WildTracker

Final System Design

Prepared by Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Alexander Santander & Tim Vu, WT18G4-GUI-OD

Checked by Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Delenn Palmer, WT18G4-PM

Approved by Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Delenn Palmer, WT18G4-PM

Authorised for use by Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Felipe Gonzalez, Supervisor

Queensland University of Technology

Gardens Point Campus

Brisbane, Australia, 4001.

This document is Copyright 2018 by the QUT. The content of this document, except that information which is in the public domain, is the proprietary property of the QUT and shall not be disclosed or reproduced in part or in whole other than for the purpose for which it has been prepared without the express permission of the QUT

…/10/2018

…/10/2018

…/10/2018

…/10/2018

**Revision Record**

|  |  |  |  |
| --- | --- | --- | --- |
| Document Issue/Revision Status | **Description of Change** | **Date** | **Approved** |
| 1.0 |  | …/11/18 | Delenn |

**Table of Contents**

**Paragraph Page No.**

[1 Introduction 5](#_Toc528175529)

[1.1 Scope 5](#_Toc528175530)

[2 Reference Documents 6](#_Toc528175531)

[2.1 QUT Systems Engineering Documents 6](#_Toc528175532)

[2.2 Numbering Scheme 6](#_Toc528175533)

[3 System Introduction 7](#_Toc528175534)

[4 System Architecture 8](#_Toc528175535)

[4.1 Subsystem Architecture – Graphical User Interface 8](#_Toc528175536)

[4.2 Subsystem Architecture – Object Detection 9](#_Toc528175537)

[5 System Design 10](#_Toc528175538)

[5.1 Subsystem Design – Graphical User Interface 10](#_Toc528175539)

[5.1.1 Subsystem Requirements 10](#_Toc528175540)

[5.1.2 Software Choice 10](#_Toc528175541)

[5.2 Subsystem Design – Object Detection 11](#_Toc528175542)

[5.2.1 Subsystem Requirements 11](#_Toc528175543)

[5.2.2 Software Choice 11](#_Toc528175544)

[5.2.3 Software Flow 12](#_Toc528175545)

[6 Conclusion 13](#_Toc528175546)

[7 Appendices 14](#_Toc528175547)

[7.1 Pseudo Code – Graphical User Interface 14](#_Toc528175548)

[7.2 Pseudo Code – Object Detections 14](#_Toc528175549)

**Definitions**

|  |  |
| --- | --- |
| WildTracker (WT) | Software implementation of an autonomous detection and tracking tool for wildlife |
| QUT | Queensland University of Technology |
| HLO | High Level Objectives |
| PMP | Project Management Plan |
| GUI | Graphical User Interface |

# Introduction

## Scope

# Reference Documents

## QUT Systems Engineering Documents

|  |  |  |
| --- | --- | --- |
| RD/1 | WT18G4-SUP-Customer Needs | Autonomous detection and tracking tool for wildlife |
| RD/2 | WT18G4-SR-01 | WildTracker Project: System Requirements Document 2018 |
| RD/3 | WT18G4-PMP-04 | WildTracker Project: Project Management Plan 2018 |
| RD/4 | WT18G4-ICD-01 | WildTracker Project: Interface Control Document 2018 |

## Numbering Scheme

For ease of identification, a numbering system has been developed.

For the requirement REQ-M-01:

REQ – This is a requirement derived from the client's brief and the associated HLOs

M – Denotes a *mandatory* requirement, whereas D denotes the *desired* requirement.

# Systems Introduction

The WildTracker system processes wildlife footage to detect and identify the animals. To facilitate this process, a GUI is used. The GUI must have the ability to load a video, play the video, analyse the video and display relevant information in an enclosed pane. This must all be done in accordance with the System Requirements (REQ-M-05, REQ-M-06, REQ-M-07). Without the GUI, the output from the machine learning video analysis would not be able to be displayed and the project would not be able to meet the customer requirements. This makes the GUI subsystem critical to the WildTracker project.

Once the footage has been acquired it must be analysed. This is performed in the backend processing and manipulation of data in conjunction with the popular object detection tool Darkflow. Through the various methods at the disposal of the machine learning team the model must accurately depict the number, species and size of every animal for each frame of entirely new footage which the model was not trained on. This was all done in accordance with the System Requirements.

# System Architecture

## Subsystem Architecture – Graphical User Interface

Although there was little variety in the method to implement the GUI, the one that was highlighted, PyQt GUI Framework alongside QtCreator, proved to be sufficient to implement what was required. PyQt’s framework provided the necessary base upon which the GUI could be built.

Before active development began on the GUI, we had planned to include two video players within the GUI, one for viewing the machine-learning-processed video and one to play the original. Additionally, a seek bar was planned to be added for ease of navigation in long videos. However, upon review of the system requirements it was determined that the two video players were unnecessary and that the additional video player to play the original video only served to clutter the GUI. Also, the seek bar was also not included due to the revelation that the datasets which would be analysed were going to be short and would not make use of the seek bar. With the removal of these features the motivation for the GUI was clear: the GUI was to be as minimalistic as possible to reduce clutter and allow for ease of use.

With the idea for the GUI set, work began on the design and creation of the GUI in PyQt. The first element of the GUI that was implemented was the video player and it’s accompanying controls. This involved creating a custom video player widget as there was no built-in video player widget in QtCreator. Once this was completed, buttons to seek forward, backward, pause and play the video were added. With the video player GUI element complete, buttons to import the original video and analyse this video were added. Within the code to analyse the video was the Object Detection python programming. Alongside these buttons are two textbox elements which display the path to the original video selected and the processed video. The last part critical of the GUI that was implemented was a label to display the number of animals in the current analysed video frame and a list box which would display a list of the animals and their sizes. The list box and label used information that is fed in from the Object Detection subsystem to display the relevant information. Information from the Object Detection subsystem was also used to implement a progress bar that would display the progress that was being made on the video analysis.

## Subsystem Architecture – Object Detection

In the initial stages of this project many avenues were researched and presented to the client regarding how to implement machine learning to achieve the desired outcome. Each strategy had its own merits but ultimately, the widely acclaimed Darkflow system along with its pretrained weights was decided upon. Through the implementation of this model and the weights that accompany it the dataset could now be trained.

The “COCO” dataset was initially planned for to train the model due to its ease of access. However, after some testing the detection was found to be substandard and so an entirely different approach was taken. A script was written (see section 7 Appendices) which simply extracted frames from videos. Knowing that the format of the client’s video would be a bird’s eye view with medium elevation, videos with similar footage were researched. Once found they were manually annotated (see 5.2.2 software choice).

In the early stages of development variables that would later be fed through by the GUI (see section 4.1 above) were hard coded in, such as the filename, destination and threshold values. From here the Darkflow model was created and various video reading and writing objects were made. One such object was used to set the video codec, framerate and resolution.

From here the raw video frames were iterated through, threading each one within the Darkflow model, thereby detecting the relevant objects in each frame. Trusting the model to accurately return whether the object was an elephant or horse the appropriate size estimation function (see section 7 Appendices) was applied. For added flexibility the key “elevation” variable in these functions is fed through the GUI and can be manually changed depending upon the elevation of the current footage. It then returns whether the object is large or small and writes the current completed frame to the exported video destination and begins the loop again.

In order for this code to interact with the front end variables such as what each object is in each frame, the number of objects, the size of the object, the current frame being processed and total number of frames were extracted to a.JSON data file. For each iteration of the loop the GUI read the .json file, thereby updating its various elements. One such example is some simple math used to calculate the export percentage completion and illustrated a progress bar in the GUI.

# System Design

## Subsystem Design – Graphical User Interface

### Subsystem Requirements

GUI design is not particularly intensive on its own. It does not require any specialist hardware and can be performed on most modern PCs and laptops. In this project the GUI was mostly designed on a low-end laptop without any dedicated graphics.

### Software Choice

#### Python

The primary choice of software language for the GUI subsystem was Python. Python is a high-level, easy to deploy and readable dynamic programming language that allows multiple programming paradigms to be used. Some of the main advantages of using python are the relative ease of use when compared to other Object Orientated (OO) programming languages such as C++, Java, and also its huge popularity. Thanks to its popularity, a large number of frequently used and well documented frameworks were available for GUI design, with the PyQt framework being one of those. As of writing this the current version of Python being used is v3.6.

#### PyQt and QtCreator

The PyQt framework and QtCreator were used alongside Python to develop the GUI. PyQt is a GUI framework that provides an API to easily access GUI elements from within Python. QtCreator provided a user interface for PyQt which allows for placing the GUI elements in a .ui file which would later be converted to PyQt API calls in a .py file.

#### Anaconda

Anaconda provides a free an open source Python distribution, and through the Anaconda Prompt allows users to easily manage installed packages and user environments. By using simple command line inputs, a user can install and manage all necessary packages to run the application on their own.

## Subsystem Design – Object Detection

### Subsystem Requirements

Machine Learning (ML) is the process of training an algorithm to generate models which can apply their newly learned classification properties to other data. However, none of this can be achieved without the necessary hardware. TensorFlow has designed their software for optimal efficiency with Nvidia GPU’s, two of our members were in possession of a high-end model which was ideal for training our algorithm.

### Software Choice

#### Python

The primary choice of software language for the OD subsystem was Python. Python is a high-level, easy to deploy and readable dynamic programming language that allows multiple programming paradigms to be used. Some of the main advantages of using python are the relative ease of use when compared to other Object Orientated (OO) programming languages such as C++, Java, and also its huge popularity. Additionally, Python was found to be the best way to streamline integration because all systems were collectively chosen to be written in it. As of writing this the current version of Python being used is v3.6.

#### OpenCV and TensorFlow

The TensorFlow and OpenCV libraries were used alongside Python to develop the OD subsystem. The Python OpenCV (Open Computer Vision) library is an open source library developed by intel, allowing for the pre and post processing of drone footage in conjunction with the NN classifiers provided by TensorFlow.

#### Anaconda

Anaconda provides a free an open source Python distribution, and through the Anaconda Prompt allows users to easily manage installed packages and user environments. By using simple command line inputs, a user can install and manage all necessary packages to run the application on their own.

#### Darkflow

Darkflow is an “out of the box” tool used for YOLO object detection. The model comes with pretrained weights for either the COCO dataset or VOC PASCAL. With very simple installation and execution it is a very user friendly and effective modern tool for object detection.

#### LabelImg

A simple desktop tool with the purpose of loading images within its interface. The user manually clicks and drags to create bounding boxes with labels to specify what and where the model is training within the image. The annotations are stored in an .XML file to be read in conjunction with the original image by the model.

### Software Flow

The weights implemented in our “Darkflow” model use an entirely convolutional approach making them perfect because it is by nature an image classification algorithm, suiting our needs perfectly. Convolutional Neural Networks (CNN) are favorable because they require very little preprocessing compared to other modern algorithms. This is because they take advantage of the inherent properties of images such as the local spatial coherence and ordered inputs. Through this as well as many other avenues the CNN essentially learns filters that would usually be hand crafted, resulting in a much more streamlined process. Figure 1 illustrates the overall structure of the algorithm.

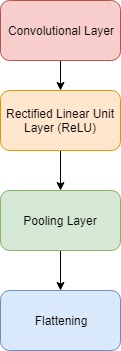


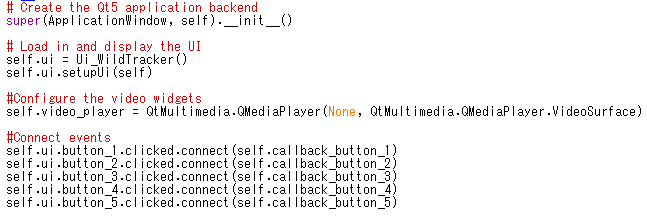
Figure 1

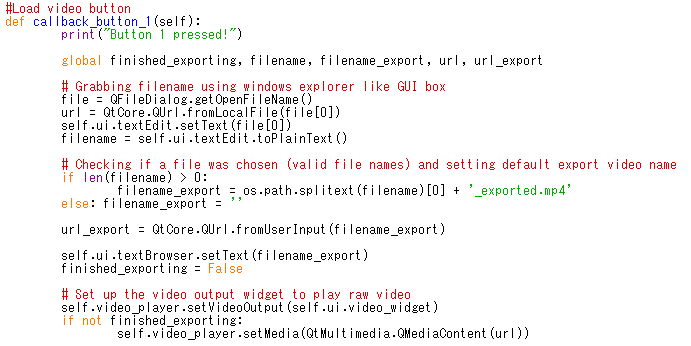
# Conclusion

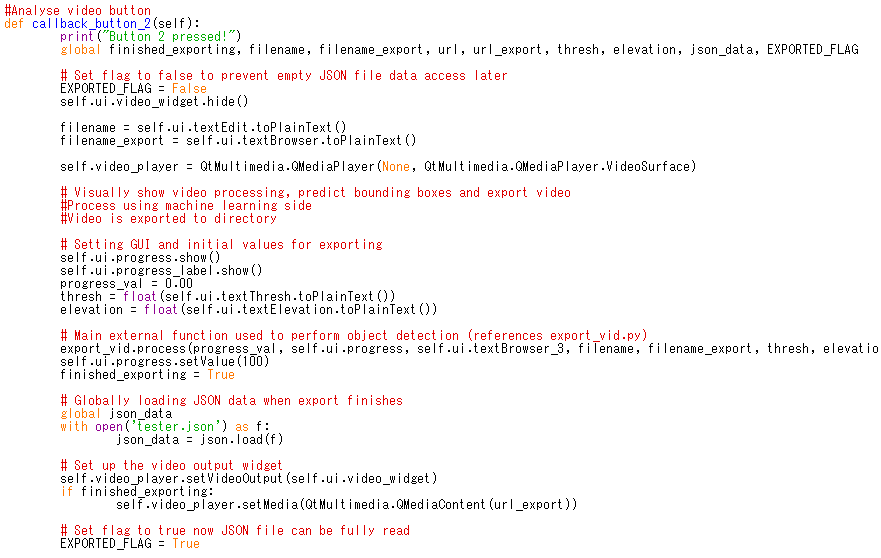
# Appendices

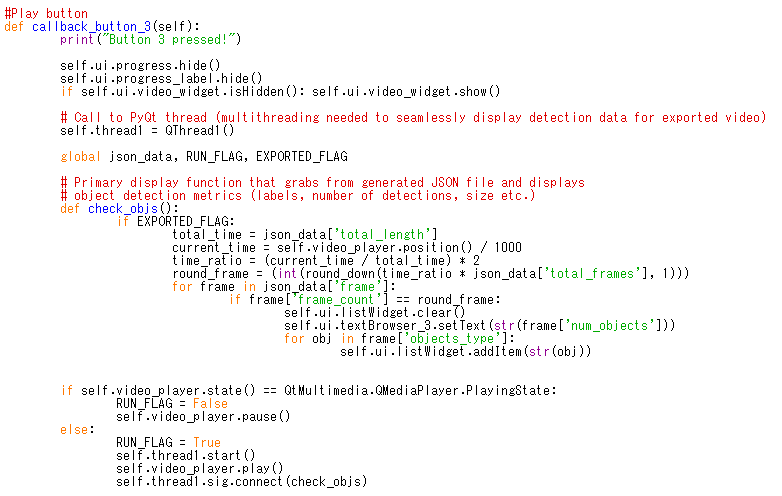
## Pseudo Code – Graphical User Interface

Set up function

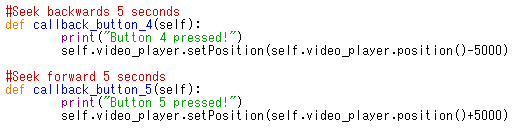
Load video button

Analyse video button

Play/Pause button



Seek buttons



## Pseudo Code – Object Detections