SURVEILIA

Memorandum

To: Kimberly Lemieux
CC: Mel Dundas, James Van Oort, Justin Curran
From: Surveilia Team
Date: October 15, 2021
Re: Surveilia Progress Report

SUMMARY

Surveilia is making strong strides in achieving the outcomes that we have laid out for completion of the drone. We have just finished the final print and assembly of our drone body, the motor system successfully passed operational and control testing, and we are planning the first controlled flight. As well, using OpenCV AI software, our human recognition software is now able to detect people entering the frame of the camera. The focus of our effort is now on finishing the circuit board design, finishing the ground station software, and achieving our first controlled flight. We anticipate that Surveilia will be complete in the first week of December, allowing us time to ensure the product is ready for presentation at the Camosun College Capstone Symposium.

BACKGROUND

The security industry is undergoing a period of high demand with a workforce that lacks appropriate resources to provide services. Much of the problem is driven by supply and demand. However, intrinsic limitations exist in the available resources. Guards are capable of intellectual tasks, but can become fatigued and face reduced awareness. Cameras can provide ongoing and recorded observation, but only in a static area and need some control measure to review observations.

Surveilia is an autonomous security drone that we designed to supplement the needs of the security industry by providing a tool to security companies or residences to augment their protective capabilities. Security guards and cameras are the two primary services that we are seeking to supplement. For concerns about finances, please see *Appendix A* for details.

We built Surveilia to compensate for the weaknesses in both strategies by autonomously patrolling and observing areas specified by the drone operator. By December 10th, Surveilia will be able to

- autonomously patrol a site
- report human activity with artificial intelligence

HARDWARE

FRAME

Our design and hardware team has completed the CAD drawings and manufacturing for the drone body. The manufacturing prototype was printed using polylactic acid (PLA) with a 20% infill. The intention of the prototype was to ensure that all specifications matched our hardware enclosure requirements and that the body had sufficient mechanical integrity for our components.

Figure 1 Print and Assembly of PLA Prototype

We have printed and assembled the final version of the drone body, which is printed using polyethylene terephthalate glycol (PETG) with a 60% infill. However, reprints of some components are required due to the technical complexity of printing with PETG filament.

FLIGHT CONTROLLER HARDWARE

We have received the Kakute F7 flight controller (FC) and are currently using it to test the functionality of the drone. We are testing Mission Planner from ArduPilot in conjunction with the electronic-speed-controller (ESC). We are currently trying to control the drone manually to ensure that all firmware and components are properly installed.

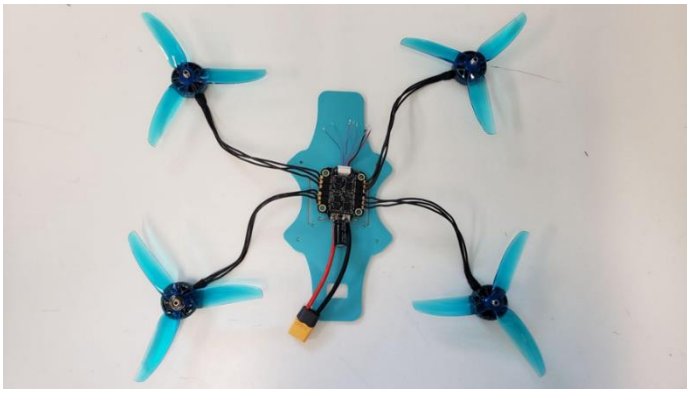

Figure 2 Drive System Assembly

Setting up the manual testing has been a challenge for us. The remote controller was failing to send signals to all the motors, despite the controller and F7 being paired. The issue stems from configurations inside of the ArduPilot interface that is preventing us from accessing the pin that we have connected to the controller receiver. The connectivity issue will be solved no later than October 17.

COMPUTE MODULE 4

The Compute Module 4 (CM4) is our flight computer that is responsible for communicating with the ground station and operating inflight software. The CM4 provides us the ability to take advantage of the power of the Raspberry Pi 4, while reducing weight and meeting our PCB design requirements.

Figure 3 Compute Module 4 [1].

The original source files contain many components that are unnecessary for our project. The size of the board also extends beyond the capacity of our drone casing. We are removing all unused connectors, reducing board size, maintaining power and camera circuits, and adding a Wi-Fi antenna to maximize range of the CM4.

FIRMWARE

ARDUPILOT

ArduPilot has been flashed to the Kakute F7 flight controller and interfaces with the Mission Planner software. However, the file containing the ArduPilot firmware for the F7 is a binary file. Because the firmware is contained within a binary file, we cannot manually alter the code to create a communication protocol between the flight controller and flight computer. The immediate resolution is to operate the computer and controller asynchronously while we research potential firmware alternatives.

Asynchronous operation will look different from our intended plan of synchronous flight. The flight computer will be responsible for running the human recognition and communicating with the ground station, but it will not be able to give guidance to the flight controller. To give the flight computer opportunities to inspect areas for people, Mission Planner will be given altered waypoints designed to give the camera a wider viewing area.

SOFTWARE

AI

We're collecting video using an 8-megapixel camera that is connected to our CM4 board. Our program recognizes potential intruders by collecting all the frames from the live video feed and checking each frame for humans. It does this by passing frames into a pre-built algorithm from OpenCV which is a real-time optimized Computer Vision library [2]. This algorithm processes each frame individually and uses pre-trained AI (agent) to check each processed frame for a person. If the agent finds a person in the frame, it returns an image of the frame with a green rectangle encompassing them, the total person count, and the agent's confidence of its accuracy.

Figure 4 AI Capturing Image of Kaden and Chase

GROUND STATION SOFTWARE

The Surveilia Ground Station is a Graphical User Interface (GUI) that displays relevant flight data in graphical form. The GUI displays data, such as average flight time, on graphs with the current value easily readable. It is also the communication center for the flight computer. The flight computer will send alerts to the ground station if the AI views a person entering the camera frame.

The Ground Station is undergoing a complex overhaul using Model-View-ViewModel (MVVM) abstractions to ensure that the underlying layers of the software communicate in an efficient and precise manner. By implementing MVVM, we can more reliably communicate with the flight computer and store information effectively.

FILE STORAGE

The file storage algorithm (*See Appendix F for flow chart)* is a subset of the Ground Station software. We are developing a file storage algorithm to ensure that data is not lost during restart of the application. The algorithm ensures that we store data returned from the drone for documentation and completes parsing to the Ground Station upon startup. The algorithm works by retrieving data points every thirty seconds and then storing it to a file line-by-line. Each line is counted and stored to the start of same file, so the software knows how many data points to cycle. The storage system is an abstraction of a packet.

CONCLUSION

The Surveilia team has completed much of the drone prototype and is making good progress towards the finalization of the drone. We are finished printing the final version of the drone frame. We are currently implementing all our hardware components, concluding our front-end and back-end software, and continuing to develop our AI software.

We expect our drone to be capable of conducting autonomous patrols, reporting human activity, and displaying accurate flight data to the ground station by December 10. For any questions, Surveilia can be contacted directly at SurveiliaECET@gmail.com.

APPENDIX A: COST-BENEFIT ANALYSIS

Guards are quite expensive, as most physical security is contractual and fulfilled by third parties. 24-hour security can generate monthly wage costs (before third party fees), approximately between $10,342.08 and $18,144 [1]. Static security cameras, although cost effective relative to guards, cannot provide wide range of observation. Cameras can still be quite expensive and vary in price based on the type of camera installed [3].

Table 1. Table outlining costs of Surveilia relative to other services available.

Camera	Guard	Surveilia
$125 - $450 per camera $500 - $1600 for 4 CCTV cameras	$10,342 - $18,144 per month	$650 onetime fee

APPENDIX B: AI TROUBLE

The AI is very reliable; however, it sometimes recognizes inappropriate objects as people. The recognition issue is likely caused by the histogram of oriented gradients (HOG) descriptor algorithm miscalculating the direction x and y gradients or the magnitude and orientation for some of the pixel values [4].

The AI has substantially more false positives in dimensionally cluttered and/or dark spaces which would imply that our classification algorithm (a support vector machine or SVM) has a limited model performance [4]. This would make sense as the hardware that our software is running on is a Raspberry Pi CM4 which has very low processing power when compared to something like a typical desktop computer.

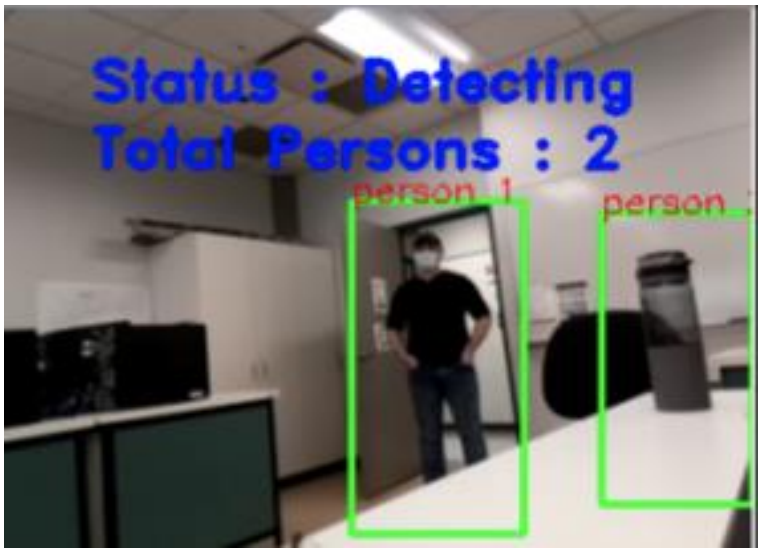

Figure 5 AI Mistaking Water Bottle for a Person.

APPENDIX C: AI CODE

```python
#------------------------------
#Program: human_detection1.0.py
#Author:  Ben Kennedy
#Date:    2021-10-08
#------------------------------

#imports required
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
import imutils
import numpy as np
import argparse

#This function receives a frame from the camera and then checks the frame for people.
#It does this using a histrogram of oriented gradient (HOG) descriptor algorithm to process
#the image and then uses a pre-trained model (support vector machine or SVM) to check for people.
def detect(frame):
    bounding_box_cordinates, weights = HOGCV.detectMultiScale(frame, winStride = (4,4), padding = (8,8), scale = 1.03)
    person = 1
    for x,y,w,h in bounding_box_cordinates:
        cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,0), 2)
        cv2.putText(frame, f'person {person}', (x,y), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,255), 1)
        person += 1

    cv2.putText(frame, 'Status : Detecting ', (40,40), cv2.FONT_HERSHEY_DUPLEX, 0.8, (255,0,0), 2)
    cv2.putText(frame, f'Total Persons : {person-1}', (40,70), cv2.FONT_HERSHEY_DUPLEX, 0.8, (255,0,0), 2)

    # Returns the frame with any detected persons bound by a rectangle
    # The frame is also returned including the total person count and SVM's confidence value
    return frame


# This function is required for capturing frames from the PiCamera
def detectByCamera():

    # Initializes the camera
    camera = PiCamera()
    camera.resolution = (320, 240)
    rawCapture = PiRGBArray(camera, size=(320, 240))

    # Tells the camera to capture video continously while storing the current frame
    # From the array into the image variable. It does this by grabbing the raw NumPy array
    # Representing the image, and then initalizing the timestamp
    for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
        image = frame.array

        # Check the current frame for persons
        detect(image)

        # Stream live video feed on a seperate window
        cv2.imshow("Camera Feed", image)
        key = cv2.waitKey(1) & 0xFF

        # Clear the stream in preparation for the next frame
```

Figure 6 Code Written by Surveilia Team for AI.

APPENDIX D: HARDWARE FLOW CHART

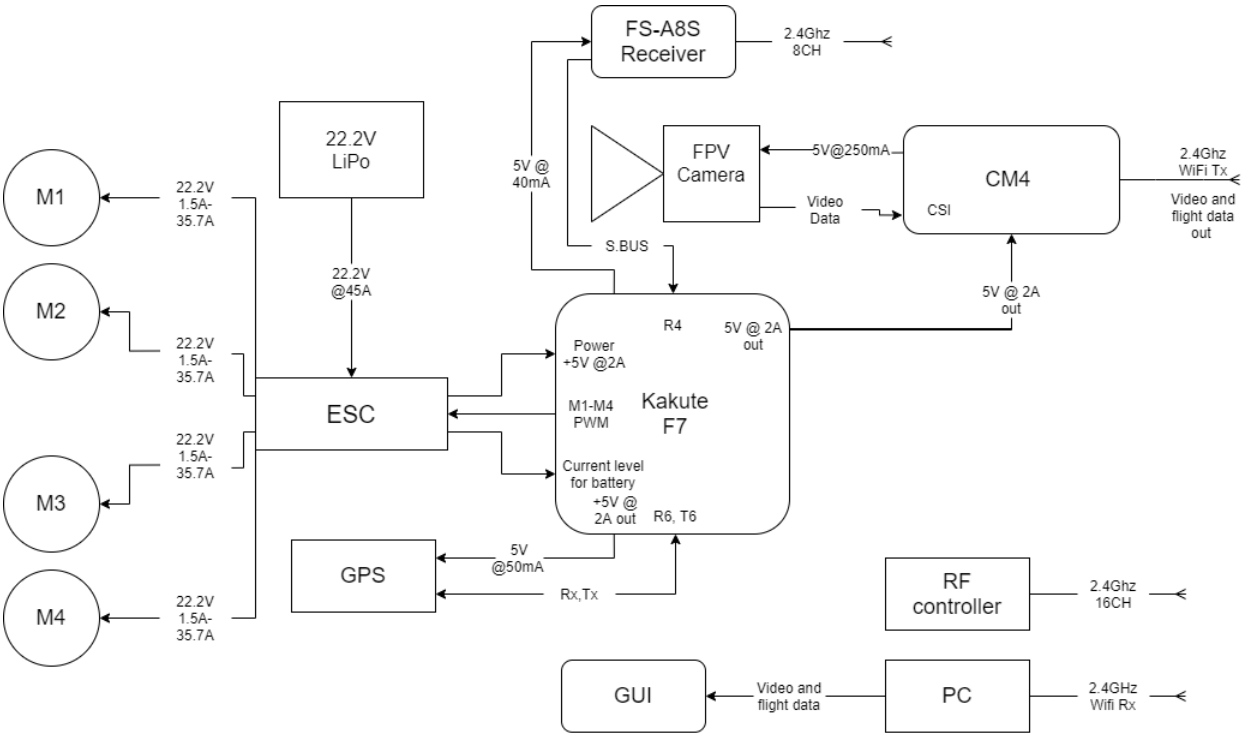

Figure 7 Hardware Flow Chart of Drive and Control Systems.

APPENDIX E: HOG DESCRIPTOR ALGORITHM

We are using a Histogram of Oriented Gradient (HOG) Descriptor algorithm to process each frame and a Support Vector Machine (SVM) to analyze the data for classification.

To understand the algorithm, it is important to understand video file structure. A video file combines a sequence of images to form a moving picture, typically referred to as frames. Once these frames are stored into an array, they can be easily accessed and manipulated [5]. OpenCV [2] has already been implemented in an efficient way to combine the HOG Descriptor algorithm with an SVM, so we are only required to send our array of frames through OpenCV's algorithm.

The first thing our software does is call the pre-model for human detection of OpenCV and then feeds our SVM with it. Whenever a frame is received from the camera, it is passed to OpenCV's algorithm. It then takes the frame and looks for people within the frame. If there are people detected OpenCV returns the frame with the person(s) bound by a green box and provides the person count, and the confidence value of its reliability.

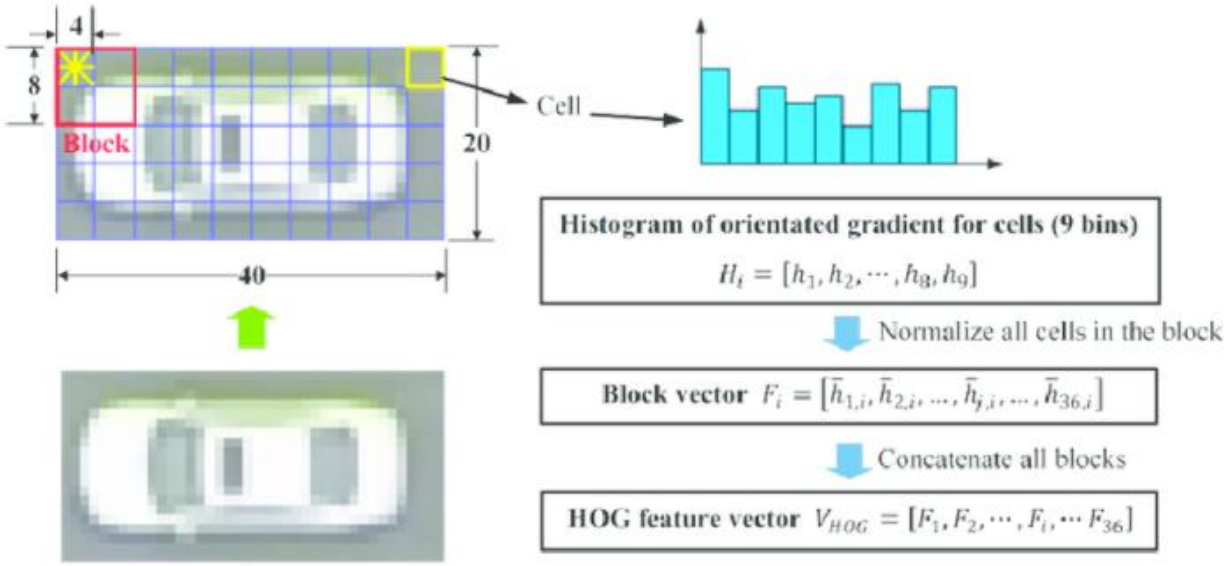

Figure 8 HOG Descriptor Algorithm [4].

APPENDIX F: DATA STORAGE FLOW CHART

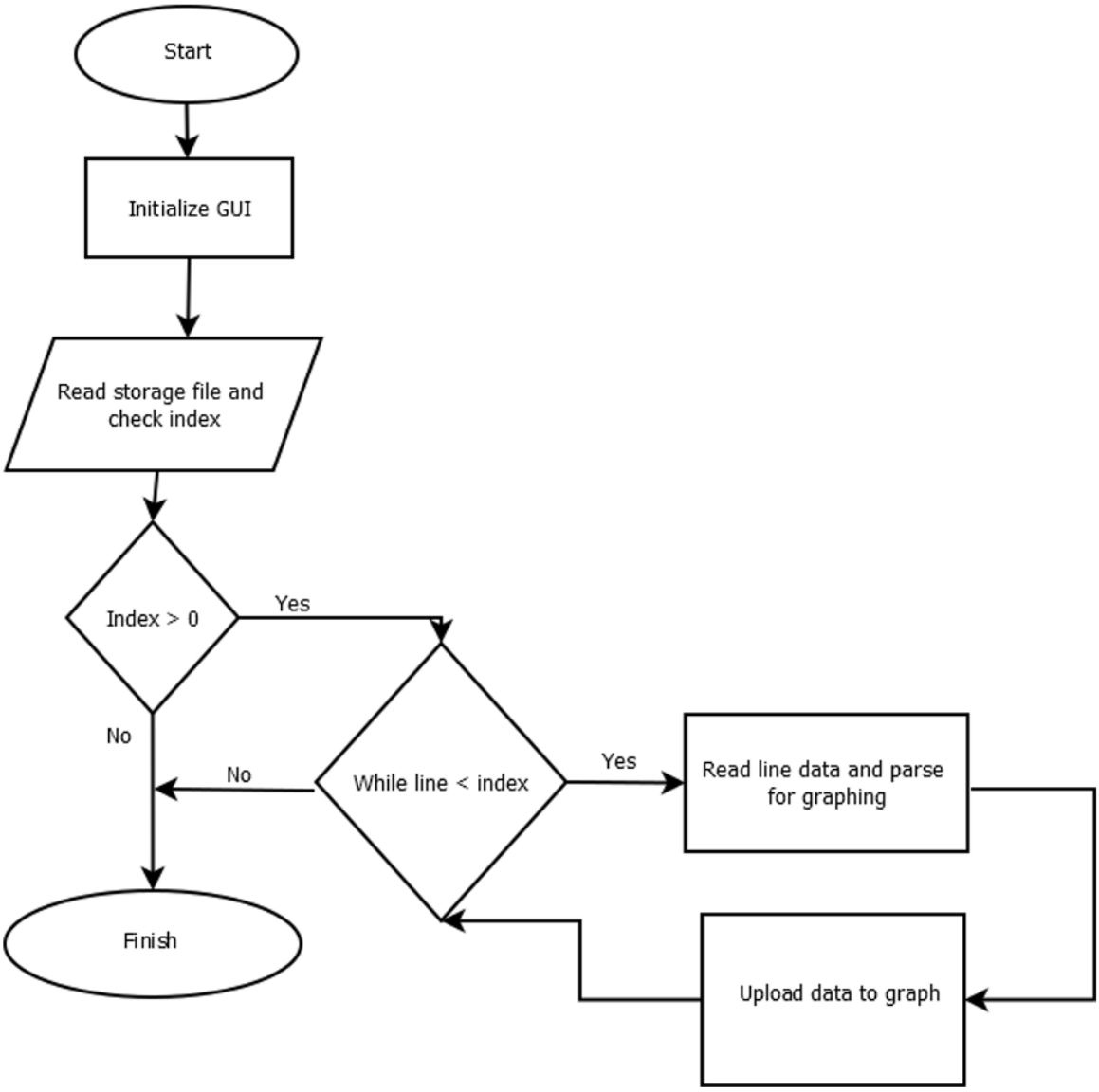

Figure 9 Flow Chart Defines Basic Strategy for Parsing Storage to GUI

REFERENCES

[1] Iottrends, “What is Raspberry Pi Compute Module 4 – A Complete Guide,” 16 Jan, 2021 [Online]. Available:https://www.iottrends.tech/blog/raspberry-pi-compute-module-4-everything-you-need-to-know/ [Accessed: Oct 12, 2021].

[2] A. Rosebrock, “Install OpenCV 4 on your Raspberry Pi,” 26 Sept, 2018. [Online]. Available https://www.pyimagesearch.com/2018/09/26/install-opencv-4-on-your-raspberry-pi/. [Accessed: Sept. 23, 2021].

[3] Paladin Security, “Security Guard Salary in Canada by Province in 2021,” Nov 14,2021. [Online] Available: https://paladinsecurity.com/security-careers/security-guard-salary/. [Accessed: Sept 16, 2021].

[4] Wikipedia, “Histogram of Oriented Gradients,” 01 Feb, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients. [Accessed: September 24, 2021].

[5] Y. Xu, G. Yu, Y, Wang and X. Wu, “A Hybrid Vehicle Detection Method Based on Viola-Jones and HOG + SVM from UAV Images”, ResearchGate, Sensors. 16. 1325., p.6, Aug, 2016. [Online]. Available: https://www.researchgate.net/publication/306333664_A_hybrid_vehicle_detection_method_based_on_viola-jones_and_HOG_SVM_from_UAV_images. [Accessed: Oct 12, 2021].