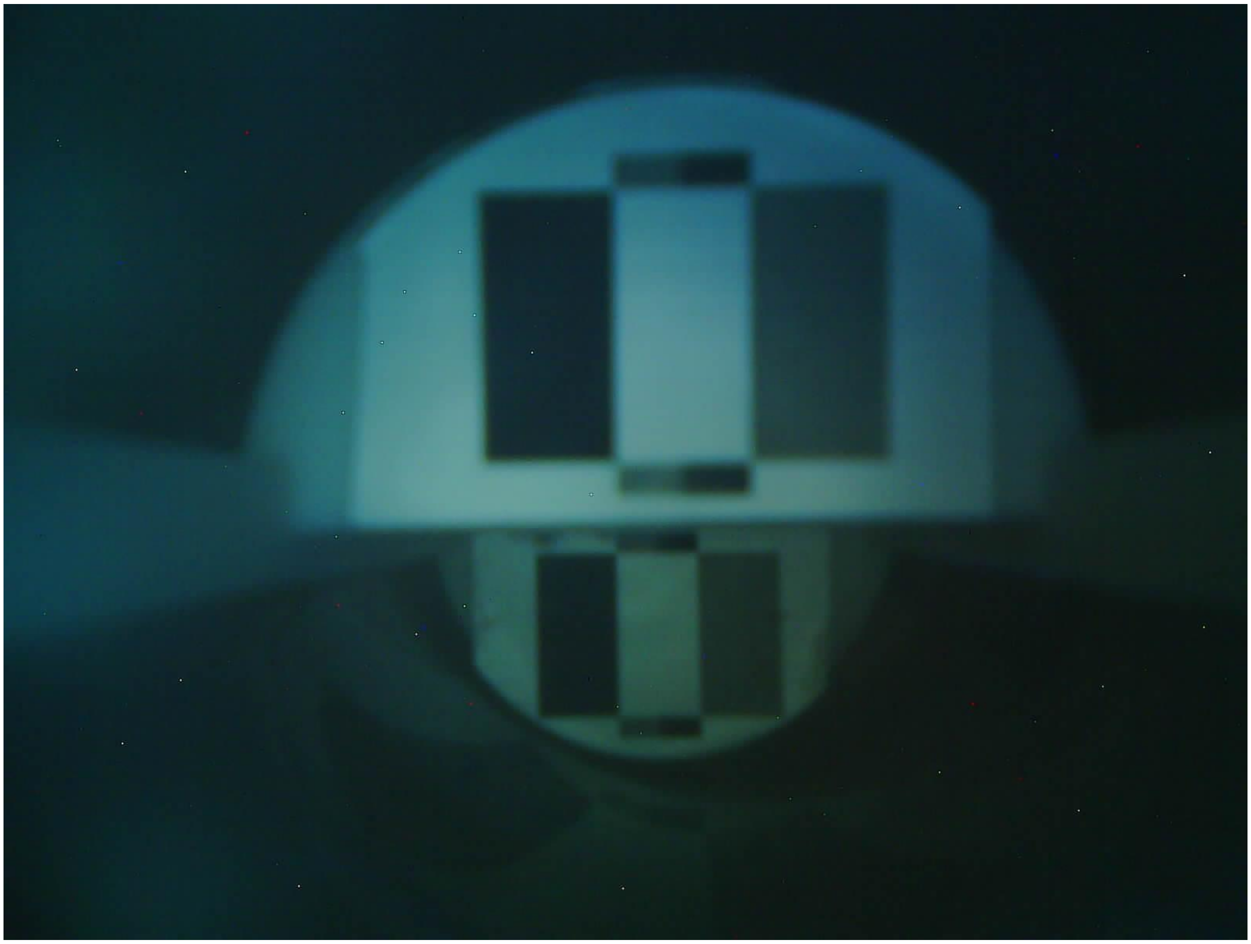
DOCUMENTATION
WATER QUALITY MEASUREMENT



Introduction

The water quality of a water sample can be measured using the device provided and algorithms to compute turbidity from the images taken using the camera present in the device.

Flow of Code

1. Pattern Detection.

The two patterns namely the upper and lower areas present in the image are detected using one of the two methods given below,

A. Using Machine Learning Model.

A YOLO ML model is trained against custom data to detect the two areas present in the image.

- Clone https://github.com/ultralytics/yolov5 into your local directory.
- Download the weights file and copy it into the repository cloned above.

  (https://github.com/jewelben-igc/Turbidity-Measurement/blob/main/best.pt)

- Run the command given below in a terminal.

```
python3 detect.py --weights best.pt --img img_size --conf conf_thres
--source <test_data_location>
```

  Here **img_size** is the image batch size (80, 160, 320, 640 etc.), **conf_thres** is the confidence threshold for detection (0 to 1) and **<test_data_location>** is the location of the test image folder.

- The output image will be present in a folder named `runs/detect/exp{number}` present in the cloned repository.

B. Using Hardcoded Coordinates.

The coordinates of the areas are manually calibrated and hardcoded.

- Download the python notebook using the given link.

https://github.com/jewelben-igc/Turbidity-Measurement/blob/main/hardcode.ipynb

- Assign IMG_PATH = r<test_image>.
- The coordinates of the four corners of the upper and lower areas can be manually calibrated in the notebook. For example, in the notebook provided, the coordinates of two opposite diagonals can be set for both areas. An example is provided below,

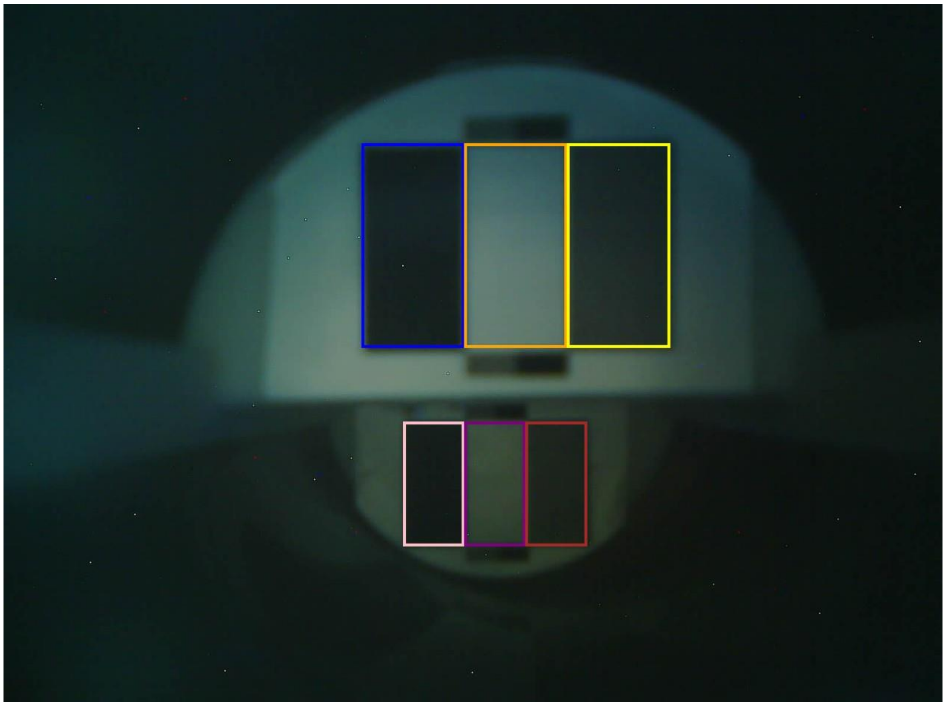
```
xmin_l = 660
ymin_l = 300
xmax_l = 1010
ymax_l = 555

xmin_u = 610
ymin_u = 620
xmax_u = 1010
ymax_u = 890
```

2. Detecting Subareas Present in the Areas.

The grey, white, and black subareas in both the upper and lower areas are detected.

Using the coordinates passed, the area is split equally into 3 areas column-wise to obtain the subareas. The vertical splits are hard coded as given below,

```
split_u1 = xmin_u + (xmax_u - xmin_u)//3
split_u2 = xmin_u + 2*(xmax_u - xmin_u)//3

split_l1 = xmin_l + (xmax_l - xmin_l)//3
split_l2 = xmin_l + 2*(xmax_l - xmin_l)//3
```



3. Calculation of RGB Pixel Values from the Subareas.

The red, green and blue pixel values of all six subareas are computed. This is done by taking the median of each colour channel of each subarea.

```
ub_m_r = np.median(img[ymin_u:ymax_u,xmin_u:split_u1,0])

ub_m_g = np.median(img[ymin_u:ymax_u,xmin_u:split_u1,1])

ub_m_b = np.median(img[ymin_u:ymax_u,xmin_u:split_u1,2])


uw_m_r = np.median(img[ymin_u:ymax_u,split_u1:split_u2,0])

uw_m_g = np.median(img[ymin_u:ymax_u,split_u1:split_u2,1])

uw_m_b = np.median(img[ymin_u:ymax_u,split_u1:split_u2,2])


ug_m_r = np.median(img[ymin_u:ymax_u,split_u2:xmax_u,0])

ug_m_g = np.median(img[ymin_u:ymax_u,split_u2:xmax_u,1])

ug_m_b = np.median(img[ymin_u:ymax_u,split_u2:xmax_u,2])
```

```
lb_m_r = np.median(img[ymin_l:ymax_l,xmin_l:split_l1,0])

lb_m_g = np.median(img[ymin_l:ymax_l,xmin_l:split_l1,1])

lb_m_b = np.median(img[ymin_l:ymax_l,xmin_l:split_l1,2])


lw_m_r = np.median(img[ymin_l:ymax_l,split_l1:split_l2,0])

lw_m_g = np.median(img[ymin_l:ymax_l,split_l1:split_l2,1])

lw_m_b = np.median(img[ymin_l:ymax_l,split_l1:split_l2,2])


lg_m_r = np.median(img[ymin_l:ymax_l,split_l2:xmax_l,0])

lg_m_g = np.median(img[ymin_l:ymax_l,split_l2:xmax_l,1])

lg_m_b = np.median(img[ymin_l:ymax_l,split_l2:xmax_l,2])
```

4. Input the Pixel Values and Coefficients into the Algorithm.

The pixel values calculated are used for further processing along with some predefined coefficients.

```
W1 = np.array([uw_m_r, uw_m_g, uw_m_b]) # Upper White Area [R G B]

W2 = np.array([lw_m_r, lw_m_g, lw_m_b]) # Lower White Area [R G B]


G1 = np.array([ug_m_r, ug_m_g, ug_m_b]) # Upper Gray Area [R G B]

G2 = np.array([lg_m_r, lg_m_g, lg_m_b]) # Lower Gray Area [R G B]


B1 = np.array([ub_m_r, ub_m_g, ub_m_b]) # Upper Black Area [R G B]

B2 = np.array([lb_m_r, lb_m_g, lb_m_b]) # Lower Black Area [R G B]


depth = np.array([-5 , -15]) # Depths of upper and lower plates


Secchi_coefficients = np.array([11.97, -0.7899])

Turbidity_coefficients = np.array([1.32, -1.39])

TSM_coefficients = np.array([1.2333, 0.6602])   #NOT YET CALIBRATED

CDOM_coefficients = np.array([5.2564, -6.1705]) #NOT YET CALIBRATED
```

5. Computation of Attenuation.

A. Black Area Correction

The correction for black areas is done using the formula **C = (W-B)/G** for both the upper and lower areas.

```
C1 = (W1-B1)/G1 # Upper Level

C2 = (W2-B2)/G2 # Lower Level
```

B. Attenuation

The attenuation for each colour channel is calculated by fitting the depths of the plates and the natural logarithm of the C1 and C2 values of each channel into a straight line. The attenuation is calculated as **P*100** where **P** is the slope of the straight line obtained.

```
# Attenuation
p = np.polyfit(depth, np.log([C1[0], C2[0]]), 1)

K_R = p[0]*100 # Attenuation for Red

p = np.polyfit(depth, np.log([C1[1], C2[1]]), 1)

K_G = p[0]*100 # Attenuation for Green

p = np.polyfit(depth, np.log([C1[2], C2[2]]), 1)

K_B = p[0]*100 # Attenuation for Blue
```

6. Computation of Water Quality.

The parameters K_mean_RG, Secchi Depth, Turbidity, CDOM ratio, Absorption by CDOM and Total Suspended Matter are computed by the algorithm using the formulas given below,

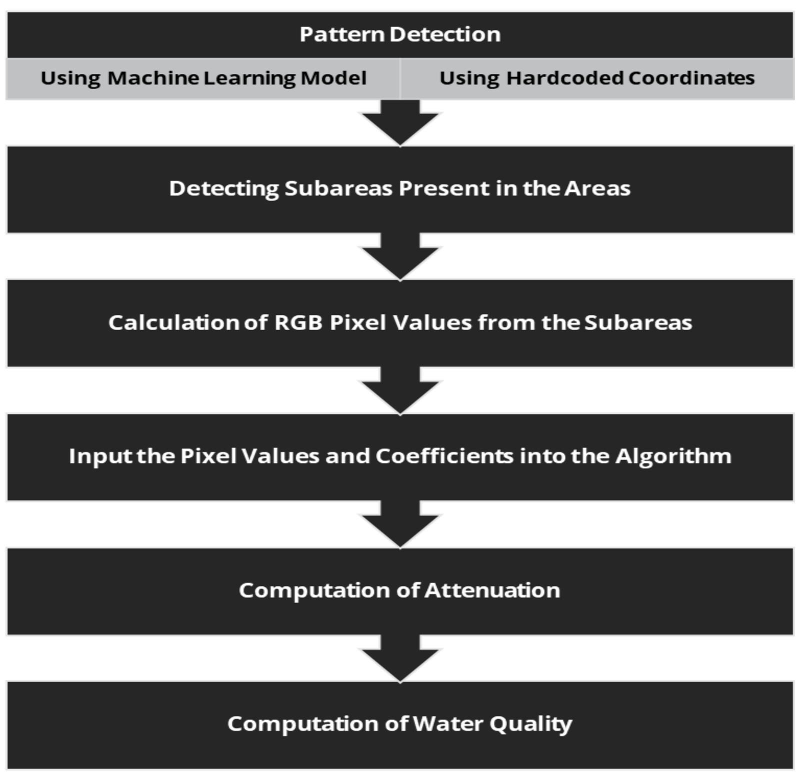
A. Mean of Red and Green Attenuations K_mean_RG = (K_R+K_G)/2
B. Secchi Depth SD = (Secchi_coefficients[0]/K_mean_RG) + Secchi_coefficients[1]
C. Turbidity Turb = Turbidity_coefficients[0]*K_R + Turbidity_coefficients[1]
D. CDOM ratio cdom_ratio = K_B/K_R

E. Absorption by CDOM CDOM = CDOM_coefficients[0]*cdom_ratio + CDOM_coefficients[1]

F. Total Suspended Matter TSM = TSM_coefficients[0]*K_R + TSM_coefficients[1]

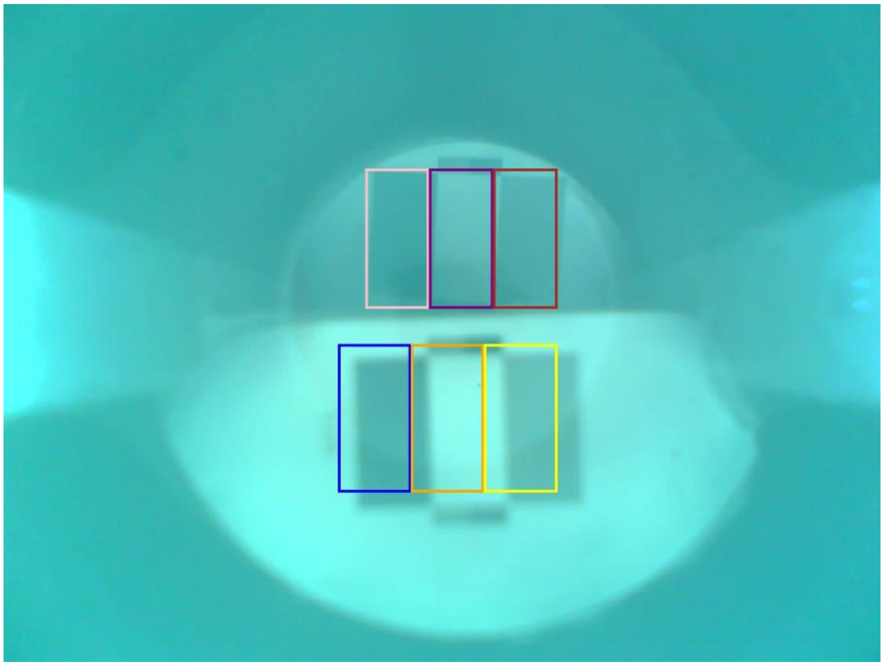
```
K_mean_RG = np.round(np.mean([K_R, K_G]),2) # Mean of red and green attenuations

SD = np.round((Secchi_coefficients[0]/K_mean_RG) + Secchi_coefficients[1],2) # Secchi Depth

Turb = np.round(Turbidity_coefficients[0]*K_R + Turbidity_coefficients[1],2) # Turbidity

cdom_ratio = K_B/K_R

CDOM = CDOM_coefficients[0]*cdom_ratio + CDOM_coefficients[1] #Absorption by CDOM at 400 nm

TSM = TSM_coefficients[0]*K_R + TSM_coefficients[1] #Total suspended matter
```

```
K_mean_RG:  6.89
Secchi Depth:  0.95
Turbidity:  4.91
cdom_ratio:  2.047019062993416
CDOM:  4.589451002718593
Total Suspended Matter: 6.550039792953696
```



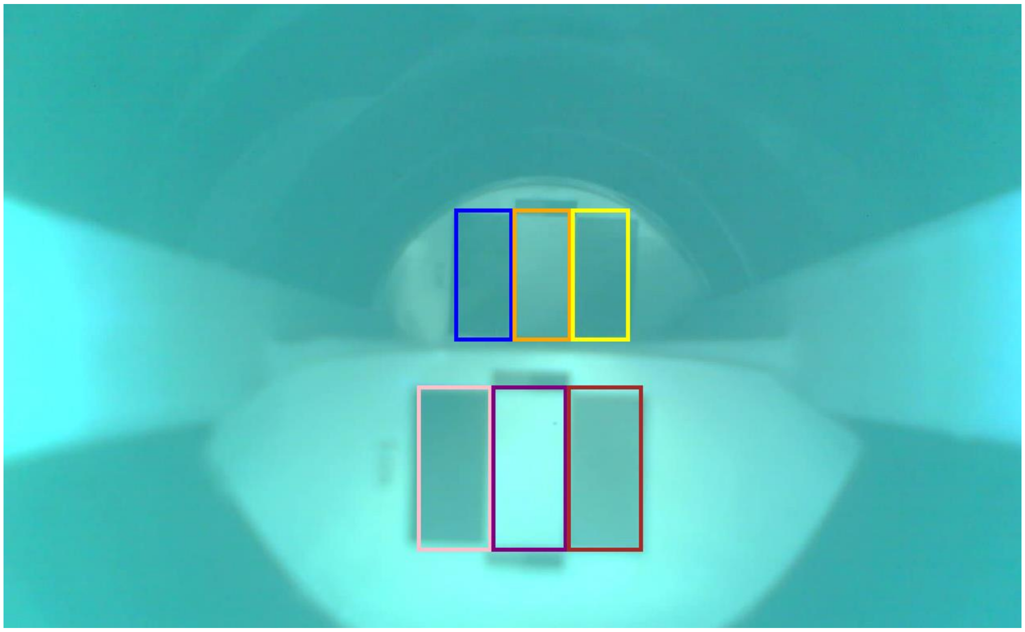
The outputs of some other images are given below,

A. Image 1



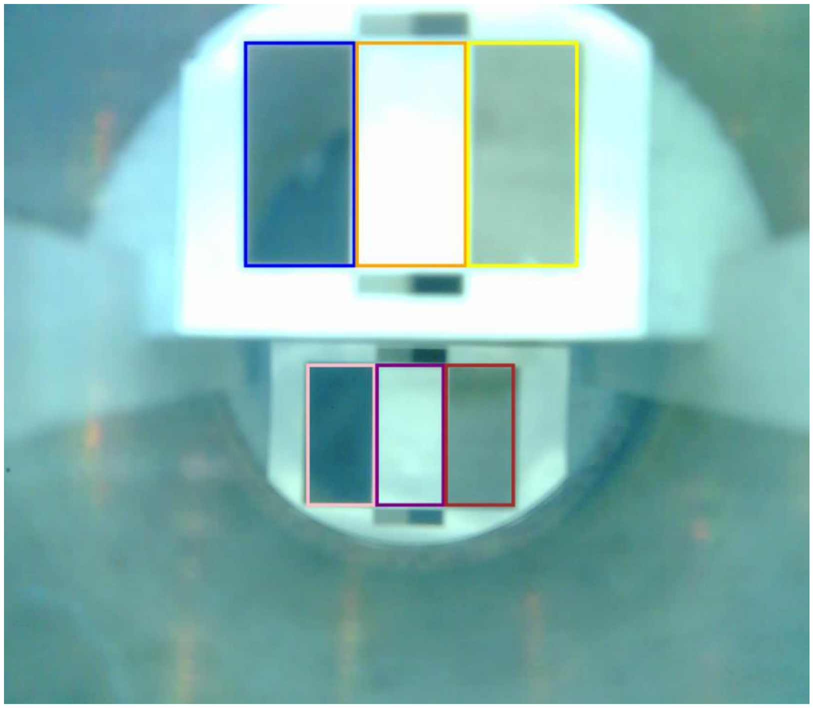
```
K_mean_RG:  8.22
Secchi Depth:  0.67
Turbidity:  3.8
cdom_ratio:  2.9501760744052175
CDOM:  9.336805517503585
Total Suspended Matter: 5.507594239155786
```

B. Image 2



```
K_mean_RG:  -6.99
Secchi Depth:  -2.5
Turbidity:  -6.42
cdom_ratio:  3.1912956979349043
CDOM:  10.604226706625031
Total Suspended Matter: -4.042189112941887
```

C. Image 3



```
K_mean_RG:  -2.73
Secchi Depth:  -5.17
Turbidity:  -3.99
cdom_ratio:  0.8625576554480702
CDOM:  -1.6365519399027635
Total Suspended Matter: -1.7729887929254389
```