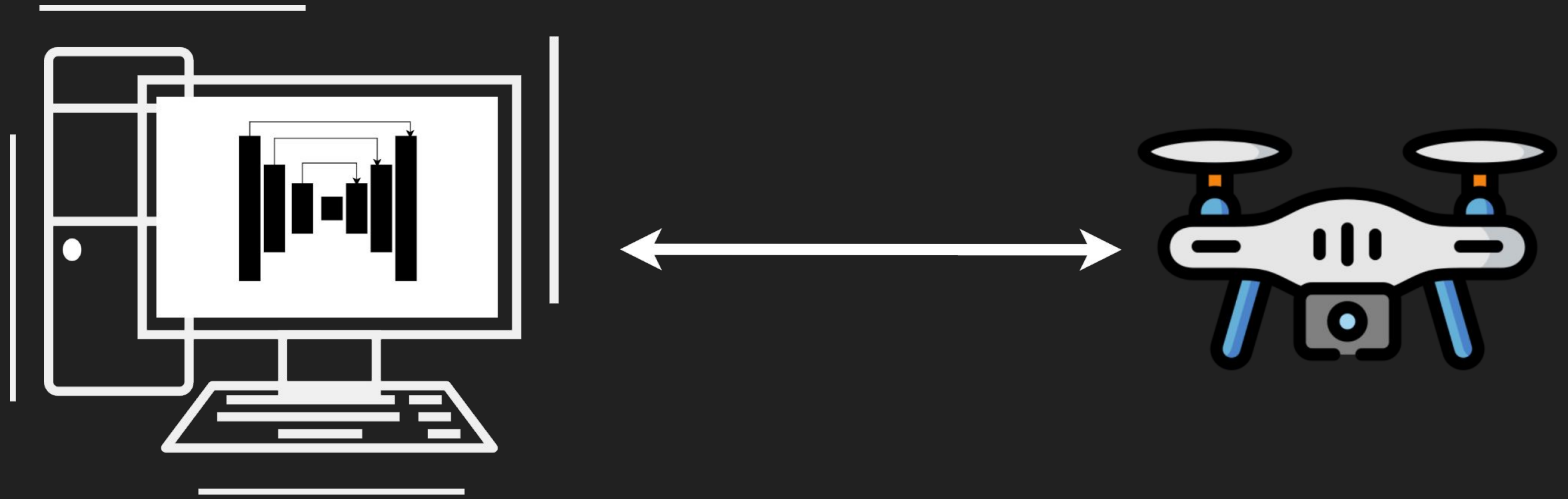
Prototype of an autonomous selfie drone utilizing pose estimation

Felix Hamburger

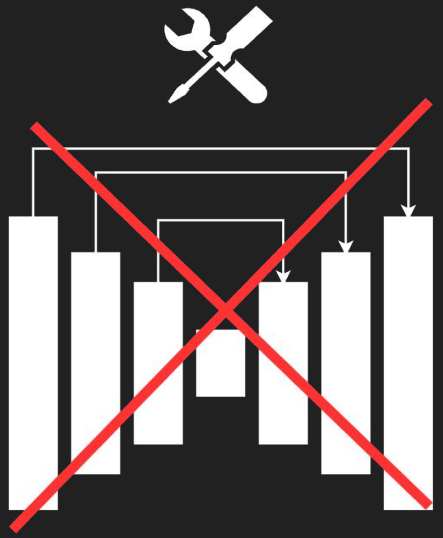
Structure

1. Aim of work
2. Tello Drone
3. Mediapipe
4. Implementation
    a. Algorithm
    b. Autonomous flight
    c. Poses
5. Problems
6. Conclusion

Aim of work



Aim of work



Tello Drone

Tello Drone

Ryze Robotics

Price: 100€ ~

Flight Performance:
Max. Flight Distance: 100 m
Max.Speed : 8 m/s
Max. Flight Time: 13 min
Max Flight Height: 30 m

Camera:
Photo: 5 Megapixel (2592×1936)
FOV: 82,6°
Video: HD720p30
Format: JPG (Foto), MP4 (Video)

Tello SDK

UDP Port 8889

Commands

UDP Port 8890

States

Port 11111

Video Stream

Control

up x	Ascend to x cm
cw x	Rotate x degrees clockwise
right x	Fly right for x cm

Set

speed x	set speed to x cm/s
rc a b c d	Control via 4 channels. a = left/right b = forward/backward c = up/down d = yaw

Read

speed?	Current speed cm/s
battery?	Current battery percentage
time?	Current flight time

```python
def send_command(self, command):
    """
    Send a command to the Tello and wait for a response.

    :param command: Command to send.
    :return (str): Response from Tello.

    """

    print (">> send cmd: {}".format(command))
    self.abort_flag = False
    timer = threading.Timer(self.command_timeout, self.set_abort_flag)

    self.socket.sendto(command.encode('utf-8'), self.tello_address)

    timer.start()
    while self.response is None:
        if self.abort_flag is True:
            break
    timer.cancel()

    if self.response is None:
        response = 'none_response'
    else:
        response = self.response.decode('utf-8')

    self.response = None

    return response
```

Port 11111

Video Stream


0:00/3:53

DJITelloPy

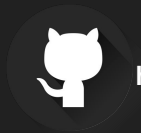
→ Python interface using the official Tello SDK

- implementation of all tello commands
- easily retrieve video stream

```
connect(wait_for_state=True)
```

```
send_rc_control(left_right_velocity, forward_backward_velocity, up_down_velocity, yaw_velocity)
```

```
get_frame_read().frame
```



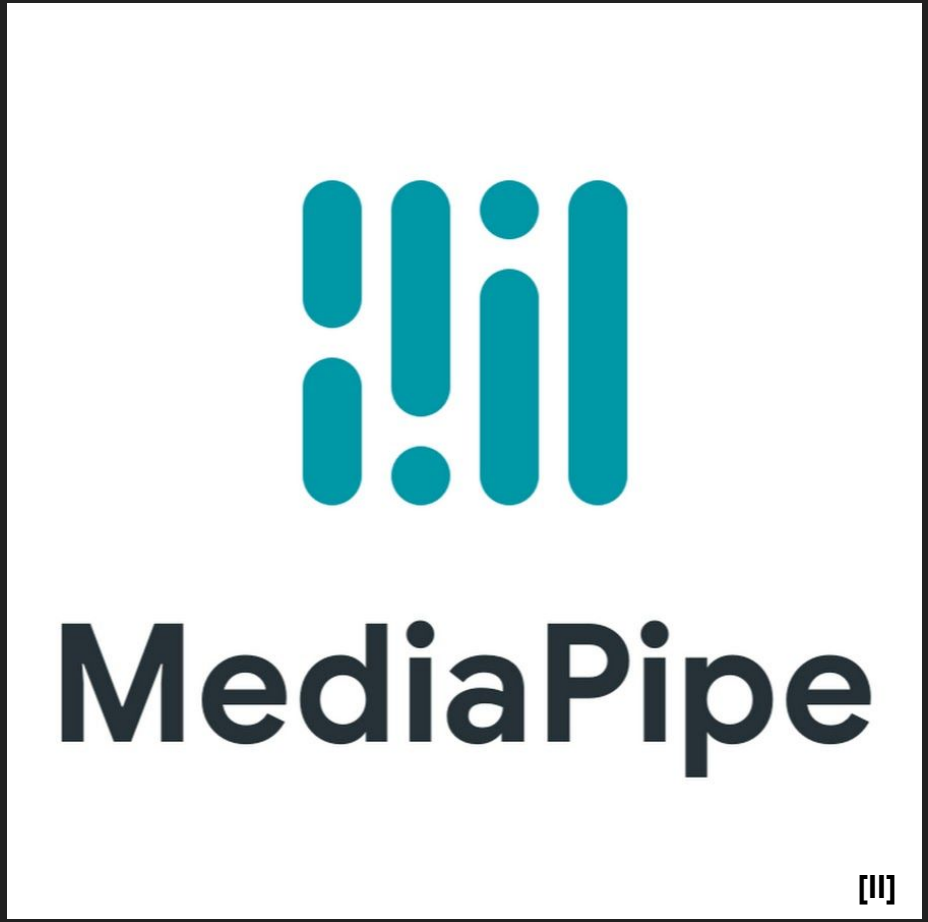
https://github.com/damiafuentes/DJITelloPy

Media Pipe

Media Pipe

- From Google
- Open Source
- Offers several ML Solutions

→ **Pose Estimation**



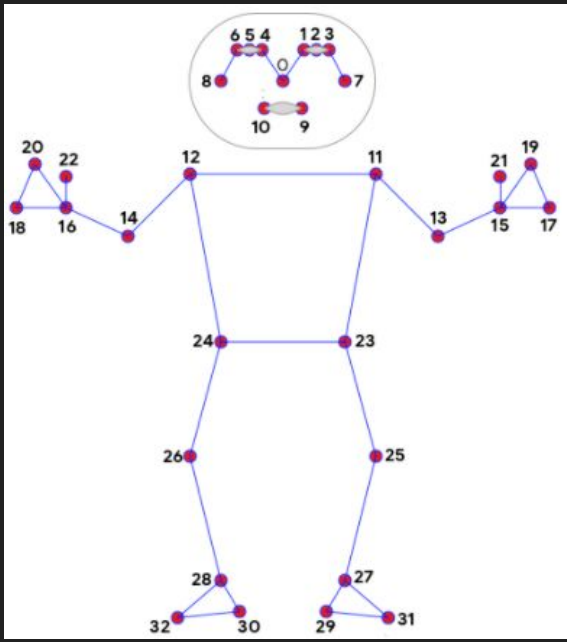
Blaze Pose

Landmarks:

x and **y**: Landmark coordinates normalized to [0, 1.0] by the images height and width respectively.

visibility: Indicating the likelihood of the landmark being visible [0, 1.0].

(**z:** Depth of the landmark with midpoint of the depth at hips centre.)



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

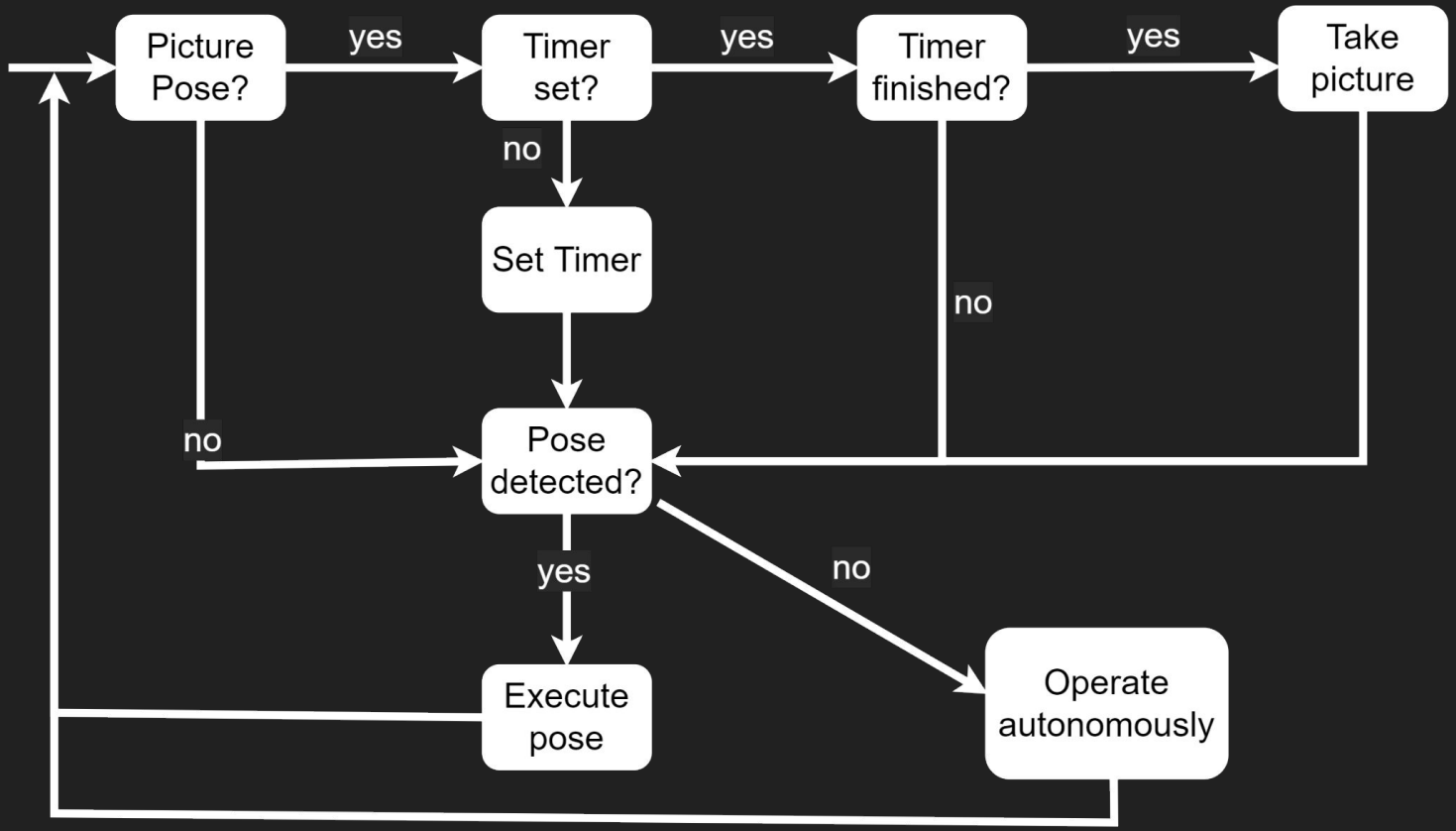
[II]

Blaze Pose

	Pixel 3 CPU FPS	Pixel 3 GPU FPS
BlazePose lite	44	112
BlazePose full	18	69

[IV]

Algorithm

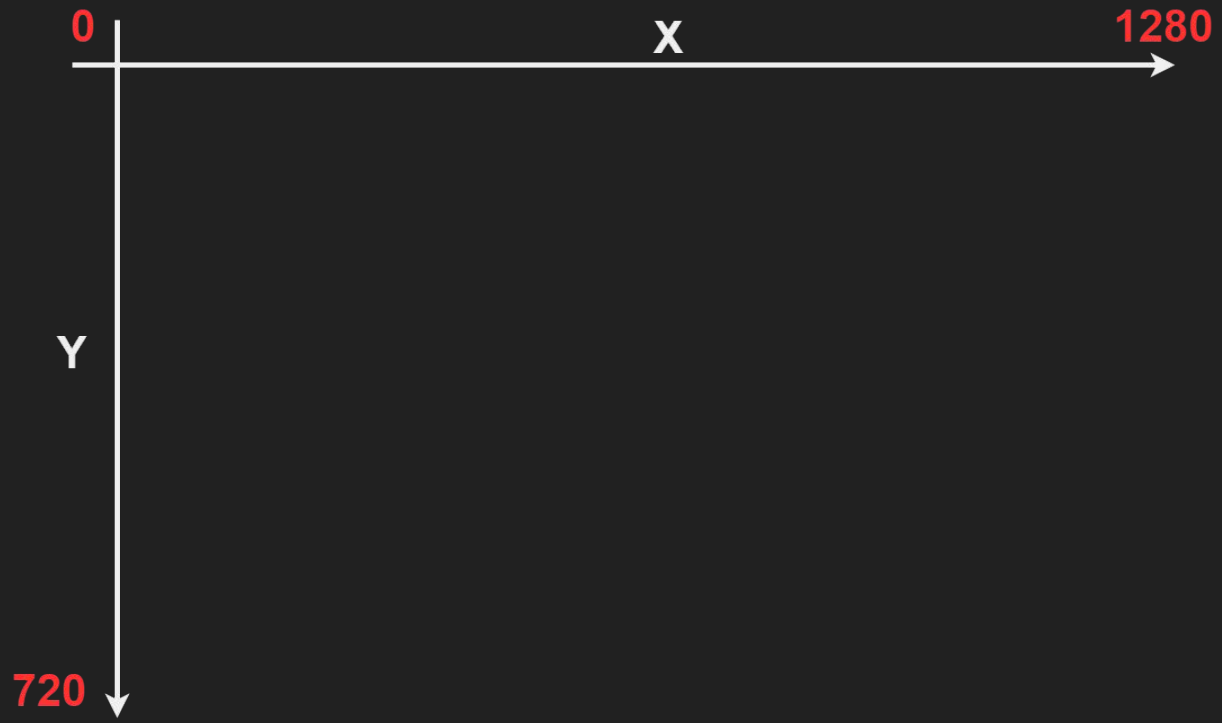

Picture Pose?
yes
Timer set?
yes
Timer finished?
yes
Take picture
no
Set Timer
no
no
Pose detected?
yes
no
Execute pose
Operate autonomously

Frame

Videoframe

- The **origin** is located in the **upper left corner** of the screen
- Y-Value positive in downwards direction.
- X-Values positive in right direction

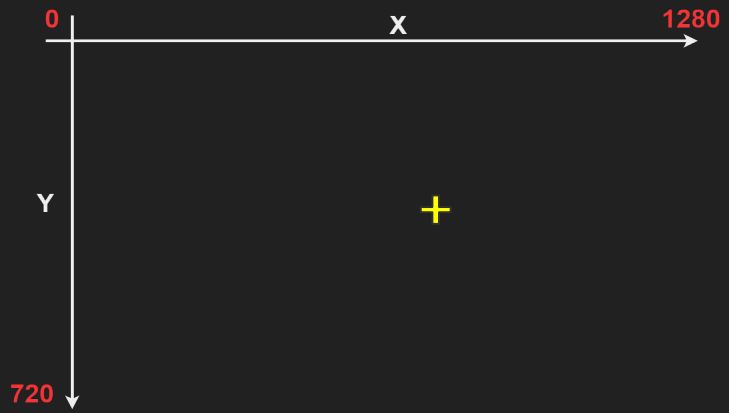
→ Perspective from the drone's point of view

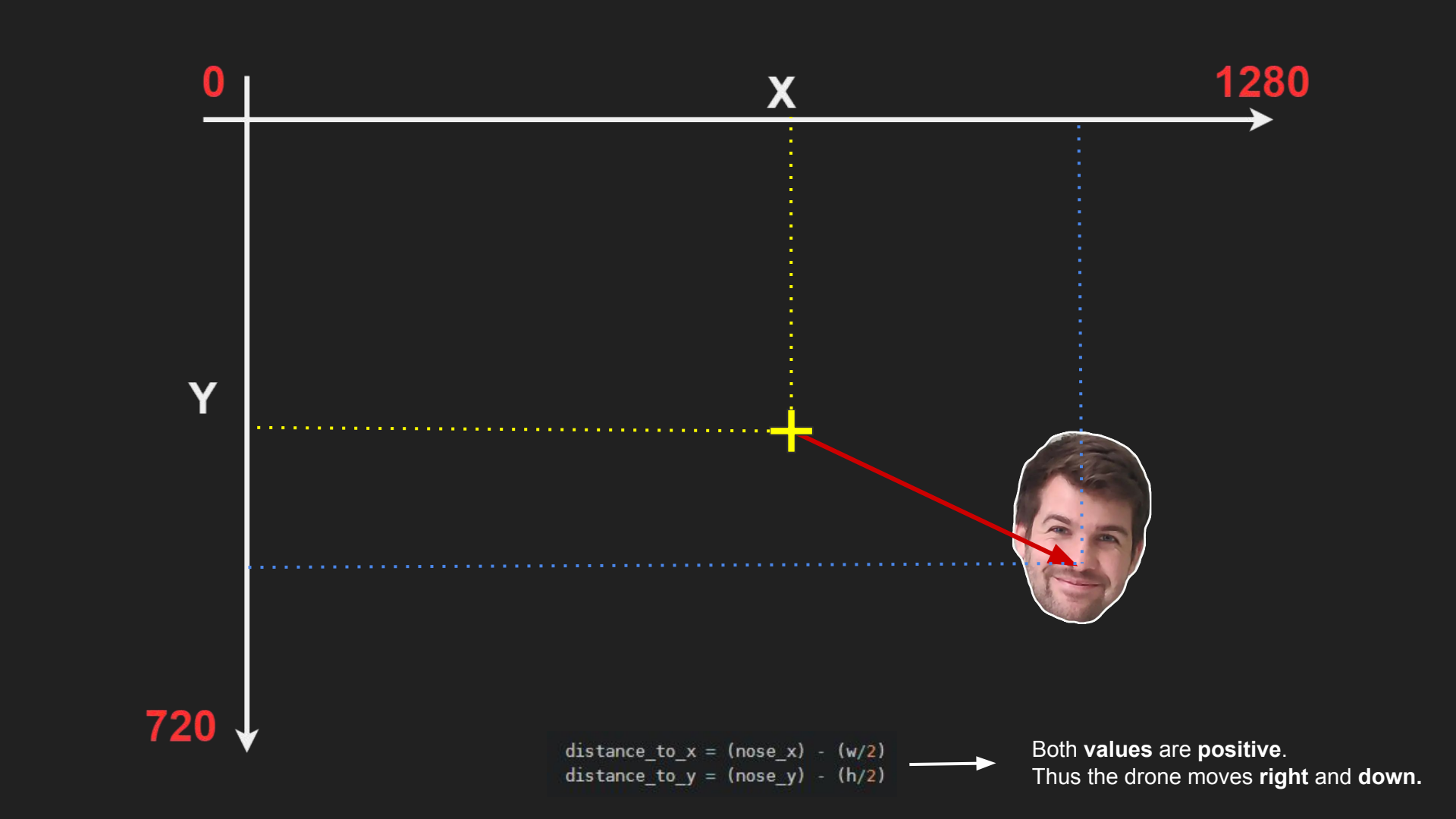


Autonomous movement

Autonomous movement

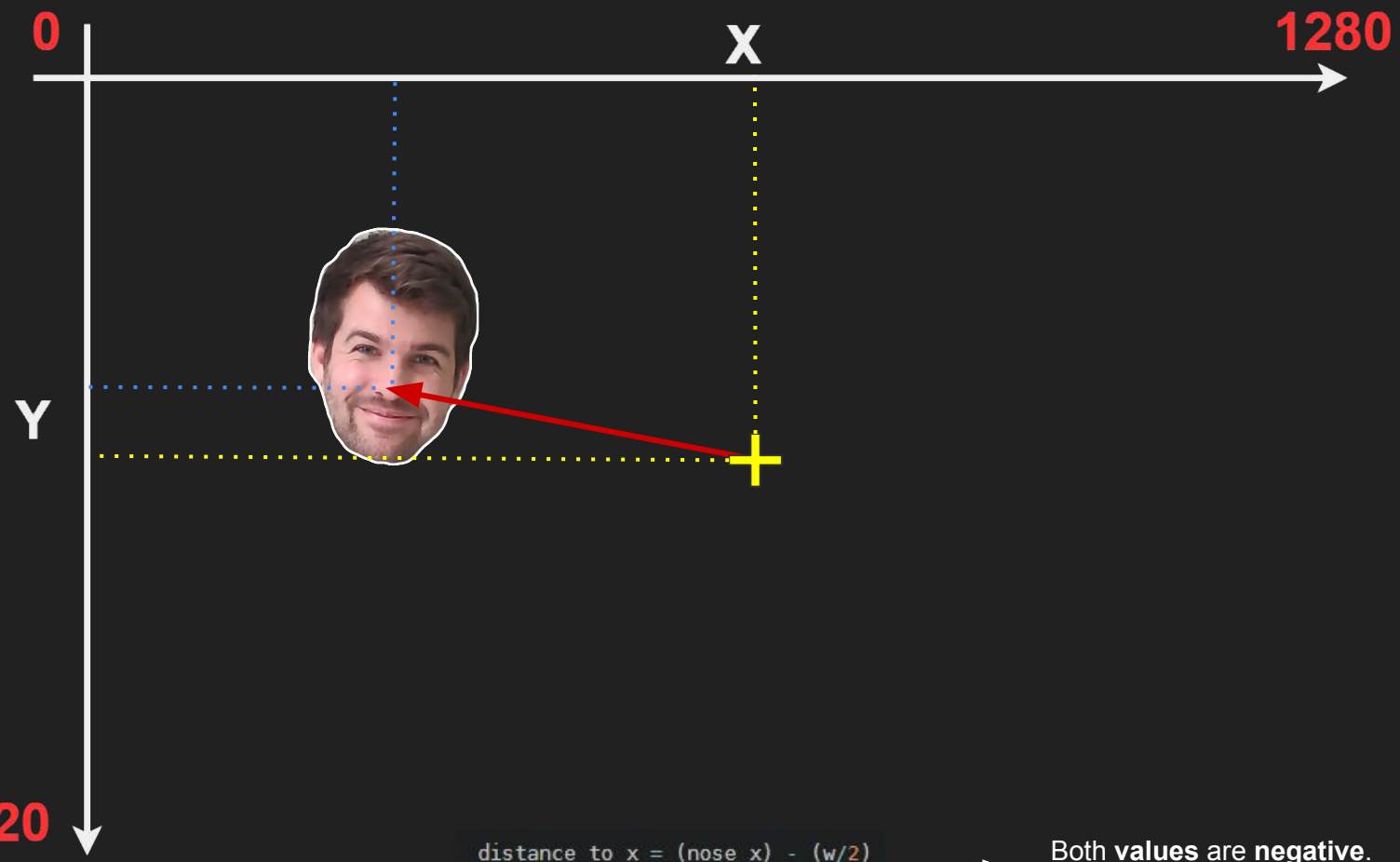
```
distance_to_x = (nose_x) - (w/2)
distance_to_y = (nose_y) - (h/2)
```





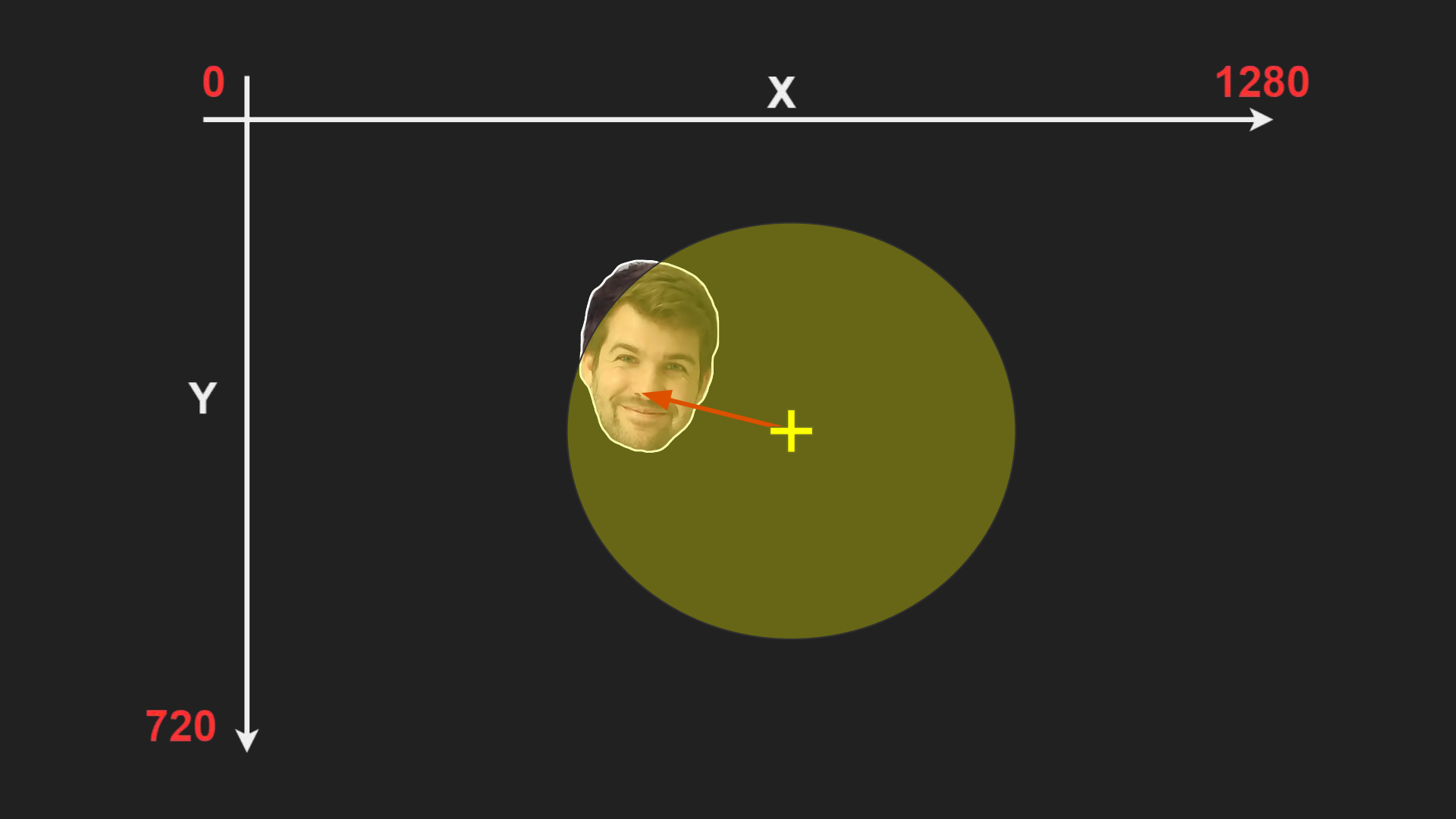

```
distance_to_x = (nose_x) - (w/2)
distance_to_y = (nose_y) - (h/2)
```

Both **values** are **positive**.
Thus the drone moves **right** and **down.**

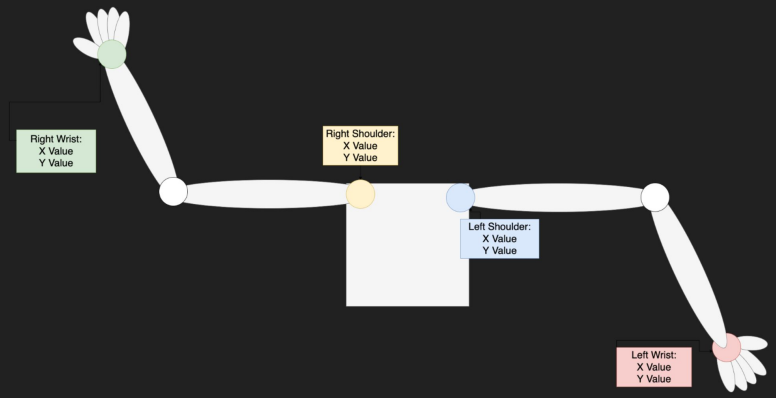



```
distance_to_x = (nose_x) - (w/2)
distance_to_y = (nose_y) - (h/2)
```

Both **values** are **negative**.
Thus the drone moves **left** and **up**


0
X
1280
Y
720

Poses



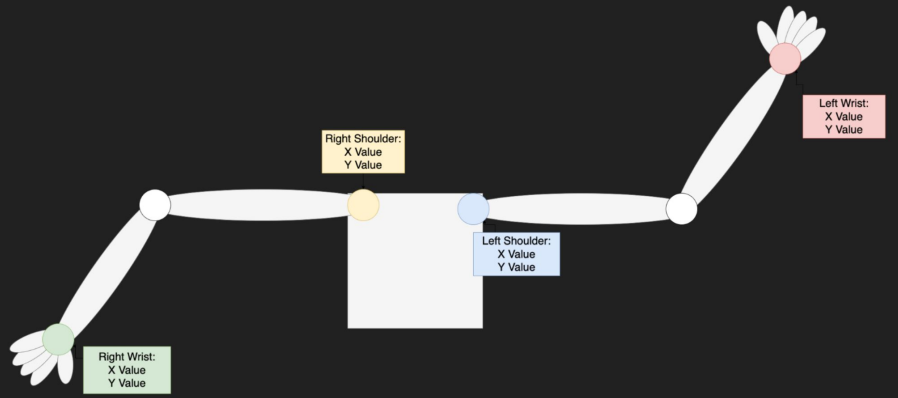
Right Arm Up:

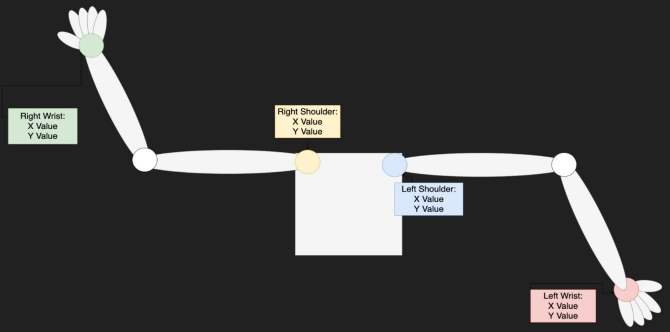
- Right Arm is above right shoulder
- Left arm is below left shoulder.

→ Due to inverse Y-Axis the values are also inverted.
E.g: Right Arm is up if y-value of right wrist is smaller than the y-value of the right shoulder.

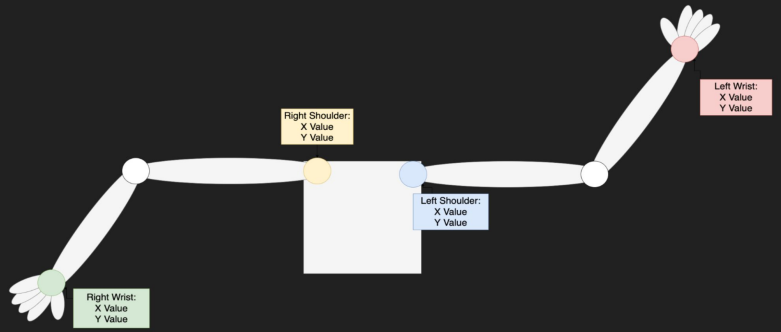
Left Arm up:

- Left arm is above left shoulder
- Right arm is below right shoulder.





Move left



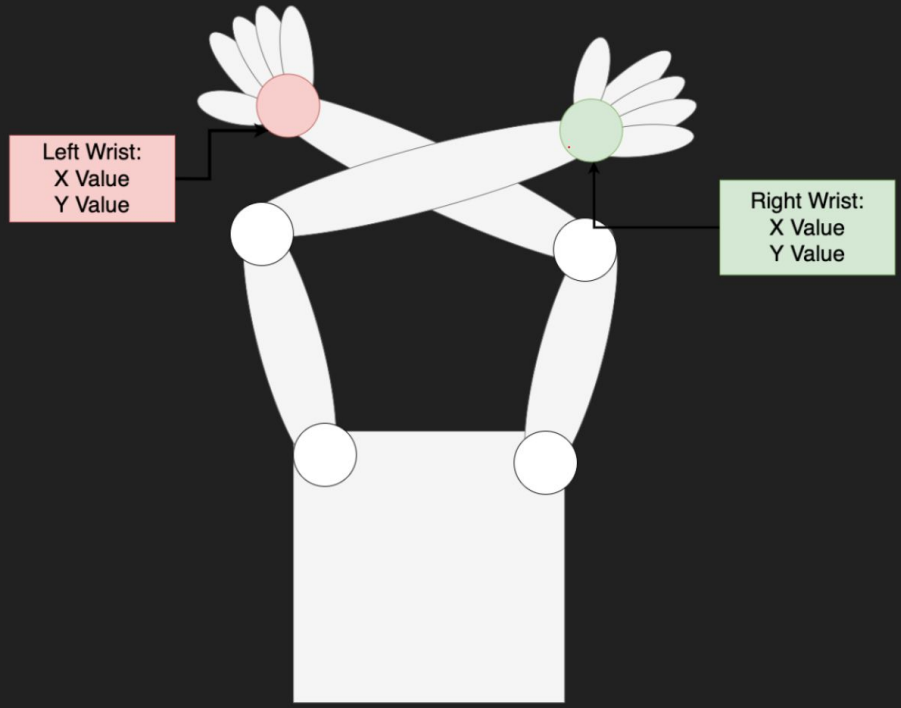
Move right

Picture Pose

- Pose detected when the right wrist passes the left wrist.

  → **X- Value of right wrist becomes larger**

→ You might notice the ambiguity of the pose the way it is detected.



Video

Problems

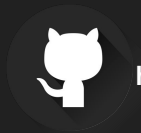
- Movement on Z-axis.
- Optimizing parameters is a bottomless pit.

→ High wear on rotor blades.

Conclusion

- Due to the fast CNN, 18 FPS can be achieved which is sufficient.
- Poses are detected reliably.
- Autonomous movement is stable with slow speed.

Thank you.



https://github.com/Baumwollboebele/autonnomous_selfie_drone

Quellen

- [I] https://cdn.shopify.com/s/files/1/0263/8469/5395/files/Ryze-Tello-review-closeup-749x500.jpg
- [II] https://yt3.ggpht.com/ytc/AKedOLRjqH5YNixEUi8z2zHilzJYPkBECw_UXlBp3uKc=s900-c-k-c0x00ffffff-no-rj
- [III] https://google.github.io/mediapipe/images/mobile/pose_tracking_full_body_landmarks.png
- [IV] https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html

Extras

Constants

```python
class Constants():
    """This class contains the neccesary values for the drones settings.
    Changing these values will result in a different behaviour of the drone.

    """
    @property
    def DRONE_SPEED_X(self) -> int:
        """The function returns the setting for the drones movement in the x direction (left and right).
        Movement is measured in centimeters and can be set from -100 to 100.

        Returns:
            (int): value of the distance
        """
        return 20

    @property
    def DRONE_SPEED_Y(self) -> int:
        """The function returns the setting for the drones movement in the y direction (up and down).
        Movement is measured in centimeters and can be set from -100 (down) to 100 (up).

        Returns:
            (int): value of the distance
        """
        return 10
```