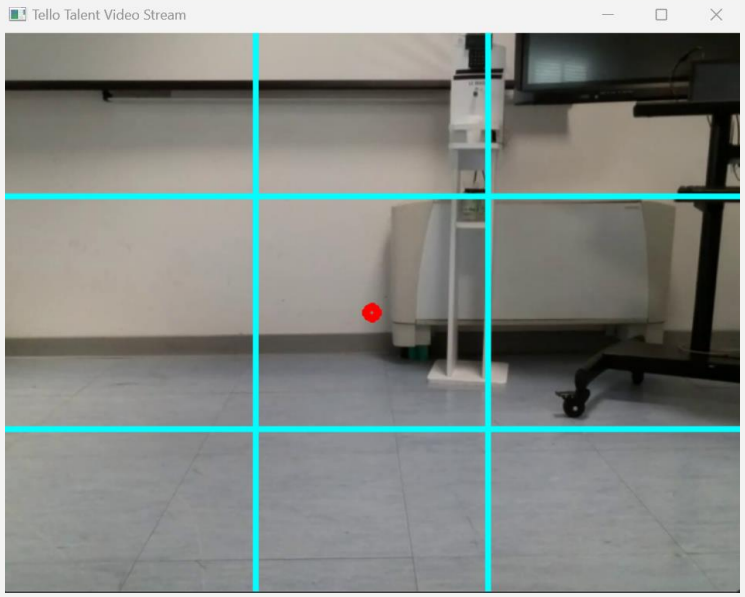
DRONE TRACKING

CAPPIELLO MARCO

2023/2024

PROBLEM

- Given two drones, create a program capable of recognizing the movement of one of the two and aligning with its position in flight.

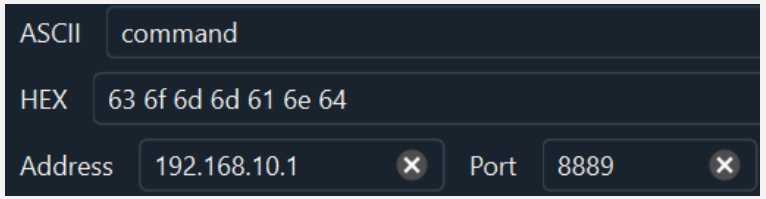


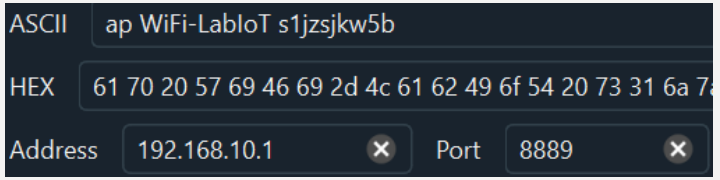
HARDWARE

- The drones are tello talent
- 5 megapixel camera
- expansion board with network board

CONNECTION

- To connect the drone to a WiFi network, you need to follow a simple procedure:Collegare la scheda di espansione al drone
- Connect directly to the network generated by the drone.
- Send a udp package «command» to ip 192.168.10.1 and port 8889
- Send a udp package to «ap ssid password»
- Turn off the drone, lift the switch on the expansion board and restart the drone





RECOVER THE IP

- Once the drone is connected to the WIFI, you need to retrieve its IP address (which cannot be set as fixed) using nmap. To do this, a regular expression has been used to find the IP within the output of nmap using the MAC address of the expansion board.

```
1 usage
def find_device(ip_address, mac, mac2):
    command = ["nmap", "-sn", ip_address]
    process = subprocess.Popen(command, stdout=subprocess.PIPE)
    ris, err = process.communicate()
    output=ris.decode('utf-8')
    print("Ip del primo drone: ")
    getIP(output, mac)
    print("Ip del secondo drone: ")
    getIP(output, mac2)


2 usages
def getIP(text, pattern):
    lines = text.split('\n')
    match_index = None
    for i, line in enumerate(lines):
        if pattern in line:
            match_index = i
            break
    if match_index is not None and match_index >= 2:
        print(lines[match_index - 2])


if __name__ == "__main__":
    ip_address = "192.168.1.0/24"
    mac1 = "9C:50:D1:3B:5B:94"
    mac2="9C:50:D1:3B:54:08"
    out=find_device(ip_address,mac1, mac2)
```

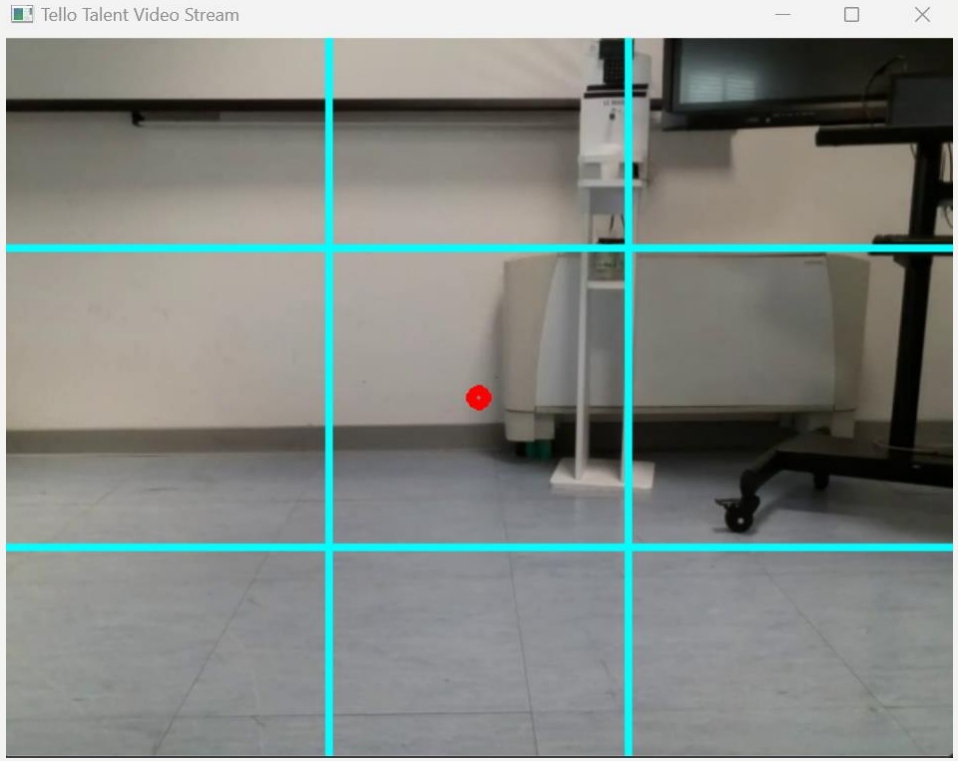
```
Ip del primo drone:
Nmap scan report for RMTT-3B5B94.csedu.unisa.it (192.168.1.144)
Ip del secondo drone:
Nmap scan report for RMTT-3B5408.csedu.unisa.it (192.168.1.107)
```

SOFTWARE: STREAMING

- The stream is received on port 11111 on a separate thread.
- The cv2.medianBlur() function can be called to eliminate noise from the image.
- A mask is applied to exclude all objects of non-red color with H: (150-179), S: (100-255), V: (100-255).

```
def receive_video():
    global dir
    dir = 0
    tello_address = ("192.168.1.107", 11111)
    cap = cv2.VideoCapture(f'udp://{tello_address[0]}:{tello_address[1]}')
    while True:
        ret, vid = cap.read()
        img = cv2.resize(vid, dsize: (width, height))
        frame = img.copy()
        hsvImage = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(hsvImage, lower, upper)
        #mask = cv2.medianBlur(mask, 15)
        mask_ = Image.fromarray(mask)
        bbox = mask_.getbbox()
        if bbox is not None:
            x1, y1, x2, y2 = bbox
            frame = cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 5)
            get_command(frame, x1, x2, y1, y2)
            if dir!=0:
                print(dir)
        display(frame)
        if not ret:
            break
        cv2.imshow( winname: 'Tello Talent Video Stream', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            me.land()
            break
    cap.release()
    cv2.destroyAllWindows()
```

SOFTWARE: VIDEO MATRIX



- Grid applied on the stream video

```
def display(img):
    cv2.line(img, pt1: (int(frameWidth/2)-deadZone,0), pt2: (int(frameWidth/2)-deadZone,frameHeight), color: (255,255,0), thickness: 3)
    cv2.line(img, pt1: (int(frameWidth/2)+deadZone,0), pt2: (int(frameWidth/2)+deadZone,frameHeight), color: (255,255,0), thickness: 3)
    cv2.circle(img, center: (int(frameWidth/2),int(frameHeight/2)), radius: 5, color: (0,0,255), thickness: 5)
    cv2.line(img,  pt1: (0,int(frameHeight / 2) - deadZone),  pt2: (frameWidth,int(frameHeight / 2) - deadZone),  color: (255, 255, 0),  thickness: 3)
    cv2.line(img,  pt1: (0, int(frameHeight / 2) + deadZone),  pt2: (frameWidth, int(frameHeight / 2) + deadZone),  color: (255, 255, 0),  thickness: 3)
```

SOFTWARE: GET POSITION

```
def get_command(img,x1, x2, y1, y2):
    global dir
    dir=0
    cx = int((x1+x2)/2)
    cy = int((y1+y2)/2)
    if (cx <int(frameWidth/2)-deadZone):
        cv2.putText(img, text: " GO LEFT ", org: (20, 50), cv2.FONT_HERSHEY_COMPLEX, fontScale: 1, color: (0, 0, 255), thickness: 3)
        cv2.rectangle(img,(0,int(frameHeight/2-deadZone)),(int(frameWidth/2)-deadZone,int(frameHeight/2)+deadZone),(0,0,255),cv2.FILLED)
        dir = 1

    elif (cx > int(frameWidth / 2) + deadZone):
        cv2.putText(img, text: " GO RIGHT ", org: (20, 50), cv2.FONT_HERSHEY_COMPLEX, fontScale: 1, color: (0, 0, 255), thickness: 3)
        cv2.rectangle(img,(int(frameWidth/2+deadZone),int(frameHeight/2-deadZone)),(frameWidth,int(frameHeight/2)+deadZone),(0,0,255),cv2.FILLED)
        dir = 2

    elif (cy < int(frameHeight / 2) - deadZone):
        cv2.putText(img, text: " GO UP ", org: (20, 50), cv2.FONT_HERSHEY_COMPLEX, fontScale: 1, color: (0, 0, 255), thickness: 3)
        cv2.rectangle(img,(int(frameWidth/2-deadZone),0),(int(frameWidth/2+deadZone),int(frameHeight/2)-deadZone),(0,0,255),cv2.FILLED)
        dir = 3

    elif (cy > int(frameHeight / 2) + deadZone):
        cv2.putText(img, text: " GO DOWN ", org: (20, 50), cv2.FONT_HERSHEY_COMPLEX, fontScale: 1, color: (0, 0, 255), thickness: 3)
        cv2.rectangle(img,(int(frameWidth/2-deadZone),int(frameHeight/2)+deadZone),(int(frameWidth/2+deadZone),frameHeight),(0,0,255),cv2.FILLED)
        dir = 4

    else: dir=0
```

SOFTWARE: SEND COMMAND

```
me.takeoff()
while True:

    time.sleep(2)
    if dir != 0:
        print(dir)
    if dir == 1:
        me.move_left(40)
    if dir == 2:
        me.move_right(40)
    if dir == 3:
        me.move_up(40)
    if dir == 4:
        me.move_down(40)
```

CONCLUSIONS

- Applying the median filter (O(n^2log(n)) could eliminate the noise, but it might affect the performance of the program.
- Color-based tracking is computationally efficient but may generate many false positives. An alternative approach could be to use YOLOv4 for real-time drone detection after appropriate training.
- By modifying the 'deadzone' variable, the shape of the matrix and the resulting relative tracking can be altered.

SOFTWARE: DOCUMENTATION

- https://dl.djicdn.com/downloads/RoboMaster+TT/Tello_SDK_3.0_User_Guide_en.pdf
- https://dl.djicdn.com/downloads/RoboMaster+TT/RoboMaster_TT_Tello_Talent_User_Manual_en.pdf
- Github SDK python: https://github.com/damiafuentes/DJITelloPy
- Github of the project: https://github.com/CapMark/TelloTalent-ColorDetection