School of Science and Technology

CSD3999

Software Development Project

Autumn/Winter term

2017/2018

Date: 25 April 2018

Supervisor: Michael Heeney

Student Name: Adam Jarzebak

Student ID Number: M00525614

Campus: Hendon - London

Title: Birds eye new image analysis using OpenCV

School of Science and Technology

……..

(Student Name)

……..

(Student ID Number)

Module Number: **CSD3999**

I hereby confirm that the work presented here in this report and in all other associated material is wholly my own work. I confirm that the report has been submitted to TURNITIN and that the TURNITIN results are on media device attached to this report. I agree to assessment for plagiarism.

Signature: ……..

Date: ……..

Abstract

This report and application aim to develop a platform which helps to analyse and have better understanding of video stream of certain interest.
This report presents the work and research that has been done for the final year project “Birds Eye New Image Analysis Using OpenCV”. The purpose of this report is to include all reports that have been written previously and summarize preceding work for this project. Reading this report reader will get a basic understating of what the OpenCV is as well as associated to its software and different technologies. The leading software for this project is python programming language. Python in this case is just a wrapper for entire library of C++ coded tools. Certainly, main advantage is very quick adaptation of most of available tools without time consuming programming prior to testing and seeing results.
This project consists of several phases, accordingly to the plan provided in the project requirements. The first stage was an initial planning and discussions as well as interviewing and meeting with people that would be possibly providing some feedback to the project.
Following step was research and literature review which will be described in this paper, also entire software implementation and testing.
Initially suggested ideas in the project proposal, about crowd behaviour did not shaped well into this project, hence couple of small changes were made towards the direction of this project. While the new concepts have succeeded in reaching a deeper level of understanding of the problem undoubtedly delivering to the project new perspectives with reference to this task. A very important aspect of the application will be the matching model, which involves many other components in order to successfully perform this task. Matching is essentially gathering information about targets and matched with some pre-defined patterns. Following by detailed explanation of how it was achieved, tools and software used.

Table of Contents

Introduction

Within last couple of years, the plague of fly-tipping has increased and started to spread to all over country. Fly-tipping statistic for England, 2016/17, show that this has become serious problem for governments as well as local communities. Since the big number of this type of incidents occur on the motorways, this problem has also considerable impact on number of accidents. “For the 2016/17-year, local authorities in England dealt with around 1 million (1,002,000) fly-tipping incidents, a 7% increase from the previous year.”
- as identified by the Department for Environment Food & Rural Affairs (2017).

The possibility of monitoring the environment for many years has increased the chances of detecting a potential threat to people. In recent years, the advancement of technology has gone a long way towards making the most of accurate recording and automatic detection of dangerous situations. This causes alarm to the observer and reactions from the security services.
The objective of my project is to build a system that will observe human' behaviour, and analysis it in disparate ways. According to Sjarif et al. (2014), crowd analysis consists of four phases: crowd density estimation, crowd motion detection, crowd tracking and crowd behaviour understanding.

In the initial project proposal, as well as interim report was proposed number of problems that relates to the crime as well as inappropriate behaviour of people. Likewise, in project proposal, the number of potential solutions was presented. During the initial software development, there was number of different issues that would not allow to finish the project on time, hence, with the approval of my supervision, slight changes have been undertaken of the project direction. However, topics do not differ very much, hence, all of the knowledge gained so far can be applied to this project which is very similar in some sense to what was proposed at the beginning.
At the beginning of the project the system development was more focused on people’s behaviour. Then moving on car and other object recognition and eventually focused purely on motion and people detection. At this stage the system is able to

successfully recognize even small movement in the stream of video. Followed by extraction of objects of interest. Successful detection of these objects is potentially very useful for more deeper understanding of what is actually proceeding on a video. These data are used to identify any abnormal behaviour and potentially decrease occurrence of this type of incidents.

The system will search for patterns from data collected using a camera and choose which patterns are the most relevant for given aim. In the first place an aim is to develop a system to perform anomaly behaviour detection.
The result of this analysis may be data that can be represented in different forms, such as row in tables, graphs, histograms, plot area.
System may not be able to able to fully understand the environment or be sure of accuracy of detected object, hence staff may need to be involved in order to system be fully functioning. OpenCV is an open source library for picture and video processing, initially presented more than decade back by Intel. From that point forward, various software engineers have added to the latest library advancements and new functionalities. Nowadays the library has more than 2500 enhanced tools available. It is widely used in disparate fields. To master every library element, it is necessary to study many books available on the topic of OpenCV. However, reading such more comprehensive material should be easier after comprehending some basics about OpenCV from this paper and referenced materials.

Keywords: image processing, image analysis, surveillance, object tracking, object detection, OpenCV library

Background and literature review

In 2015/2016, local authorities in England dealt somewhere in the range of 900,000 occurrences of fly tipping, which is a 5.6% expansion on numbers from the earlier year. In addition, more than 60% of fly tipping episodes included household waste. What's more, it's costly. In 2014/2015, local authorities in England spent almost £50m on clearing fly tipping. That is an expansion on 11% contrasted with the earlier year. Also, local government burned through £17.6m on required activities. So, fly tipping is a major issue that is just deteriorating - provided by the Department for Environment Food & Rural Affairs (2017).

This project proposes a certain solution to this problem, namely, by constant mapping of fly-tipping areas should improve ability to identify who is behind these acts and reduce number of this incident.
This will be achieved by deploying a drone up in the air and which then will have ability to see the scene and using a camera and specifically designed software detect these accidents.
The hardest part of this project is write a software in order to perform mentioned above tasks.
OpenCV appear as very satisfactory tool for this project’s purpose considering a following succeeding reasons.
First of all, it is a free and open source software. Secondly, global community provide an extensive support which is also very active. It is well known framework and has been already implemented in various of similar projects by many titled people. Thirdly, many researchers have already accomplished great outcomes using this library. One of the most inspiring article I came across was about training detectors and recognizers in Python and OpenCV. In this article Howse (2014) describes how to train own detectors then recognize desired objects in given either video stream or single frames. One cannot deny that we must distinguish carefully between couple of terms while using this software. Based on my personal experience, very often I found these terms confusing during, either reading documentations or using this library. Saying that I am referring to following terms; detect, recognize and track. First term refers to finding a location of given object in an image. Second, decide the subtype

or the unique character of a detected object. Lastly, track checks if the same detected object appear in an array of images.

Following article adopted two types of the detection models. This project also will be based on these types of detector. Namely, he talks about Haar Cascade and Local binary pattern (LBP). Haar cascade detects light-to-dark edges, corners, and lines at multiple scales. Much reliable method, what leads directly to more accurate results, however this model is more time consuming. Local binary pattern model is definitely faster in comparison with previous but less accurate. Both of these training detectors return XML format. The main reason why this system needs to have own detector is because, desired objects in this case cars, could potentially be situated in various different positions. These detectors allow us to train our classifier according to provided data. And this is what will be attempting to achieve in further parts of the project. Howse (2014) in his project was working on face detections. It does not make any significant difference what type of the object we are detecting. The principles of training detector remain the same.

The system needs to be endowed with strong detection features, due to required high accuracy for identifying objects. Manlises et al. (2015) prove that this is current technology this task becomes not only effortless to accomplish but also it can result in higher accuracy and performance. Even when using such a “small” in size technology what is called Raspberry Pi. His article talks about processing data using OpenCV for detection. The webcam was utilized to catch pictures progressively. The pictures were handled at the microchip. At the point when people on foot were recognized, the chip played the recorded voice and yields it to the interfacing speaker to advise those walkers or the person on foot to hold up before intersection until the point when such time. Assumptions of this system are nearly identical. Our drone is going to act as a flying bird. Having little computer chip like raspberry on the board would be able to receive the images from the camera attached to it. Following by processing this data, recognising, gathering evidences and lastly alert adequate people about occurred event. At this stage of the project it is hard to answer the question whether simple Raspberry Pi or even other single board with a better capability will be enough to perform these heavy calculations. Therefore, single board computer could be used as a device to transfer data from a drone to another machine, which then would perform these tasks.

Object detection from a certain distance, moreover, very often background of scanning area will be either uniform such as; landscapes or cities are too colourful. One of the examples would huge cornfields. Where homogenous type of the field is leading in the image background. In order to mitigate this issue, certain algorithms need to be entwined into our program. Author of this article, is detecting of moving objects through colour thresholding.

In image processing area and segmentation algorithms in view of thresholding, the force of the picture (grayscale) is normally gotten keeping in mind the end goal to separate the regions of the items and the background. The segmentation on the threshold functions admirably when the picture has a high intensity in the contrast, this trademark is vital to make a decent characterization of the pixels. All of these procedures where very well explained by Barba et at. (2017). These techniques, not only will be implemented when detection of the object but can be used in live mode. Live mode will be explained in detail later in this paper. But briefly saying this mode allows to view in current time what a camera from the drone is currently viewing. This possibly will help a user to resolve unrecognised objects or situations. The project proposal introduces the importance of surveillance, I would like to underline this term again because even though the project direction has changed slightly, the problems that we are trying to solve have very similar origin. Therefore, different methods of surveillance will be applied in certain ways to this project.

A viable security and surveillance frameworks can give provide early notices in the event of any sort of unwanted event. Nonetheless, if the observation framework fits for wandering inside the region of surveillance, more area of intrigue can be seen with ideal number of surveillance gear under compelled spending plan. This what the system is going to do. Advantages of real-time surveillance is a reaction time. Nothing would be quicker and easier to find out than having a live time detection system. Hossain et at. (2015) were targeting this issue. The team have built a small robot which have had embedded systems to perform image classification and object detection which then can be utilized for various surveillance purposes. Finally, I would like to make a research about thermal images. What are the advantages of using the thermal camera over standard RGB camera. RGB camera with little enhances are being widely used in CCTV equipment. As a part of hardware implementation, I will replace RGB camera with thermal camera to see what would produce better results. Fortunately, some research has been done toward this

subject. Below, are illustrated one the outstanding articles, which was written by Leira et at. (2015). Their studies on this subject show capability of achieving thermal images processing. During this project they used unmanned aerial vehicles, respectively low-cost and light machines, which can carry other equipment in order to perform these tasks. This system incorporates the use of a thermal camera and on-board processing power to perform real-time object detection, classification and tracking of objects in the ocean surface. Producing very satisfying results. Similarly, to this project where execution of this system based on thermal video data from flights of a drone and is found to be able to detect 99,6% of target objects. Of the detected objects, only 5% were false positives.

Requirements specification

This section describes the requirements analysis that took place before development.
There were many projects based on similar requirement, but not exactly with the same results or not necessary slowing these problems.
The application being developed hopes to help with the first two problems, identification of potential danger to an environment and secondly improve vision analytics by implementation of more complex algorithms. Target users: large companies, governments, analytics companies, etc.

Video streaming

Hardware

The Raspberry Pi Camera board features

- Improved Resolution
- 8 megapixel native resolution high quality Sony IMX219 image sensor
- Cameras are capable of 3280 x 2464 pixel static images, remaining High Quality Capture video at 1080p30, 720p60 and 640x480p90 resolutions
- All software is supported within the latest version of Raspbian Operating System

- 1.12 µm X 1.12 µm pixel with OmniBSI technology for high performance
- Optical size of 1/4"

Raspberry Pi features

The current Raspberry Pi Model
Here is the full list of specs for Raspberry Pi 2 Model B:

- SoC: Broadcom BCM2836 (CPU, GPU, DSP, SDRAM)
- CPU: 900 MHz quad-core ARM Cortex A7 (ARMv7 instruction set)
- GPU: Broadcom VideoCore IV @ 250 MHz
- More GPU info: OpenGL ES 2.0 (24 GFLOPS)
- Memory: 1 GB (shared with GPU)
- USB ports: 4
- Video input: 15-pin MIPI camera interface (CSI) connector
- Video outputs: HDMI, composite video (PAL and NTSC) via 3.5 mm jack
- Audio input: I²S
- Audio outputs: Analog via 3.5 mm jack; digital via HDMI and I²S
- Storage: MicroSD
- Network: 10/100Mbps Ethernet
- Peripherals: 17 GPIO plus specific functions, and HAT ID bus
- Power rating: 800 mA (4.0 W)
- Power source: 5 V via MicroUSB or GPIO header
- Size: 85.60mm × 56.5mm
- Weight: 45g (1.6 oz)

Receiving video and processing is another part of the backend processes. An equipment and software are responsible for processing and analysis of the video received from a stream endpoint. High requirements of heavy processing inclined to choose more powerful server which in this case is a Digital Ocean server.

Server features

Digital Ocean hosting services

- 4 GB Memory
- 80 GB Disk / LON1
- System - Ubuntu 16.04.3 x64
- CPU – 4 x 2.4 GHz

Software

This section will briefly introduce software and libraries used for the project purposes.

Dependencies requirements

- Linux OS
- Python 2.7 - A multi-paradigm scripting language
- NumPy 1.8 - A math library for fast array operations with Pythonic syntax
- OpenCV 2.4 - A computer vision library with lots of algorithms and I/O features OpenCV Python treats images as NumPy arrays.
- Flask - A web interface framework
- SQLite – Database library
- Scipy – Scientific library for python
- Pygal – Graphics and visualisation package

Front-end

This part also will include the administrative view, which actually is majority of this component. GUI components will be very simple which will allow admin to control the camera behaviour. Including switching between observation mode, tracking mode, recording mode and other additional views. Coupled with observation mode subcomponents such as selection of live mode views. This part of the project was optional part of the project. This is only improvement and addition to the software. Please notice that entire software will perform well as expected without GUI implementation.

Flask is a web framework. This means flask is equipped with tools, libraries and technologies that allow to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Flask is part of the categories of the micro-framework. Micro-framework is normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins. In the case of Flask, its dependencies are:

- Werkzeug a WSGI utility library
- Jinja2 which is its template engine

Using a template engine will save you a lot of time when creating your application but also when updating and maintaining it.

Description of the flask framework should provide enough evidence to support my decision about choosing this great tool to develop my application. In my development work, least issues I had with building a web interface, is due to seamless and logical options provided by developers and community who develop this framework. On top of this, this tool has many relations with another libraries or frameworks. Which allowed to extend this project in non-trivial way. To give an illustration, this app has function such as;

- Security layer
- Graphs
- Extended visual effects (bootstrap)
- Additional JS libraries
- Google analytics (web analytics purposes)

Back end

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed in C++ language environment, however recently has become popular programming using this framework in python.
Before installing a OpenCV, the linux system is required. For this project, the Digital Ocean server has been utilized.

System Design

Project overview

Graph below shows an overview of how the system cycle will behave.

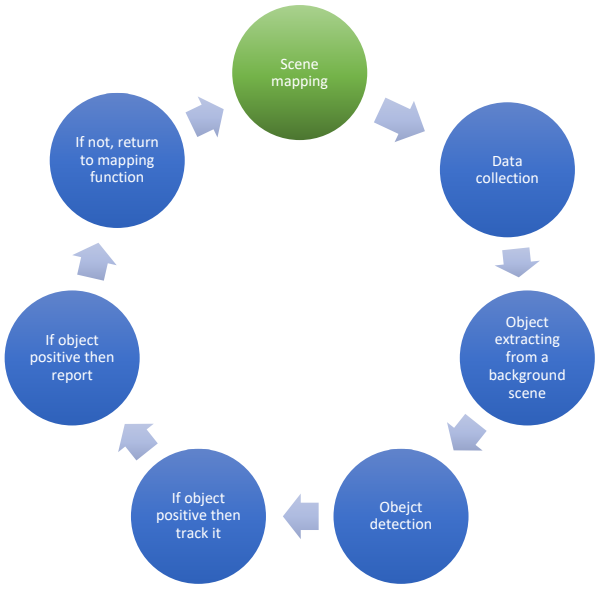


Figure 1. Graph represents a flow of the system

Objective is to develop a system which can observe, identify successfully recognize fly-tipping frauds or at least alert a user when some suspicious behaviour was detected by system. The next step, following the recognition of probably incident is

gathering of any evidences that accompanied to this event. Mainly it is going to be a video recording or pictures. However, it could also be a representation of data in numbers and displayed in different forms; such as row in tables, graphs, histograms, plot area, 3D technology.

System Architecture

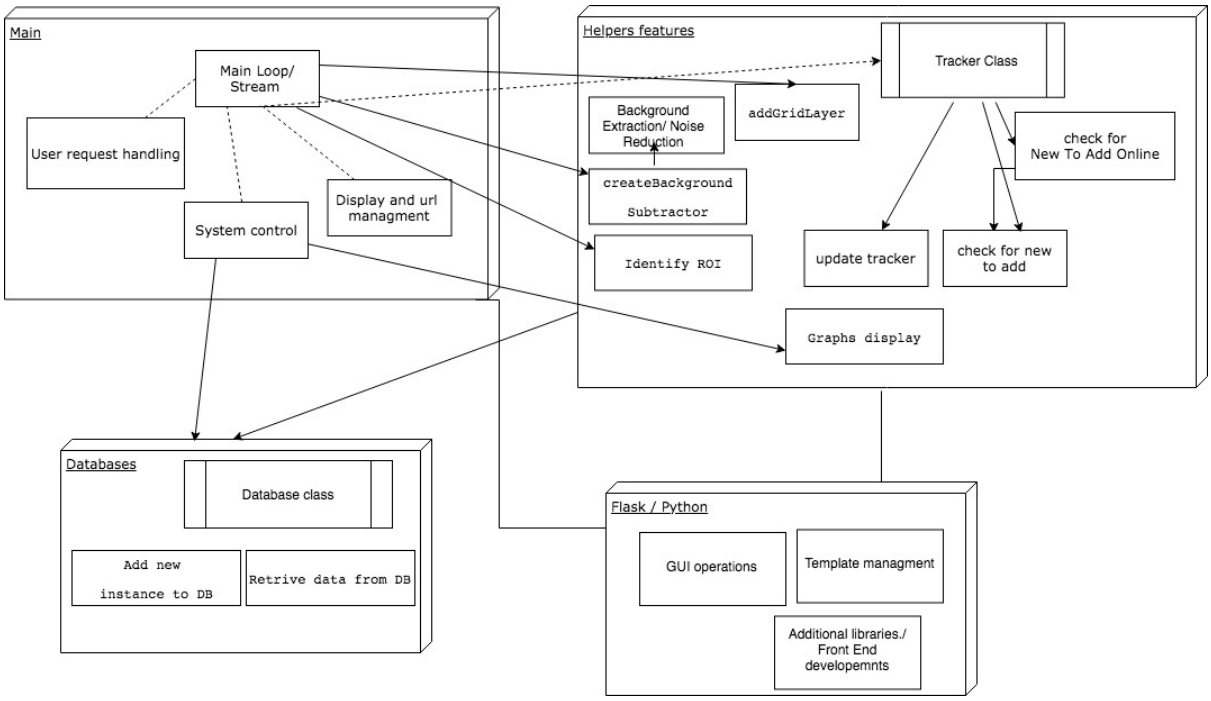


Figure 2. System architecture

System key components

- Observation
- Scanning
- If goal has been detected
    - Record evidence
- Otherwise
    - Keep scanning
- Access to live mode
- Multiple view modes

The fully developed product is targeting on government or institutions and organizations that are looking for this specific type of system.

Core system functions

Observation mode

Allows user to see the image from the camera. This mode is fully flexible, user can zoom in/out, change view mode.

Available options in live view mode:

- Greyscale
- Colour
- Zoom in
- Zoom out
- Canny Edge Detection
- Image Gradients
- Contours in OpenCV
- Interactive Foreground Extraction using GrabCut Algorithm

Tracking mode

This if default mode for this system. Once the system was deployed it will be set to scan videos for an object that is set a goal. Camera would need to be set with an area to be scanned. However, this is not part of this project. We will use a drone just for testing purposes and area to be scanned will be randomly chosen. In the future, the environment will be based on likelihood of this type of accident to occur. Information of this character is provided by governments. “Consistent with previous years, the most common place for fly-tipping to occur was on highways, which accounted for almost half (49%) of total incidents in 2016/17. The number of highway incidents has increased by 4% from 2015/16.” – information published by Department for Environment Food & Rural Affairs (2017).

Gathering evidences

When system detects desired level of confidence that current view consists of potential incident or identify any suspicious objects within a recording, this would be recorded. Recording has an option of accessing information from past. Since this mode is continuously recording an environment. Actions undertaken when objects found:

- Timestamp taken
- Staff alerted
- Data save into a database

Development Platform

The development platform and technologies for each components of the system are listed in the following table:

Unit	**Environment**
Training a custom detector	OpenCV using C++
GUI for control panel	Python Flask Framework
Detection software	OpenCV using Python
Processing server	Linux based server
Streaming device	C++, Raspberry
Hardware	RGB camera, Raspberry Pi

These components will be developed under a Linux based OS and will compatible with another Linux OS.

System Architecture

Most of the system include software to execute object detections using OpenCV and python to control entire system. As well as live streaming of the video to the base system. These will be called backend operations. Another key thing to remember is

that, in order to execute this operation will be required to have some prerequisites prior to this activity.

Back-end

In this section I will describe the outlines of main components of the system. Mainly, the backend system in divided into two parts:

- Streaming video
- Receiving video and processing

Streaming video is performed in live mode and its software is installed on the single board computer called Raspberry Pi.
This device is equipped with very small and at the same time good quality camera. Small single board computer is directly connected to Ethernet network. Because of University limitations it was required to install additional software to bypass blocked ports and to stream be accessible beyond local network. For this purpose, ngrok software was utilized. Ngrok allows to expose local servers behind NATs and firewalls to the public internet over secure tunnels. Please refer to the example below for configuration example.
For more detailed information about configuration please vision official documentation: https://ngrok.com/docs.

```
region: eu
update_channel: stable

tunnels:
        vision-app:
                addr: 4000
                proto: http
                bind_tls: false
                subdomain: visionstream
                inspect: false
```

Figure 3. Sample ngrok configuration

The last and most important part of the streaming is a software itself, without it would not be possible to achieve it. While simple web application nowadays is not a big problem, however when it comes to dealing with streaming, the word ‘simple’ does not suit to its description.
Having said that, multiple attempts have been done before reached a stage when a stream was reliable enough to receive it on another endpoint. While setting up entire streaming process, there are multiple way of debugging, it is going to be either simple checking of the domain which is being set up, or watching logs provided by software.

Receiving video and processing this part of the system is responsible for receiving a stream from URL or open local video and preform number of calculations to be able to successfully withdraw essential information from video frames.

Database design

This project includes databases, as the server will have to store data of the trackers information in databases using open-source SQL derivatives such as SQLite3 in order to retrieve these information for further use such as graphs and analytics, this means that this database will need to be designed to be secure and efficient, especially in the data-tier hosted on the server.

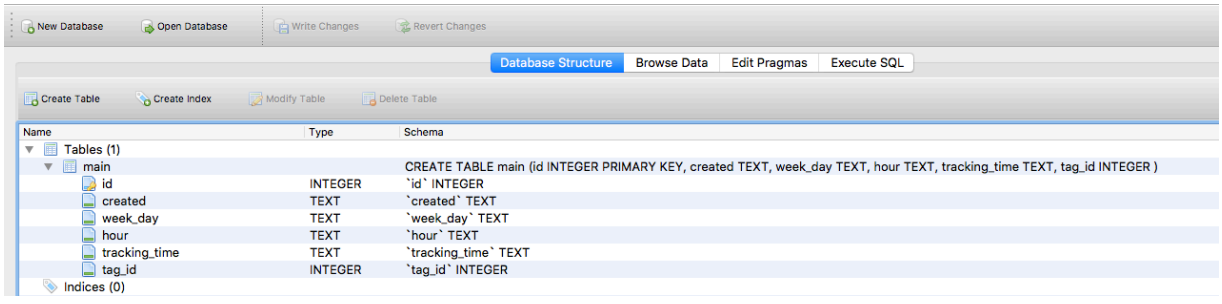


Figure 4. Database structure

This simple database structure allows for a system to save some important information about trackers. First data field is ‘ID’ of a tracker, this is a default field for almost all databases. This filed is very important when comes to more complex

database structures. Each of the row require to have at least one unique filed, so that it could be always recognized without any conflicts. It also could be utilized as a ‘connecter’ or so called ‘foreign key’ between multiple tables. Second data field is designated for timestamp of when a tracker was initialized. Followed by week day field, which simply tells us day of a week, required for graphs and other analytics purposes.

Implementation and Evaluation

Web interface

This part of the project highlights all work which is done in background and allow users to have very feasible access to all features provided.
The User Experience was very important in the design of this application, as the end-user may not have much prior knowledge of using handheld devices, so this aspect of the application was carefully considered and the designs from the initial wireframe drawings changed to simplify the interface as much as possible and make the navigation around the application simple enough that anyone using this application could easily understand the interface and quickly get known all available features without too much cumbersome.

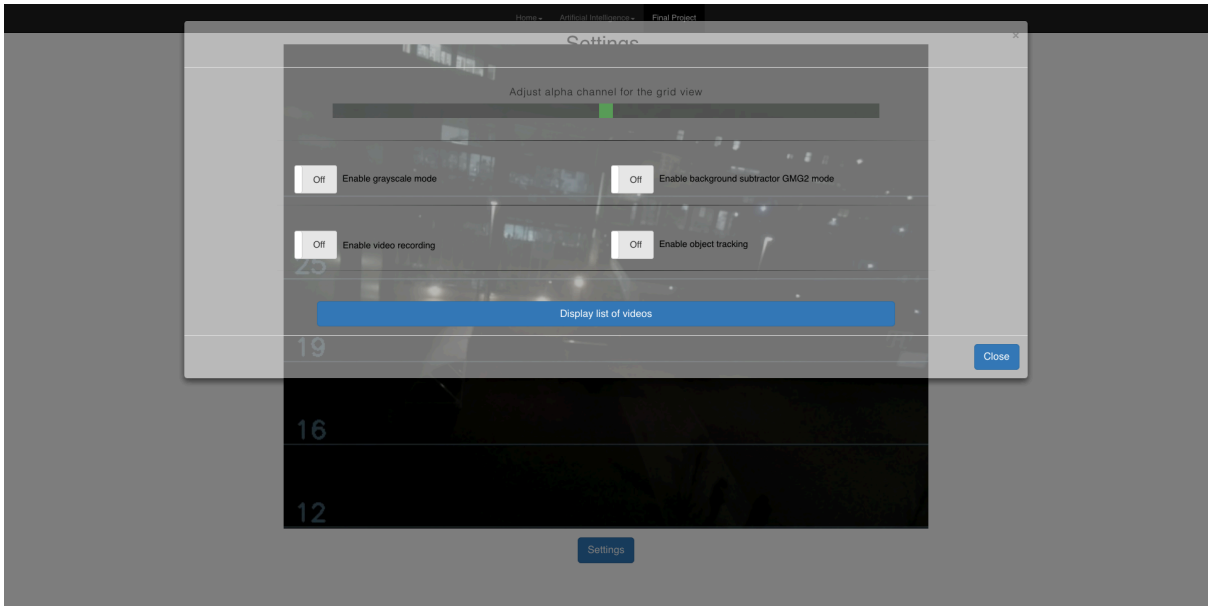


Figure 5. Web interface, settings view

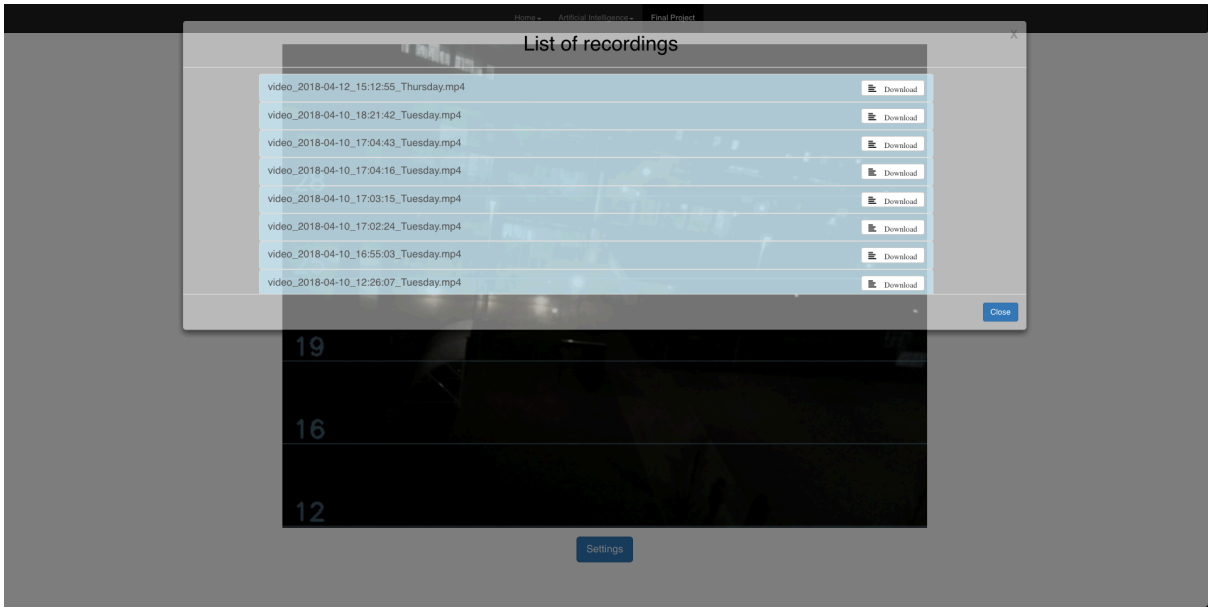


Figure 6. Web interface, files list display

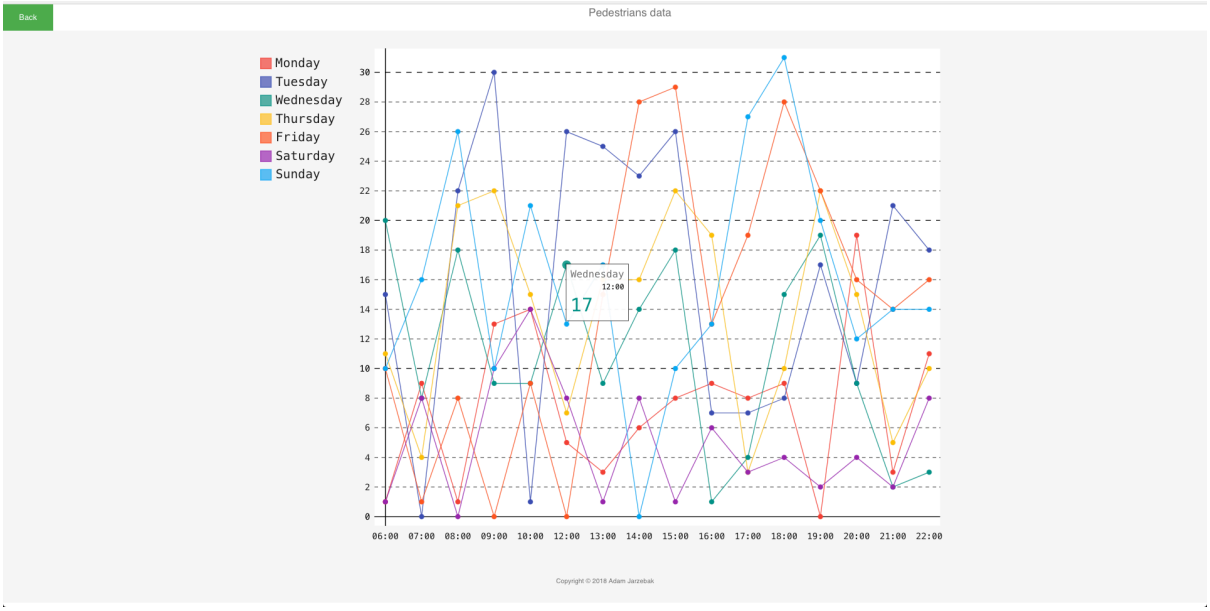


Figure 7. Data visualisation and graphs

Code base

Components of a processing part

- ROI Selection
- Tracking
- Extracting data from trackers
- Patterns matching
- Database maintenance
- Plotting data from a database into a graph

Components of live view mode

- View with a grid enabled
- View with grayscale enabled
- View in background extraction enabled
- View with tracking mode enabled

Components of recorded videos

- View recording using web interface
- Download a recording

Tracking flow:

- ROI selection
- Tracker selection
- Initialization of a selected tracker
- Passing object to a tracker
- Maintain a tracker
- Continuous seeking and mapping an image for new ROI
- Pattern matching
- Saving data into a database

Tracking is the most important part of this project. Before approaching tracking there needs to be a couple of operations performed beforehand
First of all, frame which are directly coming from a streaming camera have a large number of noise. Therefore, it is essential to run certain tasks in order to make frames 'cleaner' for further processing. There is vast number of filters provided by OpenCV. Some of them were applied successfully to this project.
These are following:

- fgbgMask
- MorphologyEx
- GaussianBlur
- MedianBlur
- BilateralFilter

Above filters allowed to receive an output in more transparent way, obviously they were carefully selected. Depending on need and image coming from camera.
Next step is a ROI selection. In this function, program is extracting motion from background using createBackgroundSubtractorMOG2, next procedure is retrieving contours of the object that are in the motion and calculating the dimensions in order to determine whether the object is on the list of targets.
Before last stage is creating and maintaining trackers class. OpenCV ships with some build in trackers. No all of them are suitable, however some of them successfully run producing satisfying results.
Trackers considered tested evolutions:

1. BOOSTING
2. MIL
3. KCF
4. TLD
5. MEDIANFLOW
6. GOTURN

The best results have KCF tracer. It does not always successfully track object, but it has really good positive results and it does not require so much CPU processing.

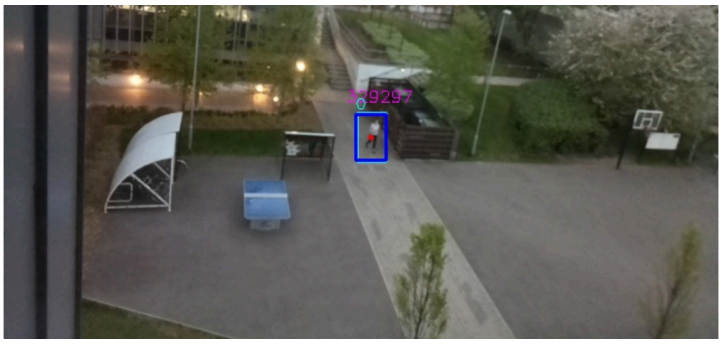


Figure 8. KCF tracker during operation

Last of stage is called pattern matching. System consist of couple already predefined contours which then are compared with those which are extracted from trackers.

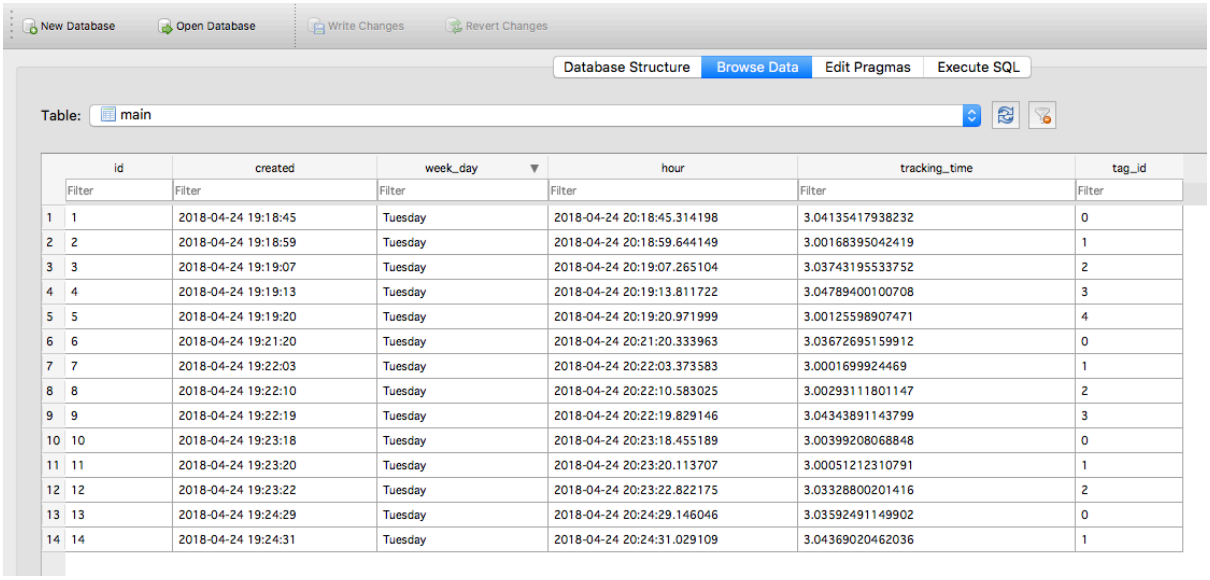
Database

SQLite3 is a compact free database you can use easily create and use a database. Though SQLite3 is not a full-featured database, it supports a surprisingly large set of the SQL standard, and is ideal for this project since is very easy in use and powerful enough to perform those simple operations. Database is stored on the same server hence, no need to switch between any extra services moreover, very quick and easy initialization can done using simple command from backend:

```
dbManage = DataManager()
dbManage.createUpdateDatabase()
```

All preparation and programming has been already done.

Below example of the data from database.



	id	created	week_day	hour	tracking_time	tag_id
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	2018-04-24 19:18:45	Tuesday	2018-04-24 20:18:45.314198	3.04135417938232	0
2	2	2018-04-24 19:18:59	Tuesday	2018-04-24 20:18:59.644149	3.00168395042419	1
3	3	2018-04-24 19:19:07	Tuesday	2018-04-24 20:19:07.265104	3.03743195533752	2
4	4	2018-04-24 19:19:13	Tuesday	2018-04-24 20:19:13.811722	3.04789400100708	3
5	5	2018-04-24 19:19:20	Tuesday	2018-04-24 20:19:20.971999	3.00125598907471	4
6	6	2018-04-24 19:21:20	Tuesday	2018-04-24 20:21:20.333963	3.03672695159912	0
7	7	2018-04-24 19:22:03	Tuesday	2018-04-24 20:22:03.373583	3.0001699924469	1
8	8	2018-04-24 19:22:10	Tuesday	2018-04-24 20:22:10.583025	3.00293111801147	2
9	9	2018-04-24 19:22:19	Tuesday	2018-04-24 20:22:19.829146	3.04343891143799	3
10	10	2018-04-24 19:23:18	Tuesday	2018-04-24 20:23:18.455189	3.00399208068848	0
11	11	2018-04-24 19:23:20	Tuesday	2018-04-24 20:23:20.113707	3.00051212310791	1
12	12	2018-04-24 19:23:22	Tuesday	2018-04-24 20:23:22.822175	3.03328800201416	2
13	13	2018-04-24 19:24:29	Tuesday	2018-04-24 20:24:29.146046	3.03592491149902	0
14	14	2018-04-24 19:24:31	Tuesday	2018-04-24 20:24:31.029109	3.04369020462036	1

Figure 9. Database fields

Testing and Results

Picture processing consist of methods that are applied on imported images, with a specific end goal to obtain an improved picture or to separate valuable data from it. It is a type of flag preparing in which input is a picture and output might be picture or qualities/highlights related with that picture. Picture or video processing essentially incorporates the following three stages:

- Importing images or video stream, this can be accomplished using variety of different tools or software, for this project it is not so crucial, however it is imperative to make sure so that an input will not lose its quality and will be successfully transferred
- Image analysis and its manipulation in order to achieve best result
- Output in which result can be modified picture or report that depends on picture analysis output

Videos(data) will be primarily saved on a hardware device attached to the drone, if conditions permit, data will be transmitted in live time to the machine based on the ground. This will allow to have a live view from a camera and control a drone.

Software is not dissimilar to other physical procedures where inputs are provided and returning an output. Although programming contrasts in results of failure. In case of failure for most physical devices it will be predicted, or it will happen in small range of ways. However, when considering this project where device will be flying and interacting in some sense with people, it is very important to reduce or totally eliminate any failure. In case of failure system will need to be prepared to handle this event. By differentiate, programming can fail in numerous strange ways. Therefore, for both software and hardware, it is very critical to perform multiple *formative evaluations.*

Testing required many hours of watching, and analysis of the output from the program, below attached some examples from the software. This results led this project to many improvement and finished with very satisfying results of being able to detect and track objects of interest.

```
root@ubuntu:~/imageAnalysis/main# python app.py
\2018-04-24 19:21:12,794 : DEBUG : Started up analysis app
2018-04-24 19:21:12,799 : INFO :  * Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
2018-04-24 19:21:12,799 : INFO :  * Restarting with stat
2018-04-24 19:21:13,239 : DEBUG : Started up analysis app
2018-04-24 19:21:13,244 : WARNING :  * Debugger is active!
2018-04-24 19:21:13,244 : INFO :  * Debugger PIN: 139-900-675
2018-04-24 19:21:14,601 : INFO : 158.94.126.195 - - [24/Apr/2018 19:21:14] "GET /vision HTTP/1.1" 200 -
2018-04-24 19:21:14,724 : INFO : Successfully opened a stream
2018-04-24 19:21:14,852 : INFO : 158.94.126.195 - - [24/Apr/2018 19:21:14] "GET /motion_detection HTTP/1.1" 200 -
2018-04-24 19:21:17,178 : INFO : 158.94.126.195 - - [24/Apr/2018 19:21:17] "GET /_apiQueryTracking?apiQ0=true HTTP/1.1" 200 -
2018-04-24 19:21:17,209 : DEBUG : Tracking enabled
2018-04-24 19:21:18,811 : INFO : Dist between current tracker and new found ROI. 212.360542474 . Add new tracker
2018-04-24 19:21:23,940 : INFO : Dist between current tracker and new found ROI. 298.519681093 . Add new tracker
2018-04-24 19:21:31,442 : INFO : Found matching patter 707.150260855 . Tracker 1
2018-04-24 19:21:39,317 : INFO : Track failure reported for tracker: 1
2018-04-24 19:21:42,362 : DEBUG : Successfully saved data to main db.
2018-04-24 19:21:42,362 : INFO : Tracker deleted 1
2018-04-24 19:21:42,403 : INFO : Dist between current tracker and new found ROI. 169.543504741 . Add new tracker
2018-04-24 19:21:42,850 : INFO : Track failure reported for tracker: 3
2018-04-24 19:21:45,879 : DEBUG : Successfully saved data to main db.
2018-04-24 19:21:45,880 : INFO : Tracker deleted 3
2018-04-24 19:21:45,923 : INFO : Dist between current tracker and new found ROI. 186.303515802 . Add new tracker
2018-04-24 19:21:46,652 : INFO : Track failure reported for tracker: 4
2018-04-24 19:21:49,688 : DEBUG : Successfully saved data to main db.
2018-04-24 19:21:49,688 : INFO : Tracker deleted 4
2018-04-24 19:21:50,690 : INFO : Dist between current tracker and new found ROI. 226.294056484 . Add new tracker
2018-04-24 19:21:52,647 : INFO : Track failure reported for tracker: 5
```

Figure 10. Server's logs

One of the goals during evaluation will be to examine what is a margin of admissible error. It needs to consider a difficulty of measuring a correct result from unstructured dataset, consisting of crowds and unknown background properties for images or video, as well as natural conditions.

As mentioned above, none of the tracers are perfect. They produce a lot of false negative results which end up taking up space and processing.

Below example of negative results of tracking:

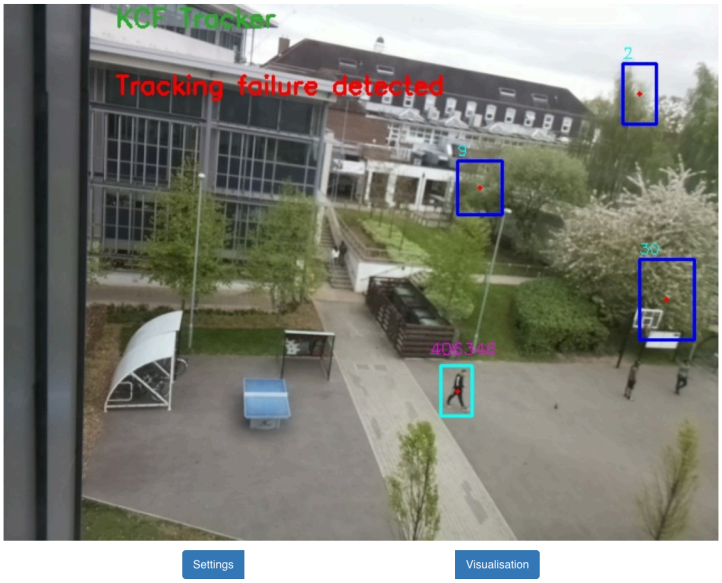


Figure 11. Example of negative failing tracking

The following elements will be considering during this evaluation:

Usability

- Navigation
- Error checking
- Correctness according to requirements
- Ease of installation
- System requirements

Pedagogic aspects of the project

- Potential role
- Clearly stated objectives
- Assessment and feedback
- Challenge and motivation

Research tasks

Notwithstanding the limitations of the methodology used which was specified in the project proposal, it was an important study in the development of how a system would potentially work and arrive at certain conclusion as well as finding another

solution to presented problems. In light of the work achieved so far, I would like to present some example of labour put into this project.
The most significant part of this scheme is: successfully recognize an object from a give stream of video. There already exist and are available detection models.
OpenCV supports a few kinds of detectors, including these two:

- Haar cascade - relatively reliable
(Detects light-to-dark edges, corners, and lines at multiple scales)
- Local binary pattern (LBP) – relatively fast
(Detects light-to-dark gradients at multiple scales)

Either of option requires to do some preparation, before attempting training.
This includes:

1) Negative training images
2) Positive training images
3) Create training data files
4) Create samples

Preparation itself is already very time-consuming process. In order to train good enough object detector, we are looking for around 2000 of negative images and about similar number for positives. Negative images can be collected arbitrarily from various websites. Another alternative is finding a source that contains packages of these images, which are already prepared for this process. Nevertheless, some extra work is required before using them.
Negative images are also called background images, and these terms are used interchangeably.
They can contain anything that we can only image, apart from the seeking object(s). Therefore, is required to go thought directory and make sure it does not contain an object we are looking for to detect in our program. Whereas for positive images, it is a little bit easier. One image is needed which contains an object we are looking for. In order to produce 2000 samples of positive images, we can use built-in functions in OpenCV library. This tool takes one image and using negative images, composes given number of samples. It is done by changing size, scale and rotating an object

then imposes this object on positive image. At the same time, it produces an 'info' file. An info file stores information about location of freshly placed objects onto negative images.

- opencv_createsamples -img superCar.png -bg bg.txt -info samples/info.lst -pngoutput info -maxxangle 0.5 -maxyangle -0.5 -maxzangle 0.5 -num 1938

Using info file given out by previous command we can make 'vec' file.
"Opencv_createsamples is used to prepare a training dataset of positive and test samples. opencv_createsamples produces dataset of positive samples in a format that is supported by both opencv_haartraining and opencv_traincascade applications. The output is a file with *.vec extension, it is a binary format which contains images."

- opencv_createsamples -info info/info.lst -num 1938 -w 48 -h 24 -vec carsData.vec

The last stage of this process is running the cascade training command. Below command uses haar cascade training. It takes a long time to train, but are definitely more accurate. You can train a Haar cascade using the following command.

- opencv_traincascade -data data -vec cars.vec -bg bg.txt -numStages 10 -nsplits 2 -minhitrate 0.995 -maxfalsealarm 0.5 -numPos 1800 -numNeg 1800 -w 48 -h 24

There are more alternatives, yet these will do. The fundamental ones here are the quantities of positive and negatives. General consensus is that for most practices, you need to have 2:1 proportion of positive vs negative pictures. A few circumstances may vary; however, this is a general lead individual appear to take after. Next, we have stages. I picked 10. Normally is need 10-20 in any event here, the more, the more it will take, and it is again exponential.
This process is very time and resources consuming, on machine with 8GB RAM took about 12 hours for 10 stages.

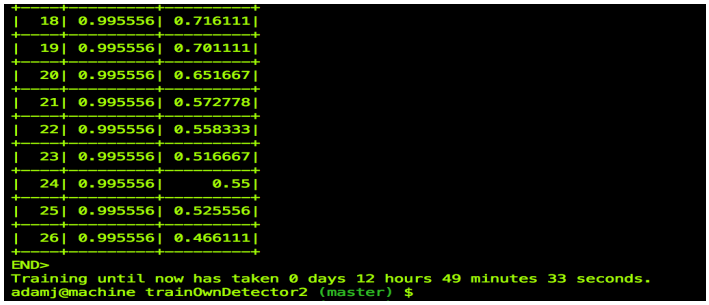


Figure 12. Sample output after a training command

These are only example of work that has been done. There is vast more details behind entire process of training an own detector. At this stage this seems to be challenging tasks, since achieving good results is not an easy task. So far detector is able to detect certain object but without too much accuracy. Analysis done on these attempts let me conclude that there needs to be higher number of samples. And perhaps better adjustment for parameters.

Challenges Encountered

Installing software such as OpenCV can be already challenging tasks. It requires deeper understanding of Linux based OS. Moreover, installing all dependencies not necessary is smooth as should be. Hence, debugging skills are required, as well as broad knowledge about C++ compilators would be very useful. I already installed OpenCV on two machines, one is Mac laptop, and another is RaspberryPi. Other challenges were related to training own detector. Where choosing parameters rarely were sufficient in order to successfully run a trainer. Various different factors are depending on this process. Such as, preparation itself is very important. More accoutre the preparation is it is more probable that training will succeed. Another challenge was encountered when adjusting variables used to identify ROI objects. Namely, set region of interest(ROI) has to be carefully selected, otherwise the system will end up with tracking loads of different object that are beyond a user scope. With this in mind, below is an example of failing tracker. Motion detection using a background subtractor of a type: MOG2, does not distinguish between movements in frames it simple returning a difference in image from previous frame and frame from current image. Therefore, objects which are in the motion would be picked up and passed to the tracker.

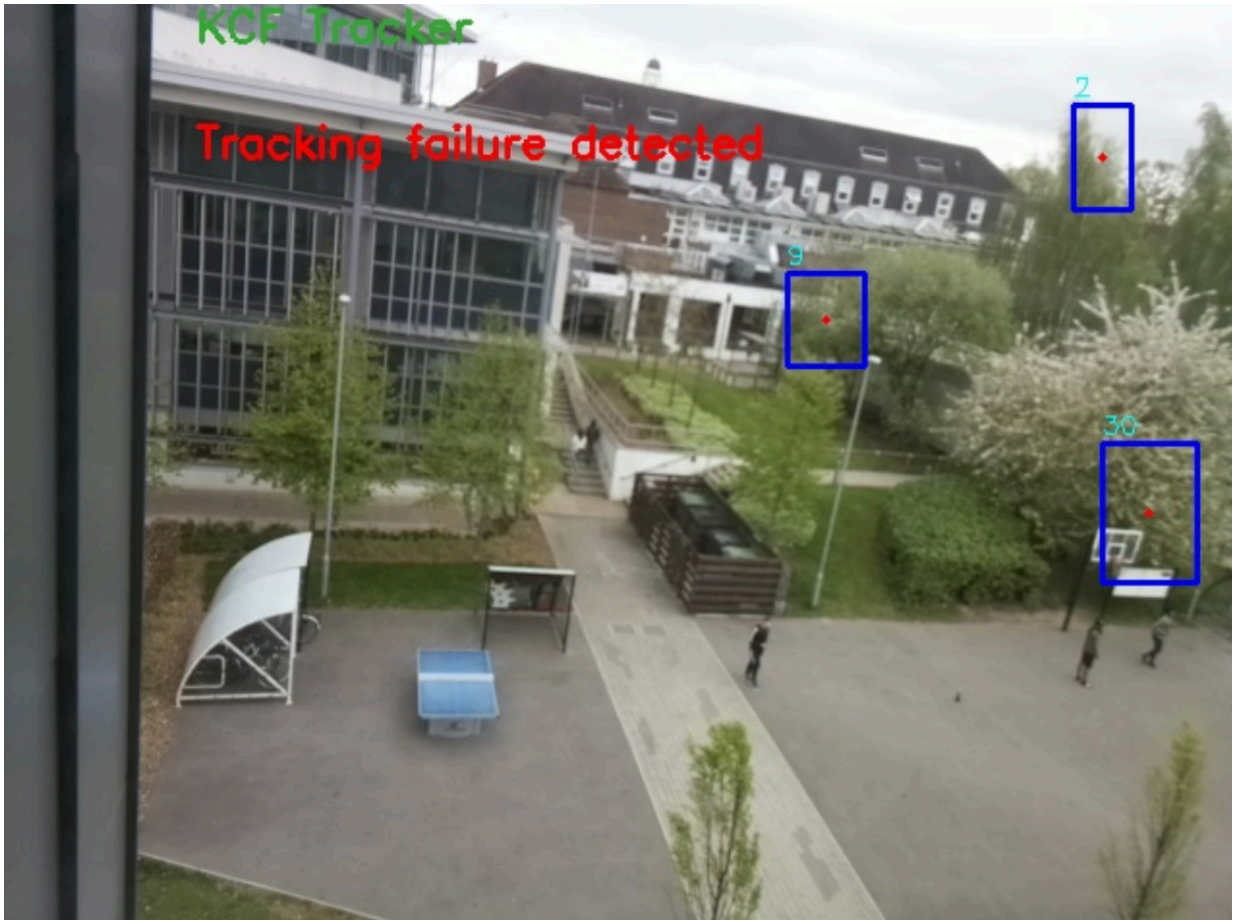
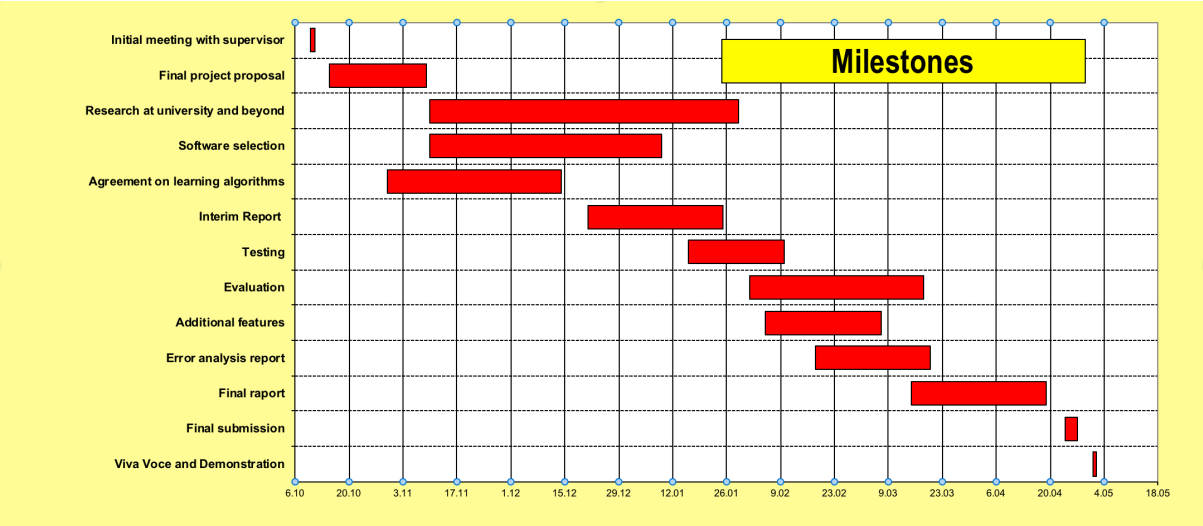


Figure 13. False, positive example of failing ROI selection

Milestones



Conclusion and further work

In conclusion, the application developed has successfully completed its goal, as can be determined by the success of the evaluation session, where many users have confirmed that it is a big improvement toward image analysis field. This project has allowed me to learn new technology that I did not know beforehand and apply it in various ways.

In the future I would like to work more on accuracy for classifier. Also, there is potential to implement extra features to improve tracking and reduce errors. Requirements are respectively high. Saying that would like to point out difficulty of extraction of ROI various backgrounds. Namely, because of motion detection applies to all object that are moving, hence is difficult to differentiate from actual object of users' interest and object that are just moving but are not targeted by user. For reference please see figure below attached.

Keeping in mind my research about thermal camera. Still requires from me to do more reading and research as well as studies on how to process this type of images. Finally, I would like to implement this system and attach the current developed system to a drone to evaluate in real word scenario.

List of references:

Barba-Guamán, L., Calderon-Cordova C., Quezada-Sarmiento P. (2017) 'Detection of moving objects through color thresholding', IEEE, 10.23919/CISTI.2017.7975755, Portugal.

Begga R., Kamruzzamanb J. (2004) 'A machine learning approach for automated recognition of movement patterns using basic, kinetic and kinematic gait data', *Journal of Biomechanics*, 38, pp: 401–408, Elsevier - Melbourne.

Bradski, G. and Kaehler, A. (2008) *Learning OpenCV: Computer Vision with the OpenCV Library.* Gravenstein Highway North: O'Reilly Media.

Davies A., Velastin S., (2005) 'A Progress review of intelligent CCTV Surveillance Systems', *IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, 0-7803-9446-1/05, Sofia.

Department for Environment Food & Rural Affairs, (2017) 'Fly-tipping statistics for England, 2016/17' Available from: <https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/652958/Flytipping_201617_statistical_release_FINAL.pdf > [Accessed 13 January 2018]

Guennouni S., Ahaitouf A., Mansouri A. (2015) 'Multiple object detection using OpenCV on an embedded platform', *IEEE*, 10.1109/CIST.2014.7016649, Morocco.

Hossain N., Tanzir Kabir M., Rahman T., Hossen M., Salauddin F. (2015) 'A Real-time Surveillance Mini-rover Based on OpenCV-Python-JAVA Using Raspberry Pi 2', *IEEE*, 10.1109/ICCSCE.2015.7482232, Malaysia.

Howse J. (2014) 'Training detectors and recognizers in python and OpenCV', *ISMAR*, Proc. of ECCV

Lee H., Choi J., Jeon E., Kim Y., Le T., Shin K., Lee H., Park K. (2015) 'Robust Pedestrian Detection by Combining Visible and Thermal Infrared Cameras', *MDPI - Sensors*, DOI:10.3390/s150510580.

Leira F., Johansen T. (2015) 'Automatic detection, classification and tracking of objects in the ocean surface from UAVs using a thermal camera', *IEEE*, 10.1109/AERO.2015.7119238, USA.

Leykin A., Hammoud R. (2008) 'Pedestrian tracking by fusion of thermal-visible surveillance videos', *Machine Vision and Application,* DOI: 10.1007/s00138-008-0176-5.

Lima J., Farias T., Apolinário E., Moura G., Silva D., Teichrieb V., Kelner J. (2007) 'Real-time pattern recognition using the opencv library', *Symposium on virtual and augmented reality,* Hilton Charlotte Center City.

Macnish K. (2012) 'Unblinking eyes: the ethics of automating surveillance', *Ethics Inf Technol,* 14:151–16, DOI 10.1007/s10676-012-9291-0.

Manlises C., Martinez J., Belenzo J., Perez C., Khristina M., Postrero T. (2015) 'Real-Time Integrated CCTV Using Face and Pedestrian Detection Image Processing Algorithm For Automatic Traffic Light Transitions', *IEEE*, 10.1109/HNICEM.2015.7393205, Philippines.

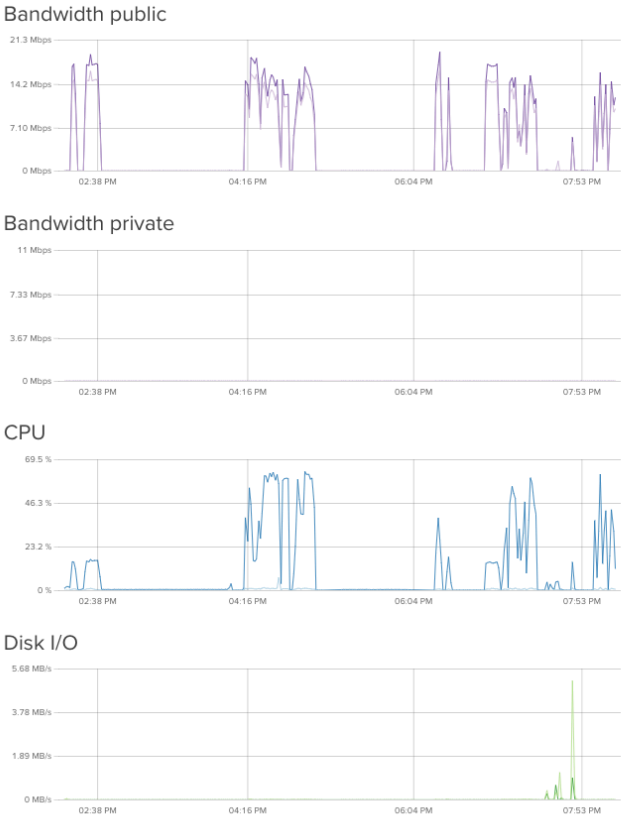
Mehran R., Oyama A., Shah M. (2009) 'Abnormal Crowd Behaviour Detection using Social Force Model', *IEEE*, 978-1-4244-3991-1/09, Florida.

Rodriguez M., Sivic J., Laptev I., Audibert J. (2011) 'Data-driven Crowd Analysis in Videos', *International Conference on Computer Vision,* Imagine, LIGM, UniversiteParis-Est.

Sjarif N., Shamsuddin S., Hashim S., Yuhaniz S., (2014) 'Crowd Analysis and Its Applications', *Soft Computing Research Group*, Universiti Teknologi Malaysia, Skudai, Johor.

APPENDICES

APPENDIX 1 – Server performance output



APPENDIX 2 - Server logs output

```
378
396
378
396
379
398
380
398
378
393
2018-04-24 20:04:55,678 : INFO : Dist between current tracker and new found ROI. 176.705970471 . Add new tracker
^CException in thread Thread-3 (most likely raised during interpreter shutdown):
Traceback (most recent call last):
  File "/usr/lib/python2.7/threading.py", line 801, in __bootstrap_inner
  File "/usr/lib/python2.7/threading.py", line 754, in run
  File "/usr/lib/python2.7/SocketServer.py", line 599, in process_request_thread
  File "/root/.virtualenvs/sub/local/lib/python2.7/site-packages/werkzeug/serving.py", line 621, in handle_error
<type 'exceptions.AttributeError'>: 'NoneType' object has no attribute 'handle_error'
(sub) root@ubuntu:~/imageAnalysis/main# ^C
(sub) root@ubuntu:~/imageAnalysis/main# python2 app.py
2018-04-24 20:05:39,176 : DEBUG : Started up analysis app
2018-04-24 20:05:39,182 : INFO :  * Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
2018-04-24 20:05:39,182 : INFO :  * Restarting with stat
2018-04-24 20:05:39,547 : DEBUG : Started up analysis app
2018-04-24 20:05:39,552 : WARNING :  * Debugger is active!
2018-04-24 20:05:39,552 : INFO :  * Debugger PIN: 301-649-881
2018-04-24 20:05:40,221 : INFO : 158.94.126.195 - - [24/Apr/2018 20:05:40] "GET /vision HTTP/1.1" 200 -
2018-04-24 20:05:40,304 : DEBUG : Successfully created db. databases/dataCollection
2018-04-24 20:05:40,355 : INFO : Successfully opened a stream
2018-04-24 20:05:40,448 : INFO : 158.94.126.195 - - [24/Apr/2018 20:05:40] "GET /motion_detection HTTP/1.1" 200 -
2018-04-24 20:05:42,038 : INFO : 158.94.126.195 - - [24/Apr/2018 20:05:42] "GET /_apiQueryTracking?apiQ0=true HTTP/1.1" 200 -
2018-04-24 20:05:42,069 : DEBUG : Tracking enabled
2018-04-24 20:05:50,453 : INFO : Track failure reported for tracker: 0
2018-04-24 20:05:50,666 : INFO : Dist between current tracker and new found ROI. 312.605182299 . Add new tracker
2018-04-24 20:05:51,627 : INFO : Found matching patter 1301.86610996 . Tracker 0
2018-04-24 20:05:51,668 : INFO : Found matching patter 1301.86610996 . Tracker 0
2018-04-24 20:05:53,257 : INFO : Found matching patter 652.88138048 . Tracker 0
2018-04-24 20:05:53,465 : DEBUG : Successfully saved data to main db.
2018-04-24 20:05:53,465 : INFO : Tracker deleted 0
2018-04-24 20:05:53,465 : INFO : Dist between current tracker and new found ROI. 354.970421303 . Add new tracker
2018-04-24 20:05:53,837 : INFO : Dist between current tracker and new found ROI. 284.397608991 . Add new tracker
2018-04-24 20:05:56,297 : INFO : Track failure reported for tracker: 3
```

APPENDIX 3 – User guide (installation)

Requirements:

- Linux based system
- Python2
- Pip2

1) Following guide and install virtual environment for pip (python2):
https://gist.github.com/Geoyi/d9fab4f609e9f75941946be45000632b

2) Then, copy/clone the source code, which can be found either on repository:
https://github.com/jarzab3/imageAnalysis/tree/submission
Or attached to this document
3) Go to the directory where is source code.
4) Run command:
   pip2 install -r requirements.txt
5) sudo python2 main/app.py
6) Next vision localhost/vision
7) Input username: super and password: superpass

Optionally visit:
jarzebak.eu/vision