```python
# This is the drone control functions to insert to the assembly
import logging
import time
from threading import Timer
import numpy as np
import cv2
from threading import Thread

import camera_functions as cam
import cflib
import cflib.crtp
from cflib.crazyflie import Crazyflie
from cflib.crazyflie.log import LogConfig


class Drone():
    """
    Here we are only using the cameras and the onboard
    control system.
    """

    def __init__(self, link_uri):
        """ Initialises with the specific uri """
        self.loc = [None, None, None]  # location
        self.vel = [None, None, None]  # velocity
        self.loc = np.array(self.loc)
        self.vel = np.array(self.vel)
        # velocity here just mean movement between frames, not actual
        # velocity
        # location and velocity are stored as np arrays.
        self._cf = Crazyflie()
        self.uri = link_uri
        self.target = np.array([0, 0, 0])
        self.loc_prev = None
        self.vel_prev = None
        self.not_found_counter = 0

    def Initialise(self):
        """
        Connect with the drone by the link uri, get initial
        location and velocity.
        """
        cflib.crtp.init_drivers(enable_debug_driver=False)
        self._cf.open_link(self.uri)
        # move the line above to main function for multi drones
        self.cmd = cflib.crazyflie.Commander(self._cf)
        self.cmd.send_setpoint(0, 0, 0, 0)
```

```
48          return self.cmd
49
50      def Go_to(self, target, Commander, Kp=30, Ki=0, Kd=-1500):
51          """ PID controller to get to specific location """
52          if self.not_found_counter > 10:
53              # drone lost
54              Commander.send_setpoint(0, 0, 0, 0)
55              print("shutting down")
56          else:
57              command = (target - self.loc) * Kp + self.vel * Kd
58              pitch = min(max(command[0], -10), 10)
59              roll = - min(max(command[1], -10), 10)
60              thrust = min(max(command[2]*3000, -15000), 5000)+37000
61              thrust = int(thrust)
62              print(self.loc, self.vel, roll, pitch, thrust)
63              self.cmd.send_setpoint(roll, pitch, 0, thrust)
64          return
65
66      def Start_up(self, thrust):
67          self.cmd.send_setpoint(0, 0, 0, thrust)
68
69      def get_loc(self, coordinates, read_failed, loc_prev, vel_prev):
70          """
71          Return a weighted sum of location from the camera
72          and the previous momentum.
73          """
74          cam_coord = np.array(coordinates)
75          if loc_prev.all() == None:
76              return cam_coord, np.array([0, 0, 0])
77          if read_failed[0] == 1:
78              self.not_found_counter += 1
79              return loc_prev+vel_prev, vel_prev
80          self.not_found_counter = 0
81          loc_current = 0.7*cam_coord+0.3*(loc_prev+vel_prev)
82          return loc_current, (loc_current-loc_prev)
83
84
85  def simplified_control(target, link_uri):
86      """
87      The main control function, to be called as a separate thread from the gesture
88      recognition part. This function continuously reads the targets and attemps to
89      move the drone to the target.
90      """
91      if type(link_uri) == str:
92          c0 = [0,0,0]
93          c1 = [0,0,0]
94          read_failed0 = [1]
95          read_failed1 = [1]
96          start_signal = [0]
97          read_failed = [read_failed0, read_failed1]
98          # Initialise
99          camera_Thread = Thread(target=cam.simplified_loop,
100                                 args=([c0,c1], read_failed))
101         camera_Thread.start()
102         cf0 = Drone(link_uri)
103         cmd0 = cf0.Initialise()
104         input("press enter when ready")
105         cf0_Thread = Thread(target=individual_control, args=(
106             target[0], start_signal, cf0, cmd0, c0, read_failed0))
107         cf0_Thread.start()
108         cf0.Start_up(40000)
109         start_signal[0] = 1
110         input("press enter to stop")
111         start_signal[0] = 0
```

```
    if type(link_uri) == list:
        assert len(target) == len(
            link_uri), "Provide exactly one link_uri for each target location"
        c0 = [0,0,0]
        c1 = [0,0,0]
        read_failed0 = [1]
        read_failed1 = [1]
        start_signal = [0]
        read_failed = [read_failed0, read_failed1]
        # Initialise
        camera_Thread = Thread(target=cam.simplified_loop,
                               args=([c0,c1], read_failed))
        camera_Thread.start()
        cf0 = Drone(link_uri[0])
        cf1 = Drone(link_uri[1])
        cmd0 = cf0.Initialise()
        cmd1 = cf1.Initialise()
        input("press enter when ready")
        cf0_Thread = Thread(target=individual_control, args=(
            target[0], start_signal, cf0, cmd0, c0, read_failed0))
        cf1_Thread = Thread(target=individual_control, args=(
            target[1], start_signal, cf1, cmd1, c1, read_failed1))
        cf0_Thread.start()
        cf1_Thread.start()
        cf0.Start_up(40000)
        cf1.Start_up(40000)
        start_signal[0] = 1
        input("press enter to stop")
        start_signal[0] = 0


def individual_control(target, start_signal, cf, cmd,
                       coordinates, read_failed):
    while True:
        time.sleep(0.01)
        if read_failed[0] == 0:
            print("drone found")
            cmd.send_setpoint(0, 0, 0, 0)
            break
    while start_signal[0] == 1:
        # updates the coordinate list from the camera feed
        # updates the drone location and velocity
        print("go to")
        cf.loc, cf.vel = cf.get_loc(
            coordinates, read_failed, loc_prev=cf.loc, vel_prev=cf.vel)
        cf.Go_to(np.array(target), cmd)
        time.sleep(0.01)
    for i in range(5):
        cmd.send_setpoint(0, 0, 0, 20000)
        time.sleep(0.5)


def control(target, link_uri, start_signal):
    """
    The main control function, to be called as a separate thread from the gesture
    recognition part. This function continuously reads the targets and attemps to
    move the drone to the target.
    """
    if type(link_uri) == str:
        coordinates = [0, 0, 0]
        read_failed = [1]
        # Initialise
        camera_Thread = Thread(target=cam.simplified_loop,
                               args=(coordinates, read_failed))
```

```
176             camera_Thread.start()
177             cf = Drone(link_uri)
178             cmd = cf.Initialise()
179             while start_signal[0] == 0:
180                 time.sleep(0.1)
181             cf.Start_up(37500)
182             while True:
183                 time.sleep(0.01)
184                 if read_failed[0] == 0:
185                     # print("drone found")
186                     cmd.send_setpoint(0, 0, 0, 0)
187                     break
188             while start_signal[0] == 1:
189                 # updates the coordinate list from the camera feed
190                 # updates the drone location and velocity
191                 cf.loc, cf.vel = cf.get_loc(
192                     coordinates, read_failed, loc_prev=cf.loc, vel_prev=cf.vel)
193                 cf.Go_to(np.array(target), cmd)
194                 time.sleep(0.01)
195             for i in range(5):
196                 cmd.send_setpoint(0, 0, 0, 20000)
197                 time.sleep(0.5)
198         if type(link_uri) == list:
199             assert len(target) == len(
200                 link_uri), "Provide exactly one link_uri for each target location"
201             c0 = [0,0,0]
202             c1 = [0,0,0]
203             read_failed0 = [1]
204             read_failed1 = [1]
205             read_failed = [read_failed0, read_failed1]
206             # Initialise
207             camera_Thread = Thread(target=cam.simplified_loop,
208                                    args=([c0,c1], read_failed))
209             camera_Thread.start()
210             cf0 = Drone(link_uri[0])
211             cf1 = Drone(link_uri[1])
212             cmd0 = cf0.Initialise()
213             cmd1 = cf1.Initialise()
214             while start_signal[0] == 0:
215                 time.sleep(0.1)
216             cf0_Thread = Thread(target=individual_control, args=(
217                 target[0], start_signal, cf0, cmd0, c0, read_failed0))
218             cf1_Thread = Thread(target=individual_control, args=(
219                 target[0], start_signal, cf1, cmd1, c1, read_failed1))
220             cf0_Thread.start()
221             cf1_Thread.start()
222             cf0.Start_up(40000)
223             cf1.Start_up(40000)
224 
225 
226 if __name__ == '__main__':
227     # simplified_control([0, 0, 0], "radio://0/80/250K")
228     link_uri = ["radio://0/80/250K","radio://0/12/1M"]
229     simplified_control([[0.1, 0.1, -0.1], [-0.15, -0.15, 0.15]], link_uri)
230 ##    link_uri = "radio://0/80/250K"
231 ##    simplified_control([0.1, 0.1, -0.1], link_uri)
```