Enemy Identifying Aerial Surveillance Vehicle

Development Research Report

Table of Contents

Problem and Significance (Jayanth)

Background Work (Jayanth)

Algorithm (Jayanth)

Strategy (Yousuf)

Implementation (Yousuf)

Results (Both)

Discussion (Yousuf)

Problem and Significance

For long periods of history, great victories have been won on the battlefield through ambush tactics. Famous victories like the Battle of Teutoburg Forest and the Battle of Trenton all were won through ambush tactics. While drones did not exist at those times, they certainly exist now in such a capacity that their abilities could be utilized to protect soldiers and alert them of ambushes beforehand.

Throughout the war in Afghanistan and Iraq, the insurgent enemy often used such ambush tactics to attack convoys and patrols of national or coalition soldiers. Many times, insurgents would set up an IED (improvised explosive device) attack and position several fighters to start shooting as soon as the convoy encountered the IED. An untold number of soldiers were either killed or injured through this tactic.

Several times, battles were lost due to an undetected, overwhelming number of enemy fighters attacking groups of soldiers in vulnerable positions. If these soldiers had access to information that indicated them of at least a partial enemy presence, then perhaps they would be better equipped to defend themselves and defeat the enemy.

While the aforementioned wars are over, the future demands better surveillance and protection. There must be a better system that can identify enemy movements and report them to ground forces such that they have an indication as to the approximate size and location of the enemy force.

Such tactics have been used in the past with targeted operations to ensure that there is minimal resistance while conducting operations. However, they aren’t used as often when it comes to actions like convoys and patrols. Drones would normally be used to provide high level surveillance for raids and operations but not so often when conducting simple security checks and transporting material across enemy lines. Sometimes, the United Nations “Blue Helmets” Peacekeepers have used drones to understand an area’s topography and population before entering it. However, these tactics are not as widespread throughout national militaries and security forces.

This project is focused on introducing drones to common military actions to allow their entire capacity to be diverted to the protection and intelligence given to the soldiers. The primary function these drones will serve is to scan over a convoy’s path or a patrol’s security area and use its camera and object recognition capabilities to identify and locate hidden enemy figures. In addition to that, infrared cameras on the drone can be used to see if there is an IED buried somewhere or hidden heat signatures, indicating fighters waiting to ambush. If such information is made available to soldiers, they will be much better equipped to handle situations with minimal resistance and casualties.

Background Work

Drones have long been used in military capacities to gather intelligence, check for mines, attack enemy positions, and monitor underwater positions. Drones are primarily used in the UAV format, Unmanned Aerial Vehicle, where they provide surveillance and give an eye-in-the-sky to ground soldiers. Recently, drones have also been used more and more in military campaigns. Larger drones have been used more to carry explosives while smaller drones have been used to scout into dangerous areas that may potentially contain insurgent fighters. We, therefore, are interested in researching more into the small, scout variation of the drone as it can provide a vital method of gathering information on the whereabouts of enemy units, especially those in caves and other tight areas.

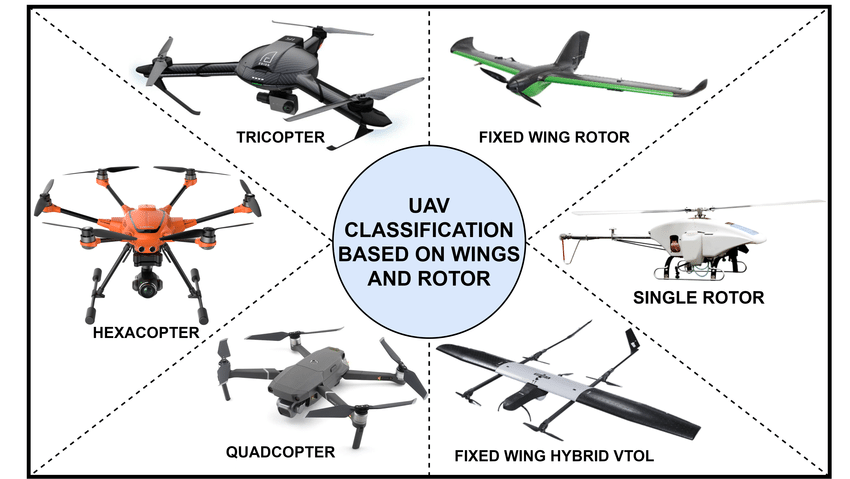
Figure 1



Figure 2

Algorithm

We constructed our algorithm part-by-part such that we understood what each portion of our program is doing. Our algorithm is constructed similar to a normal Python control system program, in that the object that is being controlled is constructed at the top of the program. This object is used to modify the drone’s movements and altitude such that its position is controlled completely by the computer. We then included features that allowed the drone to capture pictures and record videos of its current environment view. This, combined with a window feature. Our algorithm is primarily user controlled. We set up our program such that it initializes the drone state and sets it equal to an object. This makes controlling the drone so much easier as the user is able to input keyboard controls and this directly invokes a function of the drone. This means that less of the implementation is done by the user and the developer and more by the system manufacturer. In addition to this direct initialization and keyboard input, we also included other features such as keyboard controlled movement. This way, the user controls the movement of the drones such that specific areas marked as dangerous can be easily checked.

Strategy

Our strategy for this project was to have the drone move around based on user input and record live footage taken by the front-end camera. The movement based on user input is to simulate the movements that would be controlled by the military users. These movements would be intended to be towards a suspected enemy encampment or along a predetermined path. This path would then be traveled upon as controlled by the ground user. This ground user would use a connected computer to control the drone’s movement. Our strategy for testing was to first connect the used computer to the drone. We then started the program and began using the keyboard controls, WASD, to make the drone fly as we wished. We then also ensured that the drone view opened. This way

Implementation

We ran the below

Camera and manual controls

from djitellopy import tello

import keyPress as kp

import time

import cv2

kp.init()

me = tello.Tello()

me.connect()

print(me.get\_battery())

global img

me.streamon()

def getKeyboardInput():

lr, fb, ud, yv = 0, 0, 0, 0

speed = 50

if kp.getKey("LEFT"): lr = -speed

elif kp.getKey("RIGHT"): lr = speed

if kp.getKey("UP"): fb = speed

elif kp.getKey("DOWN"): fb = -speed

if kp.getKey("w"): ud = speed

elif kp.getKey("s"): ud = -speed

if kp.getKey("a"): yv = -speed

elif kp.getKey("d"): yv = speed

if kp.getKey("q"): me.land();time.sleep(3)

if kp.getKey("e"): me.takeoff()

if kp.getKey('z'):

cv2.imwrite(f'Resources/Images/{time.time()}.jpg’,img')

time.sleep(0.3)

return [lr,fb,ud,yv]

while True:

vals = getKeyboardInput()

me.send\_rc\_control(vals[0, vals[1], vals[2], vals[3]])

img = me.get\_frame\_read().frame

img = cv2.resize(img, (360,240))

cv2.imshow("Image", img)

cv2.waitKey(1)

Face tracking

import cv2

import numpy as np

from djitellopy import tello

import time

me = tello.Tello()

me.connect()

print(me.get\_battery())

me.streamon()

me.takeoff()

me.send\_rc\_control(0, 0, 25, 0)

time.sleep(2.2)

w, h = 360, 240

fbRange = [6200, 6800]

pid = [0.4, 0.4, 0]

pError = 0

def findFace(img):

faceCascade = cv2.CascadeClassifier("OpencvPython/Resources/haarcascade\_frontalface\_default.xml")

imgGray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = faceCascade.detectMultiScale(imgGray, 1.2, 8)

myFaceListC = []

myFaceListArea = []

for (x, y, w ,h) in faces:

cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 2)

cx = x + w // 2

cy = y + h // 2

area = w \* h

cv2.circle(img, (cx, cy), 5, (0, 255, 0), cv2.FILLED)

myFaceListC.append([cx, cy])

myFaceListArea.append(area)

if len(myFaceListArea) != 0:

i = myFaceListArea.index(max(myFaceListArea))

return img, [myFaceListC[i], myFaceListArea[i]]

else:

return img, [[0, 0], 0]

def trackFace( info, w, pid, pError):

area = info[1]

x, y = info[0]

fb = 0

error = x - w // 2

speed = pid[0] \* error + pid[1] \* (error - pError)

speed = int(np.clip(speed, -100, 100))

if area > fbRange[0] and area < fbRange[1]:

fb = 0

elif area > fbRange[1]:

fb = -20

elif area < fbRange[0] and area != 0:

fb = 20

if x == 0:

speed = 0

error = 0

#print(speed, fb)

me.send\_rc\_control(0, fb, 0, speed)

return error

#cap = cv2.VideoCapture(1)

while True:

#\_, img = cap.read()

img = me.get\_frame\_read().frame

img = cv2.resize(img, (w, h))

img, info = findFace(img)

pError = trackFace( info, w, pid, pError)

#print(“Center”, info[0], “Area”, info[1])

cv2.imshow("Output", img)

if cv2.waitkey(1) & 0xFF == ord('q'):

me.land()

break

Results

Discussion

Something is said