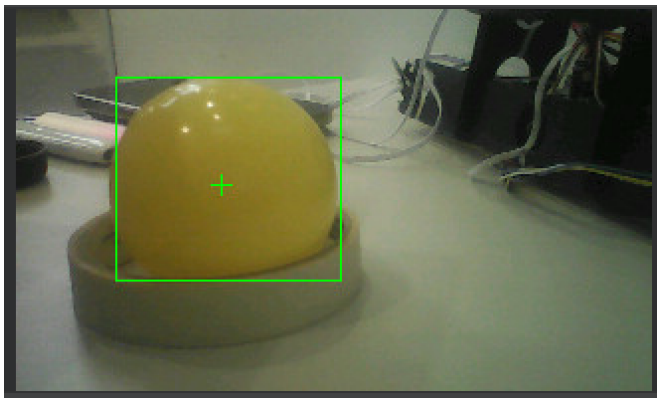
Lab8: Computer Vision

Yiwen(Robert) Wu

https://github.com/Robert1124/cse460

We wrote the program in the OpenMV to prints the coordinates of the center of the blob and its size for the yellow ball.

```
print("Center of blob: u: ",blob.cx())
print("Center of blob: v: ",blob.cy())
blob_radius = max(blob.w(), blob.h()) / 2
blob_size = math.pi * (blob_radius ** 2)
print("Blob size (area): ", blob_size)

#Calculate Distance
distance = ((actual_diameter * focal_len)/ (
    blob_radius * 2 * px_per_mm))
print("Distance: ", distance)
#Calculate Angle
dx = blob.cx() - u_0
angle = (dx / u_0) * (view_angle / 2)
print("Angle: ", angle)
```

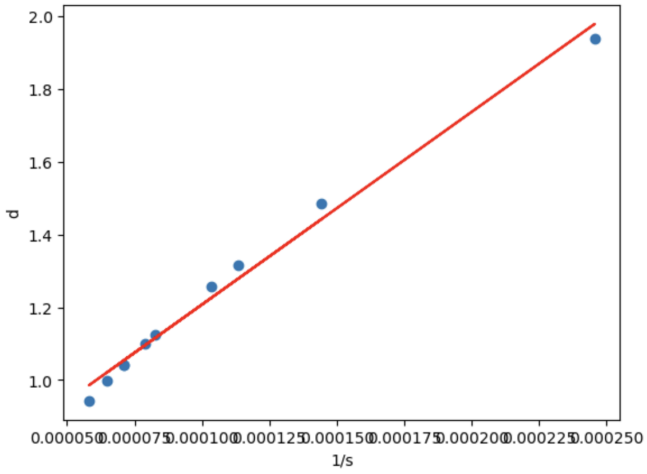


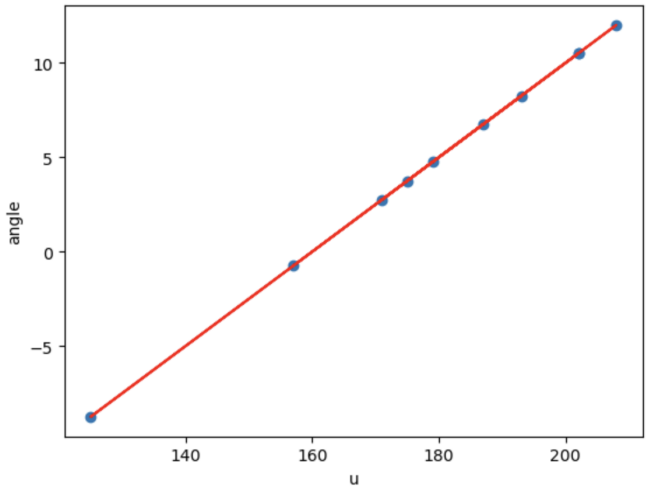
Then, we collected ten different data from different positions:(since we forgot to transfer the unit for the radius, the distance seems a little bit weird)

```
u = [157, 179, 202, 208, 125, 202, 171, 193, 187,
    175]
angle = [-0.75, 4.75, 10.5, 12.0, -8.75, 10.5,
    2.75, 8.25, 6.75, 3.75]
sb = [9676.89, 12076.3, 17203.4, 6939.78, 14102.6,
    8824.73, 4071.5, 14102.6, 15393.8, 12667.7]
distance = [1.25755, 1.12571, 0.943163, 1.48498,
    1.0417, 1.31687, 1.93872, 1.0417, 0.997058,
    1.09912]
```

After that, we plotted two scatter charts in python:

```
sb = [1/x for x in sb]

# 1st plot
coefficients = np.polyfit(sb, distance, 1)
fit_line = np.poly1d(coefficients)

plt.scatter(sb, distance)
plt.xlabel('1/s')
plt.ylabel('d')
plt.plot(sb, fit_line(sb), color='red', label='Fit
    Line')

# 2nd plot
coefficients = np.polyfit(u, angle, 1)
fit_line = np.poly1d(coefficients)

plt.scatter(u, angle)
plt.xlabel('u')
plt.ylabel('angle')
plt.plot(u, fit_line(u), color='red', label='Fit
    Line')
```





Finally, we wrote a function that receives u and s_b and returns the ball's location with respect to the front of the

robot:

```
def approximation(sb, distance, u, angle, s_in,
    u_in):
    coefficients_s = np.polyfit(sb, distance, 1)
    fit_line_s = np.poly1d(coefficients_s)
    predict_d = fit_line_s(1/s_in)

    coefficients_u = np.polyfit(u, angle, 1)
    fit_line_u = np.poly1d(coefficients_u)
    predict_angle = fit_line_u(u_in)

    return predict_d, predict_angle

predict_d,predict_angle = approximation(sb,
    distance, u, angle,12076.3, 170)
print("Predict Distance: ", predict_d)
print("Predict Angle: ", predict_angle)

'''
Predict Distance:  1.1162074062166885
Predict Angle:  2.5000000000000284
'''
```

1. How noisy is the information of the ball? The color range is always needed to be adjusted. In different light condition, it shifts a little bit so the previous values cannot be used again.

2. Would the information be reliable in a search a rescue scenario? Due to the color recognition, the object detection is not always accurate, the information based on that detection may not be very reliable in the search a rescue scenario.

3. How can you improve the detection method? I believe the color detection is not the best method since the light condition is always changing. Since we have a camera in that module, I think a object detection based on the shape of the target and using color to distinguish the target from the background is a better method.