Name: Aniket Madhukar Kanade

Minidron navigation application

Introduction:

Design and implement an application that will control a drone based on information from its cameras. Use the vertical camera to recognise the coloured object and track the object.

Implementation:

I am using **Parrot Mambo FPV**. In the Mambo FPV, there are 2 cameras one vertical and one horizontal, you can use the wifi interface. The wifi interface works on Mac, Linux, or Windows operating systems. To implement this goal I have divided it into several tasks. Combining all the tasks fulfils this goal. I am using pyparrot libraries in this project.

Task 1: Connection to parrot Mambo FPV

You can connect to the wifi on the FPV camera. Do not use this if the camera is not connected. Also, ensure you have connected your machine to the wifi on the camera before attempting this or it will not work. Mambo constructor takes 2 parameters.
IP address: unique address for this mambo (can be ignored if you are using wifi)
Use_wifi: set to True to connect with wifi instead of BLE

```
Mambo(None, use_wifi=True)
```

Task 2: Take off and increase the drone’s vertical distance.

Uses fly_direct method with 5 parameters. roll, pitch(cover horizontal distance), yaw(twist about a vertical axis), vertical_movement and duration. All the parameters range between -100 to 100.

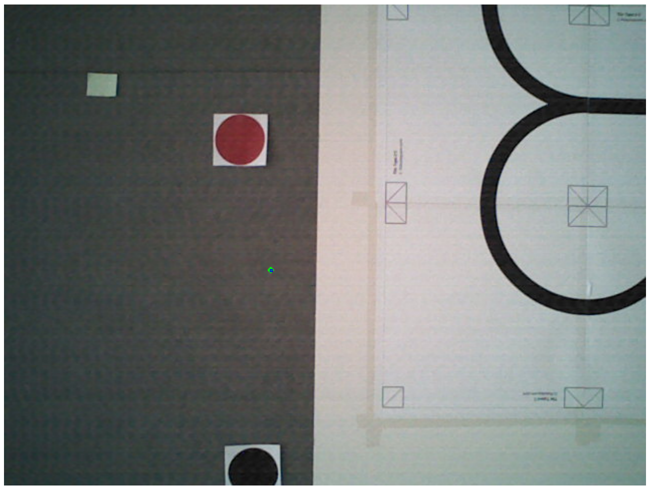
roll: roll speed in -100 to 100
pitch: pitch speed in -100 to 100
yaw: yaw speed in -100 to 100
vertical_movement: vertical speed in -100 to 100
duration: optional: seconds for a specified duration or None to send it once

```
mambo.safe_takeoff(5)
mambo.fly_direct(roll=0,pitch=0,yaw=0,vertical_movement=30,dura
tion=5)
```

Name: Aniket Madhukar Kanade

Task 3: Delete previous pictures, take a photograph by the ground camera and display the pictures.

```
picture_names1 = mambo.groundcam.get_groundcam_pictures_names()
for file in picture_names1:
mambo.groundcam._delete_file(file)
```


Ground camera image

Name: Aniket Madhukar Kanade

Task 4: Target certain coloured objects using the OpenCV library

OpenCV-Python is a library of Python bindings designed to solve computer vision problems In this task, we target the range of colours instead of targeting a single colour. The range is defined in HSV colour format.

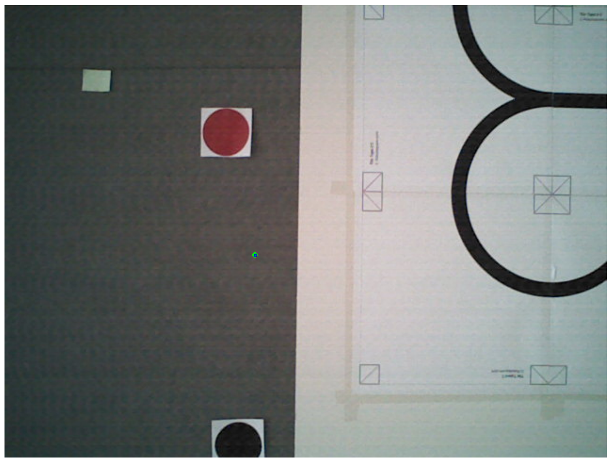
cvtColor(): Converts an image from one colour space to another.

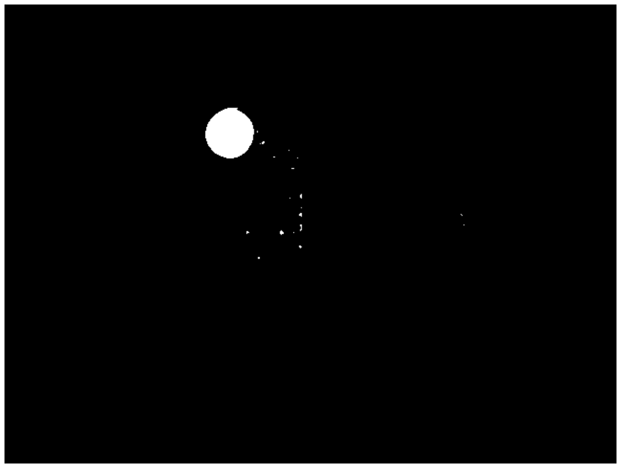
inRange(): OpenCV program in python to mask the given image by specifying the lower bounds and upper bounds then displaying the resulting image as the output on the screen using inRange() function.

bitwise_or(): A bitwise OR is true if either of the two pixels is greater than zero.

Example: For the red colour, we are using the following limits.

```
lower1 = np.array([0,50,50])
upper1 = np.array([20,255,255])
lower2 = np.array([160,50,50])
upper2 = np.array([180,255,255])
image = cv2.cvtColor(blurred_frame, cv2.COLOR_BGR2HSV)
mask1 = cv2.inRange(image, lower1, upper1)
mask2 = cv2.inRange(image, lower2, upper2)
mask = cv2.bitwise_or(mask1, mask2)
```


Original Image

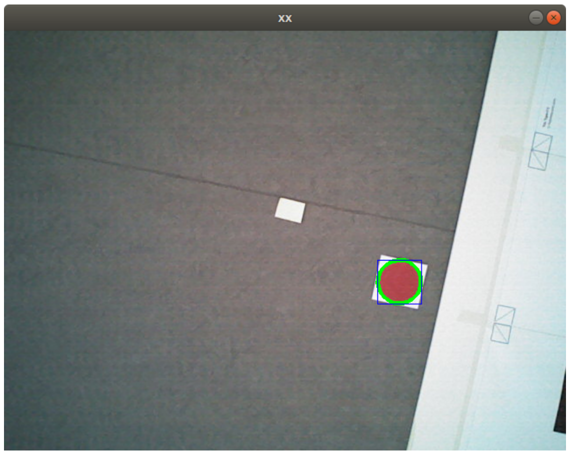

Masked Image

Task 5: Finding and drawing contours

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition. The findContours function in OpenCV, finds contours in a binary image.

Name: Aniket Madhukar Kanade

```
_, contours, _ = cv2.findContours(mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)
for contour in contours:
    area = cv2.contourArea(contour)
    if area >= 1:
        cv2.drawContours(frame, contour, -1, (0,255,0), 3)
```

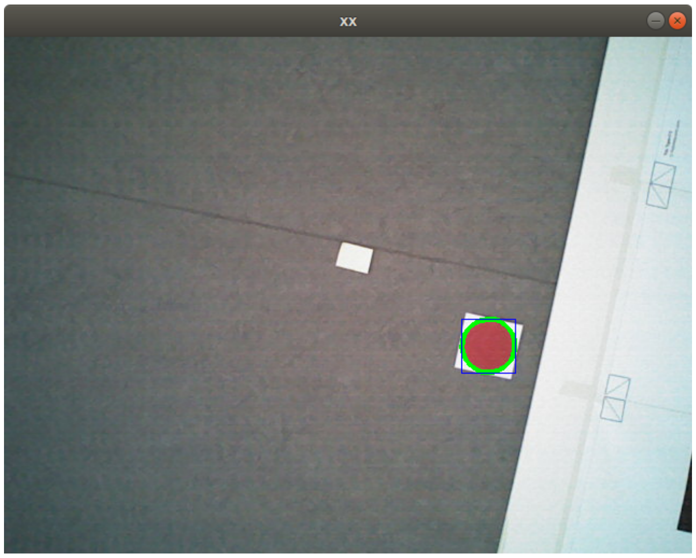


Drawing contours

Task 6: Get the coordinates of the target

The cv2. boundingRect() function of OpenCV is used to draw an approximate rectangle around the binary image. This function is used mainly to highlight the region of interest after obtaining contours from an image. Image size is 640 X 480px. We are calculating the coordinates with respect to image.

```
x,y,w,h = cv2.boundingRect(contour)
```



countours

Name: Aniket Madhukar Kanade

Task 7: Displaying the Images

In this task, we create two types of images normal image and image in BGR2HSV. cv2.imshow() -> Displays an image in the specified window.
cv2.waitKey() -> Allows you to wait for a specific time in milliseconds until you press any button on the keyword. It accepts time in milliseconds as an argument.

```
cv2.imshow("xx", frame)
cv2.waitKey(-1)
cv2.imshow("mask", mask)
cv2.waitKey(-1)
```

Task 8: Get the coordinates of the target relative to the drone

```
x1 = frame.shape[0]
y1 = frame.shape[1]
X_target = x - x1/2
Y_target = y - y1/2
```

Task 9: Get the angle of the target relative to the drone.

Calculating distance between drone coordinates and target coordinates by distance formula. As we have the side length and hypotenius by taking inverse of cos(side_length/hypotenius) we can calculate the angle. After getting the angle in radians we have converted it to degrees as drone accepts input in degrees and not in radians. We have used 1rad × 180/π = 57.296° formula for this conversion.
Doing the calculations:

```
side_length = y-y1/2
hypotenius =  math.sqrt((X_target - X_drone)**2 + (Y_target -
Y_drone)**2) angle_wrt_yaxis =
math.acos(side_length/hypotenius)
```

Task 10: Turn the drone along its vertical to the direction of the target.

This can be done by the yaw function of the fly_Direct() method. This task can also be done by turn_degrees(degrees)

```
if(X_target >= 0 and Y_target >= 0):
    angle_wrt_yaxis = angle_wrt_yaxis
elif(X_target >= 0 and Y_target <= 0):
    angle_wrt_yaxis = 90 - angle_wrt_yaxis
elif(X_target <= 0 and Y_target >= 0):
    angle_wrt_yaxis =  angle_wrt_yaxis -180
else:
    angle_wrt_yaxis =  - angle_wrt_yaxis
mambo.turn_degrees(int(angle_wrt_yaxis))
```

Task 11: Navigate the drone to target.

This can be done by the pitch function of the fly_Direct() method. The range of pitch is -100 to 100.

```
mambo.fly_direct(roll=0,pitch=hypotenius,yaw=angle_wrt_yaxis,ve
rtical_movement=0,duration=2)
```

Task 12: Land and Disconnect.

Can be implemented by safe_land and disconnect methods.
safe_land: Ensure the mambo lands by sending the command until it shows landed on sensors
disconnect: Disconnect the connection. Always call this at the end of your programs to cleanly disconnect.

```
mambo.safe_land(5)
mambo.disconnect()
```

Conclusion:

Hence, I have implemented an application that controls drones based on their vertical camera. Amongst all the tasks, tasks including `mambo.fly_direct()` method are not able to reflected on drone. Remaining all tasks including connection establishment with drone, safe take off, capturing the image from ground camera, tracking the coloured object from image, getting the coordinates from the object, visualization of object using rectangle inside image, changing the angle of the drone with respect to target object, safelanding of drone and disconnecting the drone are reflected as per our requirement.

Name: Aniket Madhukar Kanade

References:

1. Drone Simulation and Control, Part 1: Setting Up the Control Problem. YouTube, 2018. Dostupn´e tak´e z: https : / / www . youtube . com / watch ? v = hGcGPUqB67Q & amp ; list = PLy8TVa88QVfoOotVLloGWvK6PMSdjEtSb&index=1&t=65s.
2. Welcome to pyparrot's documentation! [Online]. [N.d.] [cit. 2021-04-24]. Dostupn´e z: https: //pyparrot.readthedocs.io/en/latest/
3. PARROT-DEVELOPERS. Parrot-Developers/RollingSpiderEdu [online]. [N.d.] [cit. 2021-04-24]. Dostupn´e z: https://github.com/Parrot-Developers/RollingSpiderEdu.
4. Parrot Minidrones Support from Simulink [online]. [N.d.] [cit. 2021-04-24]. Dostupn´e z: https://www.mathworks.com/hardware-support/parrot-minidrones.html.
5. Autonomous Navigation, Part 1: What Is Autonomous Navigation? YouTube, 2020. Dostupn´e tak´e z: https://www.youtube.com/watch?v=Fw8JQ5Q-ZwU&t=32s.
6. YUFKA, Alpaslan; PARLAKTUNA, Osman. Performance Comparison of BUG Algorithms for Mobile Robots. In: 2009. Dostupn´e z doi: 10.13140/RG.2.1.2043.7920.
7. LIN, Zijie; CASTANO, Lina; XU, Huan. A Fast Obstacle Collision Avoidance Algorithm for Fixed Wing UAS. In: 2018 International Conference on Unmanned Aircraft Systems (ICUAS). 2018, s. 559–568. Dostupn´e z doi: 10.1109/ICUAS.2018.8453307.
8. GONZALEZ, David; P ´ EREZ, Joshu´e; MILAN ´ ES, Vicente; NASHASHIBI, Fawzi. A Review ´ of Motion Planning Techniques for Automated Vehicles. IEEE Transactions on Intelligent Transportation Systems. 2016, roˇc. 17, ˇc. 4, s. 1135–1145. Dostupn´e z doi: 10.1109/TITS. 2015.2498841.
9. KARAMAN, Sertac; WALTER, Matthew R.; PEREZ, Alejandro; FRAZZOLI, Emilio; TELLER, Seth. Anytime Motion Planning using the RRT*. In: 2011 IEEE International Conference on Robotics and Automation. 2011, s. 1478–1483. Dostupn´e z doi: 10.1109/ ICRA.2011.5980479.
10. LIU, Yang; XIE, Zongwu; LIU, Hong. An Adaptive and Robust Edge Detection Method Based on Edge Proportion Statistics. IEEE Transactions on Image Processing. 2020, roˇc. 29, s. 5206–5215. Dostupn´e z doi: 10.1109/TIP.2020.2980170.
11. HE, Kaiming; GKIOXARI, Georgia; DOLLAR, Piotr; GIRSHICK, Ross. Mask R-CNN. ´ In: 2017 IEEE International Conference on Computer Vision (ICCV). 2017, s. 2980–2988. Dostupn´e z doi: 10.1109/ICCV.2017.322.
12. 2021. Dostupn´e tak´e z: https://opencv.org/.
13. YANG, Ming-Der; BOUBIN, Jayson G.; TSAI, Hui Ping; TSENG, Hsin-Hung; HSU, YuChun; STEWART, Christopher C. Adaptive autonomous UAV scouting for rice lodging assessment using edge computing with deep learning EDANet. Computers and Electronics in Agriculture. 2020, roˇc. 179, s. 105817. issn 0168-1699. Dostupn´e z doi: https://doi. org/10.1016/j.compag.2020.105817.
14. WU, Yuxin; KIRILLOV, Alexander; MASSA, Francisco; LO, Wan-Yen; GIRSHICK, Ross. Detectron2 [https://github.com/facebookresearch/detectron2]. 2019 [cit. 2021-04- 24].