A very clear distinction is drawn here between Envera and our system. First, our system requires no real "installation". Using already existing IP cameras in an area, we only need to be given permission to connect to those cameras over the internet. Once this is done, the system is fully operational on our cloud-based environment. "Servicing" is similarly negligible in our system, as any updates to the software would occur immediately and wirelessly. Finally, and most importantly, is the act of "monitoring". While many existing systems may boast a large department of specialists dedicated to servicing their system, our system is so simple and elegant that it puts the power of operation in the hands of the customer. Detections and alerts occur automatically via a notification system. Actions can be performed by the user with a simple click, all within the nowadays familiar app environment. This provides simplicity for the customer and virtually no need to interact with the seller after the initial, and speedy, set up.

Local governments such as London and Chicago also have tailored security systems that can track vehicles and perform facial recognition. NeoFace is thought to be used by Chicago's police department and has even been rumored to be responsible for apprehension of suspects in recent criminal cases, both in Chicago and Arizona. It is important to note that these apprehensions were made quite some time after the crimes were committed (a year later, in one case), by processing security footage from the crime scene with the software's algorithm. Our system, in contrast, aims to track criminals in real-time during and after a crime is committed for much quicker apprehension of suspects.

Implementations of such systems by local governments rely heavily on specific hardware and large operation control centers. They do not offer the flexibility or scalability that a third party “plug-and-play” software could provide. Additionally, third party software can share information seamlessly throughout its database across city boundaries in emergencies such as terrorist attacks. The recent attacks in Paris are a prime example. Had such a system been in place in major French and Belgium cities, the terrorists would likely have been apprehended relatively soon, preventing the second tragic attack in Brussels just a few months later.

### *Technological Innovations*

The major force driving changes in the surveillance market as a whole is technological innovation. More and more customers are making the switch from analogue cameras to IP cameras. IP cameras stream their video feeds directly online via standardized methods and make it simple for a third party to gain access, with the permission of the camera owners. IP cameras also provide much better quality color video, as opposed to pixelated black and white video. In 2014, it was estimated that IP camera usage will increase by a compound annual growth rate of 13.7% through 2017.

This connection of cameras to the internet makes our proposed system very feasible as it ensures that we can use already existing cameras that simply connect to our service online. This takes away the cost and complexity of installing new hardware in order to get a specific type of software, which is currently required for most companies that sell security software. This will no longer be necessary as IP cameras become more and more commonplace.

This move to devices connected over the internet that feed into certain services is referred to as the Internet of Things (IoT). IoT is becoming the “next big thing” in data collection because it allows endless devices to share their data quickly and seamlessly. This then allows a person to interact with and analyze all the data in a central location, even if they are not at the scene of the data collection. IoT is starting to have a large market in personal and home devices such as Fitbits, Google Nest, and even small, home security systems. However, the potential of IoT in the overall surveillance market in cities, large corporations, etc. has been largely unexploited, though it promises to be a very large market.

Technological novelties such as IP cameras and IoT systems are only as good as how their output data is handled. That is where the cloud comes in. The cloud allows all such devices to stream their data straight to a centralized location that multiple persons can access from anywhere in the world. Perhaps even more important than storing the data is processing the data to get useful information. With cloud computing services such as the Amazon cloud and Microsoft Azure, a user can upscale or downscale their computational requirements as needed. You can go from processing one video feed on the cloud with the equivalent of a laptop’s computing power to processing a thousand video feeds with the equivalent computational resources of a supercomputer. This allows unprecedented scalability of services and also takes away the need for the user of the data to have to manage the computing resources.

The security market as a whole is also shifting more towards a service model with the birth of Video Surveillance as a Service (VSaaS). Though different than our proposed

market (Active Surveillance as a Market), this move from a one-time-purchase model to a continuous service model shows the demand for systems that can handle needs such as data analytics, wireless video streaming, etc., which is the exact focus of our system.

The cost of such technologies is also rapidly decreasing. The price of IP cameras fell by 18 percent in 2015, driven by the high competition in the hardware market for surveillance. While this does not directly affect our product, it does increase its feasibility by allowing more institutions to be able to afford the hardware needed to feed directly into our system.

### *Government Involvement*

In contrast with the decreasing price of technology, the emphasis on bigger, better, and smarter security is increasing. Amid recent escalations in acts of terror and crime, more companies are investing heavily in better security systems and stronger IT departments. The value of the video surveillance market (including both hardware and software) in 2013 was \$15 billion and is projected to be \$50 billion in 2020. While the only direct affect this has on our system is the fact that it ensures larger integration of required hardware such as IP cameras, it does have one crucial implication. This large increase in the desire to invest in higher security shows that there are opportunities for smarter surveillance software as well, especially if this software comes with no additional hardware requirements.

This also opens the doors for government implementation, encouragement, and support of such systems. The effects of this can already be seen with the investment of hundreds of millions of dollars of government money, yearly, in support of state initiatives that align with the goals of the US Department of Homeland Security. The ESA (Economics and Statistics Administration) is also funding security upgrades for schools. A School Security Caucus of House members has even been created to address student safety and technology upgrades.

In terms of city finances, US cities spend millions of dollars annually on settlements for police lawsuits and other crime related incidents. In 2016, Cleveland paid out \$6 million to the family of Tamir Rice for the police shooting of the boy. Chicago has spent almost \$500 million from 2004 to 2014 on settlements. In 2011, Los Angeles paid out \$54 million and New York paid out \$735 million. Even cities which are considered safer, such as Denver, spend over \$1 million annually on lawsuits.

Such lawsuits are costly, but not nearly as costly as the total economic affect that crime has on the United States. In his 1999 study, *The Aggregate Burden of Crime*, economist David Anderson estimated that the cost of crime in the US alone amounts to well over \$1 *trillion* annually. This figure includes the cost of insurance fraud, theft, and other direct effects of theft (over half a trillion dollars annually) as well as opportunity costs for individuals, businesses, and cities and an estimate of the value of risks to life and health. While the latter is subjective, it can certainly become an objective metric. If people are safer and therefore healthier and happier, they may work more or at least more efficiently, spend more money on entertainment, and have more energy for ventures such as starting business ventures. These figures provide enormous possible savings for cities which implement an intelligent system to increase security and suppress crime.

### *Public Opinion*

The overall sentiment of people towards increased surveillance is also shifting from the skeptical, libertarian view to a view that is prepared to surrender a certain degree of privacy for an increased sense of security. Though the general populace plays no direct role in the market structure here, it does play a role in the overall scheme of the marketplace. For one, if people are more willing to accept increased surveillance, companies, schools, and other entities will be less hesitant to implement such measures, not having to fear public scrutiny.

Another point of relevance concerning the public's relationship to such a system is the fact that it would not alter the degree to which the public is exposed much from what is currently the state. Every time we interact with social media, we not only make our private lives public, but also expose very personal information such as our location and time. Most social media even have a feature that allows people within a certain distance "see" each other, if they choose to turn on that option. Credit card companies make the most of our purchase history, location, and buying patterns. In reality, any person that is even slightly connected in the 21<sup>st</sup> century has essentially relinquished a great deal of their privacy. And at the same time, none of the mentioned examples increase our security, just our standards of living. A system such as ours will not greatly alter our state of privacy from the current one, but will greatly increase our state of security.

In fact, we would argue that this system can actually increase the privacy of the general public. Replacing human security guards, mindlessly monitoring video screens, with a piece of software means that now the software is evaluating what it sees and only sends information to the eyes of a human when it sees a possible threat. This is a significant reduction in human intervention in this system. Human security personnel will no longer have to be watching us all of the time, but instead only when a threat is imminent – at which point, police should be watching.

### **3. Recommendations**

Due to the recent technological innovations described above as well as the desire of companies, cities, and the general populace to increase security, it seems that our proposed system would provide a much needed service for its customers. With a model that focuses solely on software and cloud computing, the Active Surveillance as a Service market seems to make use of the newest trends and opportunities in the larger overall security and surveillance market.

Since the major driving factor of change is technological innovation, it is crucial for our system to provide state-of-the-art features and be driven by the best algorithms for surveillance. Such a system would be a completely new, technologically innovated way to handle security. If the system is intelligent, fast, and easy to use, it can creatively destroy the current infrastructure that exists for security systems. However, if the system has glitches, is not intuitive to use, and does not offer high performance on a consistent basis, it will very quickly join a tech graveyard with a vast array of other creative but poorly executed technologies. To prevent this from occurring, three major components of our system must be the core of our focus: artificial intelligence, database operations, and cybersecurity.

Artificial intelligence (AI) is the centerpiece of our system. It allows the software to identify humans in a security video, track each person across multiple cameras, and determine security threats based on common actions, circumstances, and objects. AI is the new “tech bubble”, as the Internet-bubble once was. Institutions across the world, from universities such as Stanford to companies such as Google, are investing enormous resources into this rapidly growing field. AI will (and in fact, has already begun to) change the way virtually all technologies function, and security is no exception. For this reason, it is crucial for us to develop close ties to the leaders in both industry and academia with the expertise in this field and to ensure that our algorithms and implementations are on the cutting edge.

While AI provides the core functionality for our system, all the generated data must be efficiently stored and quickly accessible. This is why the database for our system is so important. Even with a cloud computing infrastructure, all of the captured video feeds as well as the data outputted by our system requires a huge amount of storage. Size is not all, however, as such a large database also has to be accessed at the speed of light to ensure that our system can consistently perform in real-time with no lag.

It is important for us to address cybersecurity concerns as well. There are many encryption methods that exist to safeguard such a system from hackers and our system must be equipped with the best of these. Everything from online banking to social media to credit card transactions present a huge liability to the consumer if that data were to be accessed by unauthorized persons with mal-intent. Realistically, our system is in fact less “dangerous” than these other digital transactions as it does not reveal the identity of any person being tracked, nor their place of employment, relatives, etc. While it is possible to acquire some of this data based on where every person went throughout their day, it is far less exposing than social media sites. Nevertheless, offering security at the same or higher level as banks and other mentioned parties is a top priority.

Finding the right people to work with and hire is no less important than offering the best technology. It is important for us to create an open-minded, dynamic atmosphere in the design of our system such that we can attract creative individuals to work with us. It is important to find people that are better than we are in their own respective niche in order to ensure that every bit of our system is the best it can be. Such individuals can be found at top companies and universities that specialize in related technologies and research, or during a casual encounter at a café. The more “out-of-the-box” a person can think, the better.

As the system progresses, new people are hired, and technologies and the world change, one important point is to remain open to change and new ideas. Many companies fail because they cannot adapt and keep up with their competition or their customers’ desires. It is important to not be so fixed on one idea or notion of how things should be done that we become unwilling to change and progress. Innovation and change are some of the key driving factors in the software industry and those that master these aspects generally control the marketplace. At the same time, a new design must also be sustainable and flawless before mass distribution, so there is certainly a degree of acumen required when making such decisions.

Of course, in order for our system to be useful, we must have customers, which means that we must have people that support what we are doing. While support from government and law enforcement plays a large role here, public opinion may play an even



bigger one. Public opinion drives much of business and government decisions. Therefore, we propose that it is just as necessary to market our system to the general public as it is to market it to police departments. In order to achieve mass adoption of our system at an affordable price to the customer and an attractive profit for us, it would be invaluable to have the support of the general public. Considering the recent tidal wave of lawsuits and riots involving police actions, implementing measures seen as beneficial by the public could go a long way towards fixing the relationship of police departments with their citizens.

To this end, we aim to raise general awareness of our system, how it works, and why it is desirable to have in one's city. Key points that should be addressed are an increase in security, an increase in privacy (as described in the Analysis section), and a decrease in racial profiling. The latter benefit is of particular importance to the public nowadays and is a direct benefit of a system which identifies suspects based on objective metrics, regardless of their demographics. This approach of targeting the "customer of the customer" is analogous to Intel's "Intel Inside" campaign. While the majority of Intel's sales are to large computer manufacturers such as Dell, Microsoft, Apple, etc., this initiative to convince the end consumer that Intel is the best processor caused computer manufacturers to prefer using Intel in their products in order to ensure that people would want to buy their computer since it had the "superior" processor. Intel's processors are not necessarily superior to others, such as AMD, but they were able to capture the majority of the market with this technique of targeting the end user. We hope to develop a similar campaign which will cause people all over to ask themselves why we do not have better security systems in urban settings, and lead them to desire having our system in place. This in turn may put pressure on police departments and cities to purchase our system, if the public's demand of increased, intelligent surveillance is strong enough.

Finally, we will need to raise appropriate funds in order to create a fully functioning system and acquire an initial customer base. For a software system such as ours, venture capitalist funding is the preferred method of financing. The investors must be prepared to invest for the long term, as it would likely require several years before the system is fully functional and initial customers are established. California seems to be the best place to implement our recommendations as it gives us the most resources in terms of investors and top tech people with which to work.

#### **4. Feasibility of Implementation**

Implementation of our system would occur on a much larger scale than what any company currently offers. While our system may be able to acquire a large part of the available market, it may only slightly affect the financial statements of our competitors. For example, our system is unlikely to greatly affect NEC's biometric department as they focus on facial recognition at locations of interest within government facilities, while our system tracks people without revealing specific identities. As our system grows and develops, however, there may be opportunities to spread into neighboring markets, such as biometrics using video analysis. Envera Systems and Super Circuits may be more affected since they also offer full-person detection, though their systems are meant for location-of-interest implementation as well. Customers interested in small area implementation such as that may not desire a more widespread implementation such as ours, nor be willing to

pay for it. Our system could, however, be implemented on a small scale and provide tracking of people, which could take customers from the aforementioned companies. At the same time, depending on the economies of scale involved with implementation of our system, it may not be efficient for us to implement it in small areas (i.e.: only for a few cameras in an area).

The operating statement of NEC for EOY 2015 is presented on the following page. While NEC does not provide quite the same system as the one we are proposing and also has many other operations across the globe, it is the closest comparison and gives a sense of the kind of resources and size that competitors have.

In order to estimate the financial trajectory of the first five years of developing our system, a pro forma is also shown on the following pages. We estimate that the expenses of the first five years will be covered by VC funding and revenue may not be generated until year six due to the amount of time required to perfect our system and get it to market. With over 210 million security cameras throughout the world, we estimate being able to implement our system across at least 10% of these cameras within the first ten years after our first sale. With an estimated price of \$10 per camera per month, this amounts to a yearly revenue of \$2.52 billion.

#### NEC Operating Statement EOY 2015

	Millions of U.S. dollars 2015	Percent change 2015/2014
Net sales .....	\$24,463	-3.5%
Overseas sales .....	4,890	3.1
Percentage of international sales to consolidated net sales (%) .....		
Operating income .....	1,067	20.6
Ordinary income .....	934	62.1
Net income (loss) .....	478	69.8
Cash flows from operating activities .....	733	-6.6
Cash flows from investing activities .....	(396)	—
Free cash flows .....	337	-26.8
R&D expenses .....	1,118	-6.0
Capital expenditures (property, plant and equipment).....	312	-62.1
Depreciation (property, plant and equipment).....	404	7.4
Per share data (in yen and U.S. dollars):		
Net income (loss) .....	0.18	69.7
Cash dividends .....	0.03	0.0
Total assets .....	21,839	4.6
Owner's equity .....	6,864	18.3
Return on equity (%) .....		
Owner's equity ratio (%) .....		
Interest-bearing debt .....	4,340	-9.5

Pro Forma for *Active Surveillance as a Service System* (in USD)

VC Funding, per year (EOY 1-5)				
Line Item	Cost per Unit	#	Unit	Total
Founder Salary	\$ 75,000.00	6	Person	\$ 450,000.00
Talent Acquisition Team Member	\$ 75,000.00	1	Person	\$ 75,000.00
Additional Employees	\$ 120,000.00	5	Person	\$ 600,000.00
Office Rent	\$ 5,000.00	12	Month	\$ 60,000.00
Cloud Computing	\$ 150,000.00	1	Year	\$ 150,000.00
Tech Supplies	\$ 20,000.00	1	Year	\$ 20,000.00
Travel	\$ 10,000.00	1	Year	\$ 10,000.00
Conferences	\$ 15,000.00	1	Year	\$ 15,000.00
Lobbying/Meetings	\$ 50,000.00	1	Year	\$ 50,000.00
Marketing	\$ 25,000.00	1	Year	\$ 25,000.00
			Total (1 year)	\$ 1,455,000.00
			Total (5 years)	\$ 7,275,000.00
Operating Statement, (EOY 6)				
Line Item	Income per Unit	#	Unit	Total
Revenue per camera per month	\$ 10.00	10,000	Camera	\$ 100,000.00
Number of cities		3	City	\$ 300,000.00
			EOY Total	\$ 3,600,000.00
			Profit/(Loss)	\$ 2,145,000.00
Operating Statement, (EOY 7)				
Line Item	Income per Unit	#	Unit	Total
Revenue per camera per month	\$ 10.00	10,000	Camera	\$ 100,000.00
Number of cities		6	City	\$ 600,000.00
			EOY Total	\$ 7,200,000.00
			Profit/(Loss)	\$ 5,745,000.00
Operating Statement, (EOY 8)				
Line Item	Income per Unit	#	Unit	Total
Revenue per camera per month	\$ 10.00	10,000	Camera	\$ 100,000.00
Number of cities		10	City	\$ 1,000,000.00
			EOY Total	\$ 12,000,000.00
			Profit/(Loss)	\$ 10,545,000.00

## References

- "Super Circuits." Telephone interview. 19 Apr. 2016.
- Hess, Eric. "NEC, NeoFace Software, Biometrics and Video Analytics." Telephone interview. 20 Apr. 2016.
- Clark, Crystal. "Envera Systems, Active Video Surveillance." E-mail interview. 21 Apr. 2016.
- "Active Video Surveillance." *Envera Systems RSS*. N.p., n.d. Web. 28 Apr. 2016. <<http://enverasystems.com/active-video-surveillance/>>.
- "AWS Total Cost of Ownership Calculator." *Amazon Web Services, Inc.* N.p., n.d. Web. 28 Apr. 2016. <<https://aws.amazon.com/tco-calculator/>>.
- Balko, Radley. "U.S. Cities Pay out Millions to Settle Police Lawsuits." *Washington Post*. The Washington Post, n.d. Web. 28 Apr. 2016. <<https://www.washingtonpost.com/news/the-watch/wp/2014/10/01/u-s-cities-pay-out-millions-to-settle-police-lawsuits/>>.
- "Chicago's NeoFace Jails First Criminal." *Government Technology*. N.p., n.d. Web. 28 Apr. 2016. <<http://www.govtech.com/public-safety/Chicagos-NeoFace-Jails-First-Criminal.html>>.
- "Citywide Surveillance Systems." N.p., n.d. Web. 28 Apr. 2016. <<http://www.supercircuits.com/law-enforcement/citywide-surveillance-systems>>.
- Erickson, Scott. "The Financial Impact of Crime." *The Daily Caller*. N.p., n.d. Web. 28 Apr. 2016. <<http://dailycaller.com/2010/02/19/the-financial-impact-of-crime/2/>>.
- "Explore Salary & Equity Data." *AngelList*. N.p., n.d. Web. 28 Apr. 2016. <<https://angel.co/salaries>>.
- Gao, George. "What Americans Think about NSA Surveillance, National Security and Privacy." *Pew Research Center*. N.p., 29 May 2015. Web. 28 Apr. 2016.

<<http://www.pewresearch.org/fact-tank/2015/05/29/what-americans-think-about-nsa-surveillance-national-security-and-privacy/>>.

Hess, Eric. "Arizona Uses NEC NeoFace Assure to Catch Felony Forgery Suspect." *LinkedIn Pulse*. N.p., n.d. Web. 28 Apr. 2016.

<<https://www.linkedin.com/pulse/arizona-uses-nec-neoface-assure-catch-felony-forgery-suspect-hess>>.

"How Many Video Surveillance Cameras Are There in This World?" *StorageServers*. N.p., 30 July 2014. Web. 28 Apr. 2016.

<<https://storageservers.wordpress.com/2014/07/30/how-many-video-surveillance-cameras-are-there-in-this-world/>>.

"IEEE Conference Registration." *IEEE*. N.p., n.d. Web. 28 Apr. 2016.

<<http://icc2015.ieee-icc.org/registration>>.

"Mass Surveillance in the United Kingdom." *Wikipedia*. Wikimedia Foundation, n.d. Web. 28 Apr. 2016.

<[https://en.wikipedia.org/wiki/Mass\\_surveillance\\_in\\_the\\_United\\_Kingdom](https://en.wikipedia.org/wiki/Mass_surveillance_in_the_United_Kingdom)>.

"NEC Global." *NEC Global*. N.p., n.d. Web. 28 Apr. 2016. <<http://www.nec.com/>>.

"Online Advertising Placement and Pricing." *Quirks Marketing Research Media*. N.p., n.d. Web. 28 Apr. 2016. <[http://www.quirks.com/advertise/online/web\\_ads.aspx](http://www.quirks.com/advertise/online/web_ads.aspx)>.

Pearson, Michael. "Tamir Rice Shooting Lawsuit Settled for \$6 Million." *CNN*. Cable News Network, n.d. Web. 28 Apr. 2016.

<<http://www.cnn.com/2016/04/25/us/tamir-rice-settlement/>>.

Puglia, Frank. "NEC Releases Latest Version of NeoFace® Industry-leading Facial Recognition Identification Technology." *Business Wire*. N.p., n.d. Web. 28 Apr. 2016. <<http://www.businesswire.com/news/home/20150803005064/en/NEC-Releases-Latest-Version-NeoFace%C2%AE-Industry-leading-Facial>>.

Rice, Derek. "State of the Market: Video Surveillance." *SDM Magazine*. N.p., n.d. Web. 28 Apr. 2016. <<http://www.sdmmag.com/articles/92006-state-of-the-market-video-surveillance>>.

"San Jose Office Space." *42 Floors*. N.p., n.d. Web. 28 Apr. 2016. <<https://42floors.com/>>.

Stroud, Matt. "Did Chicago's Facial Recognition System Catch Its First Crook?" *The Verge*. N.p., 08 Aug. 2014. Web. 28 Apr. 2016. <<http://www.theverge.com/2014/8/8/5982727/face-wreck-how-advanced-tech-comes-up-short-for-police>>.

"Technology." *City of Chicago* :: N.p., n.d. Web. 28 Apr. 2016. <<http://www.cityofchicago.org/city/en/depts/oem/provdrs/tech.html>>.

"What It Costs: Ad Prices From TV's Biggest Buys to the Smallest Screens." *Advertising Age News RSS*. N.p., n.d. Web. 28 Apr. 2016. <<http://adage.com/article/news/costs-ad-prices-tv-mobile-billboards/297928/>>.

**Appendix C:**  
**Résumés of Team Members**