KERRY LOGISTICS | HACKATHON

23 - 24 FEBRUARY 2019

TEAM HEX: OPERATIONAL GUIDE

KERRY LOGISTICS [1]

Chris KWOK **Alvin LAI**

Rohini BANERJEE **Abhishek PARYANI**

Bikram Aditya GANGWAR (TL)

[1] This logo is a trademark of Kerry Logistics Limited.

CHALLENGE #1: DRONE FLYING COMPETITION

The purpose of this hackathon is to leverage the power of autonomous drones to transform warehouses. For this challenge, we are expected to fly our drones autonomously while reading the location IDs and carton IDs along the flying path. The grading is done based on the following:

(a) Accuracy of identifying, decoding and pairing the carton ID with its corresponding location ID.
(b) Autonomous drone piloting skills, which are inclusive of drone stability, speed and efficiency of the flying path, and effective handling of faults.
(c) Coding efficiency and utilization of Microsoft AI practices.

1. Data Provided

As part of the competition, an aisle is provided with three layers of shelves on either side. The total distance to be travelled along the length of the aisle is about 26ft. With a width of ~2ft in between, each side of the aisle has cartons placed across the three shelves (total height ~170cm).
Multiple QR codes are placed on the edges of the shelves. These correspond to the location IDs. Above each such location QR code, a carton box may or may not be present. If present, it will also contain a QR code. The aim is to detect and decode both QR codes and pair them.

2. Task Breakdown

(a) SDK and framework finalization for autonomous drone flight
(b) Strategizing and creating the flight path
(c) Obtaining a steady video from the drone's flight and storing it on the system
(d) Research, study and analyze computer vision algorithms and APIs to detect and decode QR codes
(e) Efficiently pairing the location and carton QR codes
(f) Integrating entire solution into a compact and usable UI.

3. Disclaimer and Limitations of the Solution

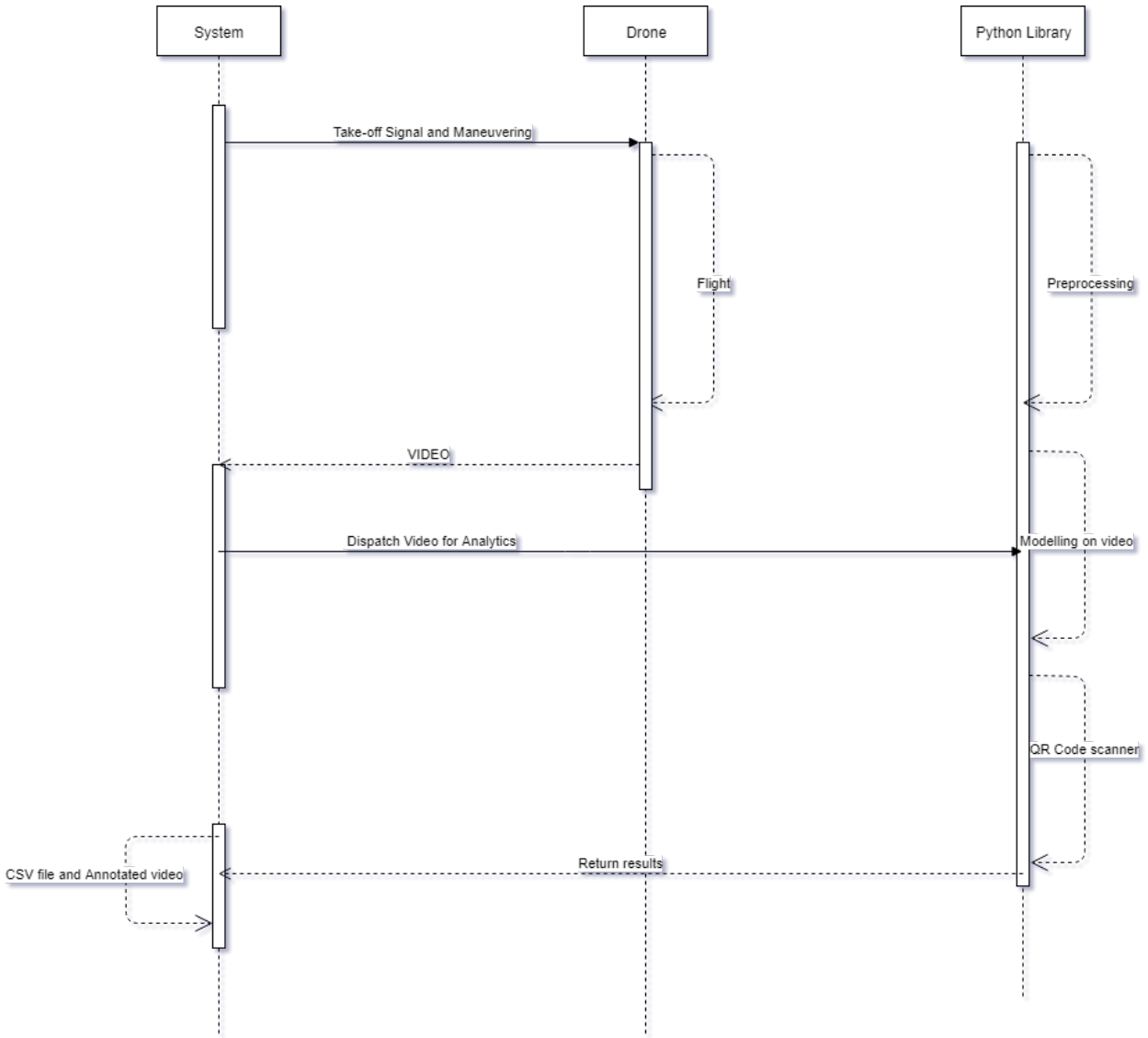
(a) Currently, the flight path parameters are hardcoded according to the provided testing arena requirements.
(b) Though the QR code detection and decoding is done real-time using the Dynamsoft barcode API, we are also detecting the same using python in semi real-time. This is done to pair the location and carton QR codes more effectively.
(c) The speed of the drone should be slow to clearly capture the QR codes (about 80-100 FPS).

(d) There should be ample light to detect the QR codes properly.
(e) We did not use the Azure Computer Vision AI as manually annotating the images would take time. Hence we used pre-trained models using pyzbar library

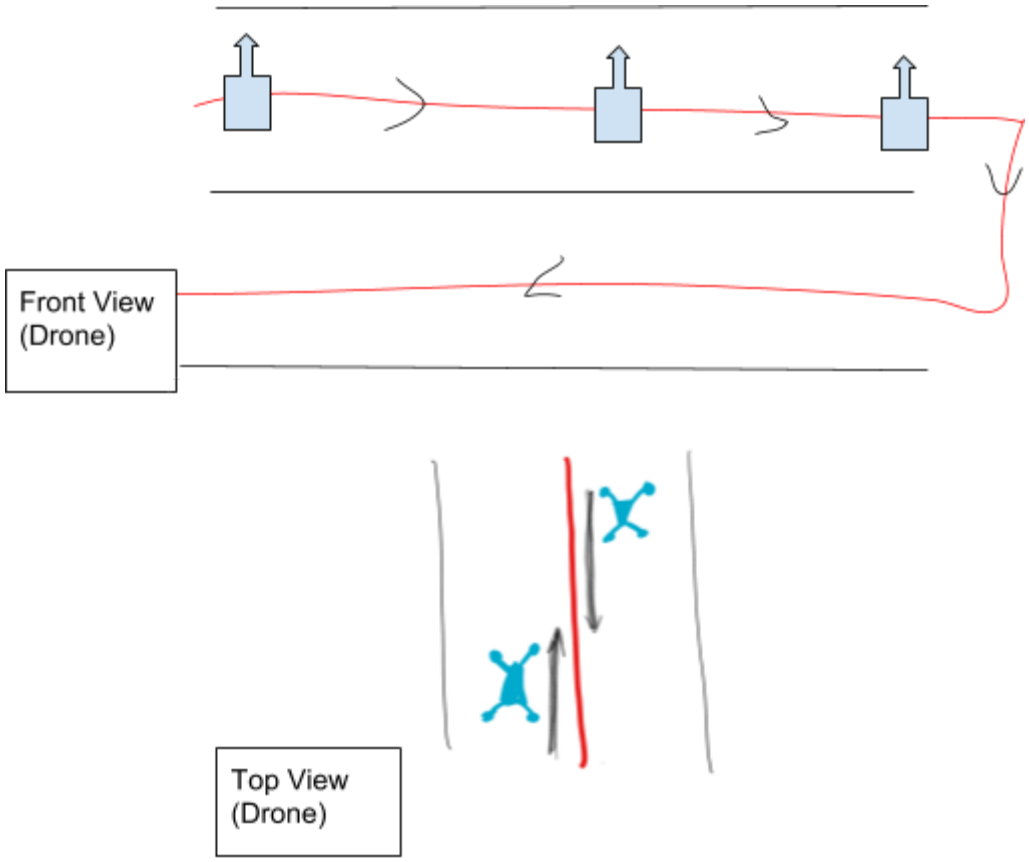
4. Solution

4.1. Drone Race

4.1.1. Workflow



4.1.2. Efficiency of Chosen Path



4.1.3. Drone Flight Path control

(a) Flight path is controlled by PID (Proportional, Integral, Derivative), which was tuned according to the data read from the sensors.
(b) The compass sensor was used to set the direction of the waypoints and the PID tuning helped in setting the rotation speed and distance of the waypoints.
(c) This helped in providing the stability and handling the faults and errors during maneuvering.

4.1.4. Output of the Video Streaming

(a) 6 videos, one for each aisle, in .mp4 format.
(b) Dividing it into 6 videos so that the memory does not crash

4.2. QR Code Detection

4.2.1. Pre-requisites

(a) pip install pyzbar
(b) input video in *.mp4* or *.avi* format

4.2.2. Libraries Used

(a) from __future__ import print_function
(b) import pyzbar.pyzbar as pyzbar
(c) import numpy as np
(d) import cv2
(e) from imutils.video import VideoStream
(f) import argparse
(g) import time
(h) import pandas as pd

4.2.3. Pyzbar: A Python-3 wrapper for ZBar

Pyzbar is a wrapper to read one-dimensional barcodes and QR codes from Python 2 and 3 using the *zbar* library (detailed working mentioned here). It has no dependencies apart from installing *zbar* itself. Built on pure python, this library is fast, accurate and works with PIL / Pillow images, OpenCV / numpy ndarrays, and raw bytes of data.

4.2.4. Algorithm

(a) Take video input from drone SDK (.mp4 format)
(b) Read the video using cv2.VideoCapture()
(c) Read each frame within the video
(d) Crop the frame so that the QR codes are more focused (fine-tuning for each video file to be done manually)
(e) Detect and decode with pyzbar.decode(). Then display the QR code outputs on the screen using cv2.putText() and cv2.imshow().
(f) Check if each frame can detect 2 QR codes or not. If yes, verify that one of them is a Location ID (i.e., starts with 'L') and pair the two QR codes together in the CSV file.
(g) If only Location ID is present in a frame without a carton ID, assume that the box is missing and only update the location QR code information in the CSV file.
(h) Save the final CSV output in required format.

4.2.5. CSV Output Format

(a) **Location ID**: Column containing all the location IDs
(b) **Carton ID**: Decodes the QR Codes on the boxes and lists them next to their corresponding location ID.

5. Edge Cases

1. No detection of empty areas (aisles/rows/columns without cartons)
2. Does not identify mistaken duplicates of QR Codes.

6. Test Cases

1. Fly drone autonomously :
    a. 3 Rows - Each side 13 ft.
    b. 3 Rows - Each side 26 ft.
    c. 3 Rows - 2 Sides - 13 ft.
    d. 3 Rows - 2 Sides - 26 ft.

7. Challenges Faced With Drone and Analytics

1. Code for saving video and sending it to the local system from DJI interface
2. Glare on QR Codes causing detection and conversion issues
3. Limitations of the DJI Mavic Air Drone:
    - FOV for the camera is 120cm(w) x 100cm(h) .
    - Focal length is >50 cm.
4. Rotation of the drone to cover opposite aisles is still an issue with autonomous flight scripting
5. QR Code matching (Aisle/Location ID + Carton ID) is still a persistent issue due to the fact that there is no link pairing the two ID's.

8. Deliverables: Files

Code, CSV file

9. How to Run the Program

C# UWP
For the analytics QR code detection:

(a) keep the input video in the same folder as the two .py codes
(b) Open command prompt and navigate to the proper folder
(c) Run **python QR_video.py --video <video path>**

10. References

a. https://www.learnopencv.com/barcode-and-qr-code-scanner-using-zbar-and-opencv/

b. https://www.pyimagesearch.com/2014/12/15/real-time-barcode-detection-video-python-opencv/
c. https://github.com/cuicaihao/Webcam_QR_Detector
d. https://github.com/VAD3R-95/QR-code/blob/master/Codes.py
e. https://pypi.org/project/pyzbar/
f. http://zbar.sourceforge.net/
g. https://github.com/NaturalHistoryMuseum/pyzbar