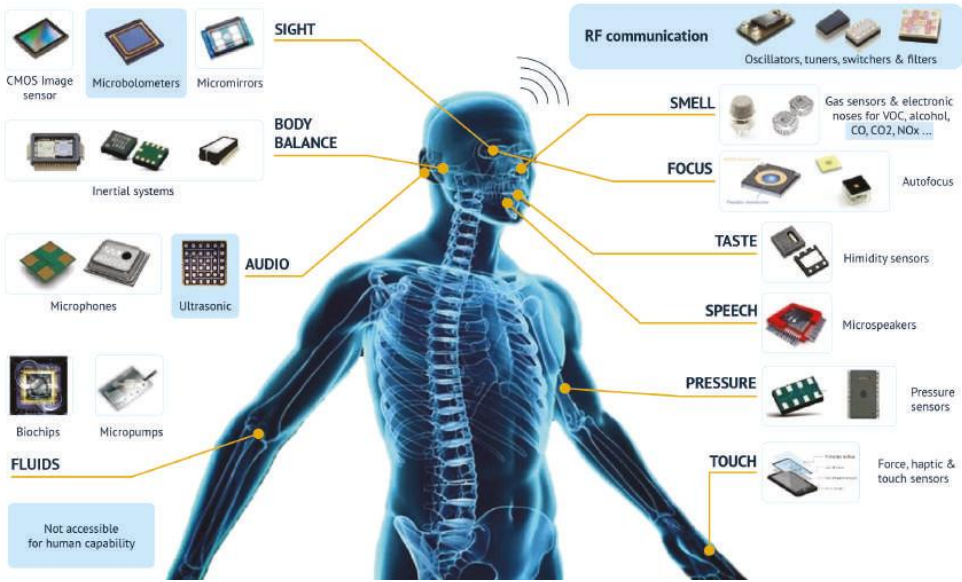
EGR 455 - Robotic Systems 1

Lab 8 – Sensors – Basic Camera Operations



https://www.protolabs.com/media/teghxnha/robotics-hardware-high.jpg

Lab’s Objectives

1) To learn about image color sorting by finding red, blue, and green objects.
2) To learn a way to center a red, blue, and green object.
3) To learn about background subtraction which can be used to find an object of any color.
4) To learn about converting pixels to centimeters.
5) To learn about a way to reduce image noise.
6) To learn about converting the camera frame of reference to the base frame of reference with the use of Homogenous matrices.

Agenda

- Expected Demonstrations
- Expected Submissions
- Tip on Assembling the Camera Stand
- Steps for Adding Open CV
- Steps for Color Sorting
- Steps for Finding the Center of the Object
- Saving Images and Reopening Images
- Steps for Background Subtraction
- Steps for Homogenous Coordinate Transformation Between Camera and Base

Expected Demonstrations

Expected Demonstrations

This is a **non-in-lab** demonstration lab.

The demonstration will be handled **with the submission screen captures** at the **different stages** of this lab.

- You will show you can sort objects by color.
- You will show that you can find the center of each colored object captured by the camera.
- You will show you can do background subtraction.
- You will show that you can convert from pixels to centimeters and do the homogenous transformation from the camera to the base frame.

Expected Submissions

Expected Submissions

A document that has the following images.

- Four images showing your pre-colored sorted image and the Red, Blue, and Green sorted images with a title that includes your last name and what it is you are showing. (i.e. - Nichols-Red Only)
- Three images showing a center mark for each of the Red, Blue, and Green objects with a title that includes your last name and what it is you are showing. (i.e. - Nichols-Center of Blue)

Expected Submissions

- Two images showing the frame image of the pre-background subtraction and post-background subtraction with a title that includes your last name and what it is you are showing. (i.e. - Nichols-Subtraction to Find Object)

Expected Submissions

- Three captured images showing

    1. The object, the base frame location, and the calculated center location.

    2. The computed measurement (in centimeters) within the camera image workspace.

    3. The computed measurement (in centimeters) within the base frame.

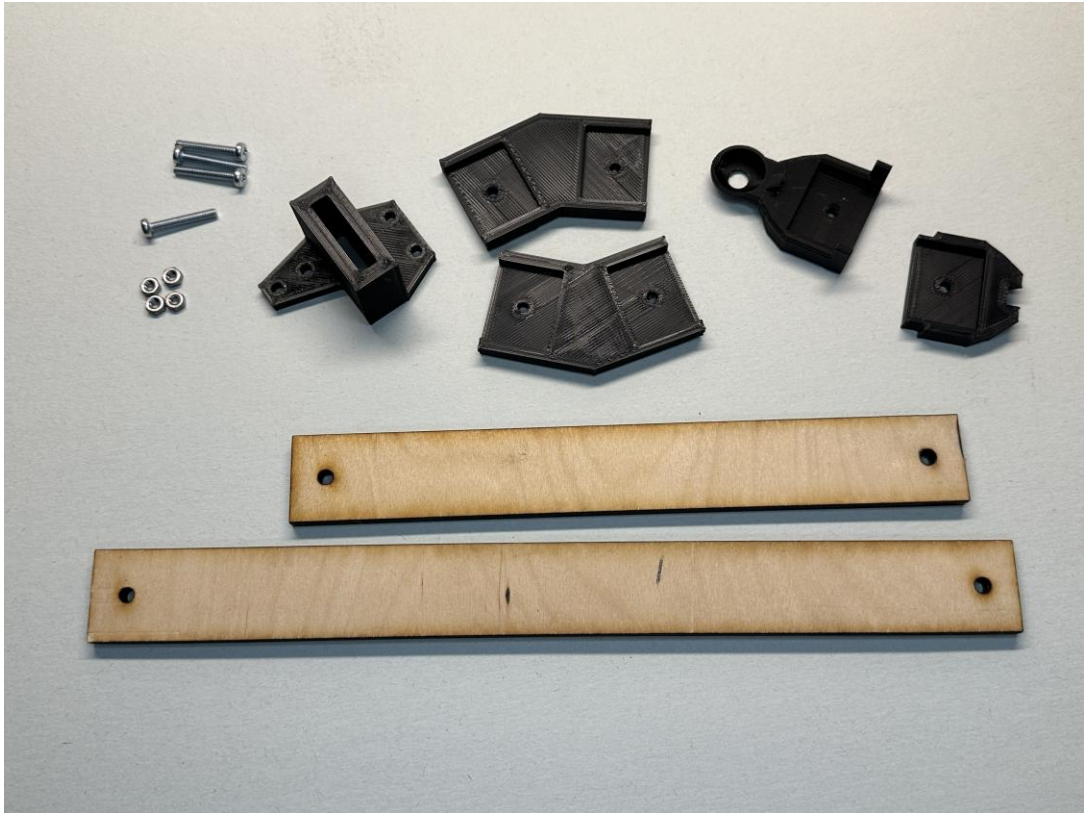
with a title for each image that **includes** your last name and what you are showing. (i.e. - Nichols-Location of the Object Relative to Base Frame)

Tip on Assembling the Camera Stand

Tip on Assembling the Camera Stand

You will need

- Long plywood arm
- Short plywood arm
- 3D printed mounting base bracket
- 2-piece, 3D printed, elbow
- 2-piece, 3D printed, camera mounting bracket
- 4 - M3 x 20 mm screws with nuts



Tip on Assembling the Camera Stand

You will need

- USB web camera - disassembled
- 4 - M3 x 10mm screws and nuts (from the provided kit)
- Robotic base

Tip on Assembling the Camera Stand

- Prep the lower section by inserting the long plywood arm into the slot of the 3D mounting base bracket and hold the two together with a M3 x 20 mm screw and nut.
- Prep the upper section by sandwiching the short plywood arm with the elbow on one end and the camera mounting bracket.

Tip on Assembling the Camera Stand

With the upper section, **note** how the elbow and the cup of the camera mounting bracket are **orientated**.

Tip on Assembling the Camera Stand

- With the camera clip screw removed so that there is a clip, a small screw, and a plastic disk are free from the rest of the camera.
- Insert the post on the camera up through the hole of the 3D-printed camera mount so that it sticks up into the concaved disk, as shown here.

Tip on Assembling the Camera Stand

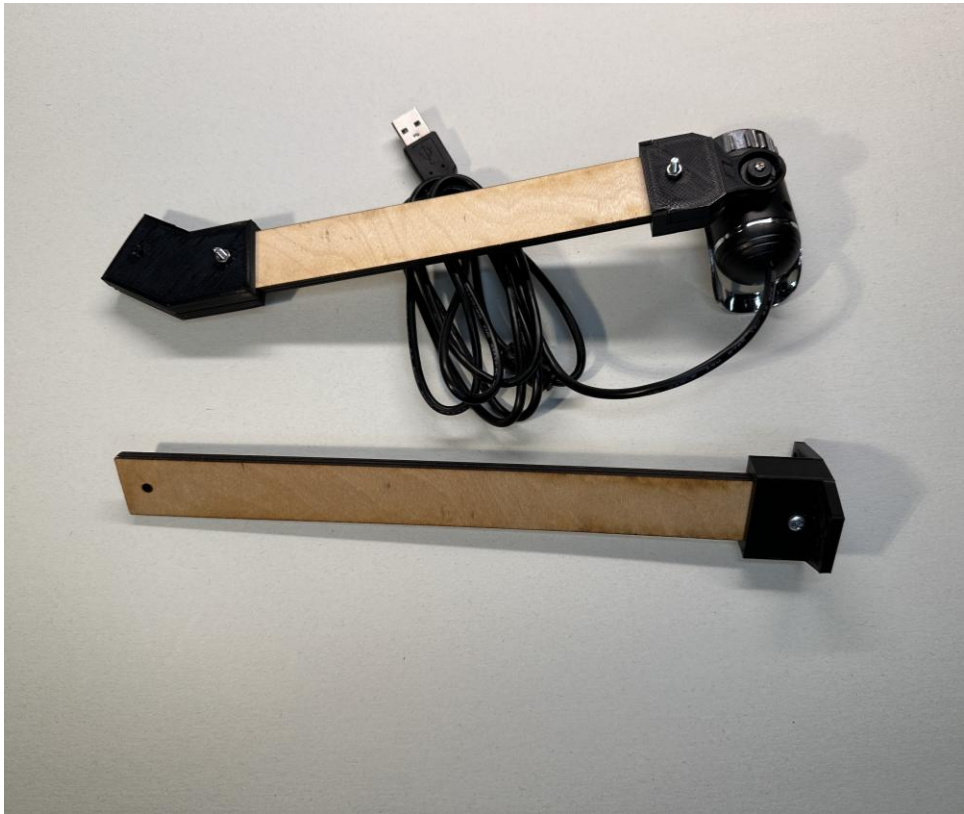
- Now place and align the plastic disk back onto the camera’s post. There are flat spots that will lock their position together.

Tip on Assembling the Camera Stand

- Re-attach the small screw onto the camera’s post to hold the disk. **Do not over-tighten.** You want the screw tight enough to keep the camera held in position but not so tight to prevent positioning of the camera or damaging the plastic that has the screw.
- Once the screw is re-inserted. Do not remove.

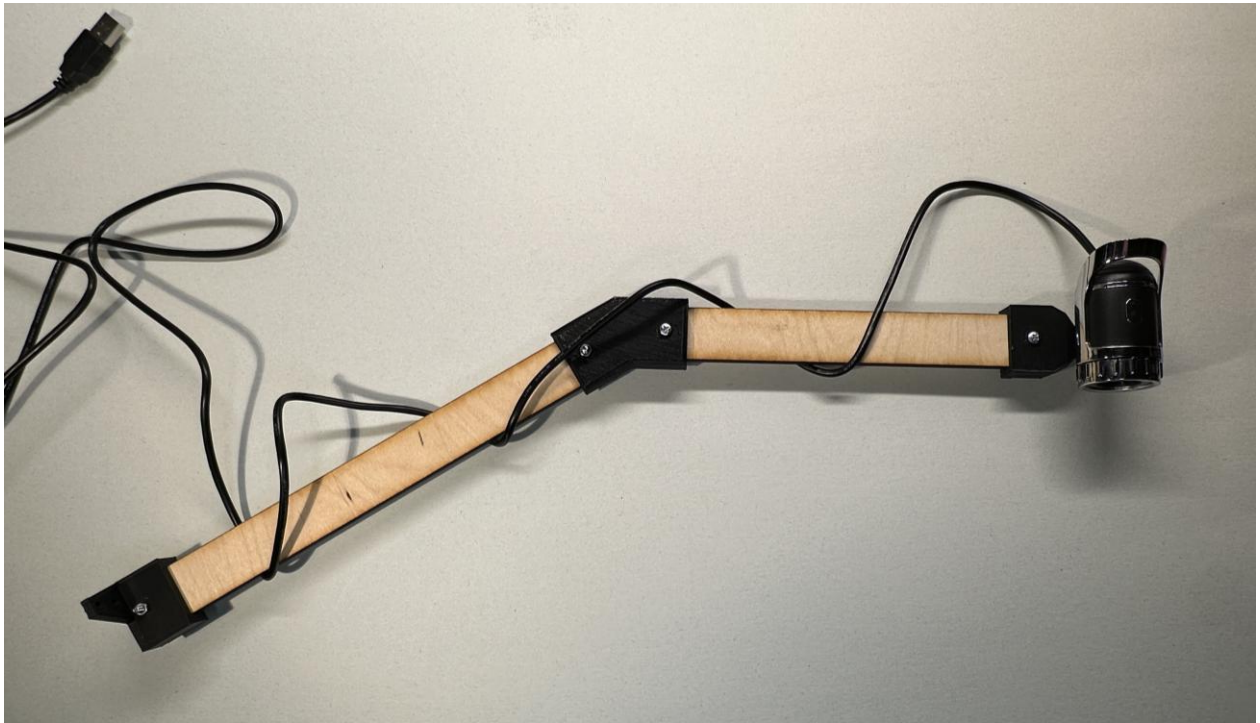
Tip on Assembling the Camera Stand

With the lower and upper sections assembled, this is a state where you can easily set up or store the camera stand.



Tip on Assembling the Camera Stand

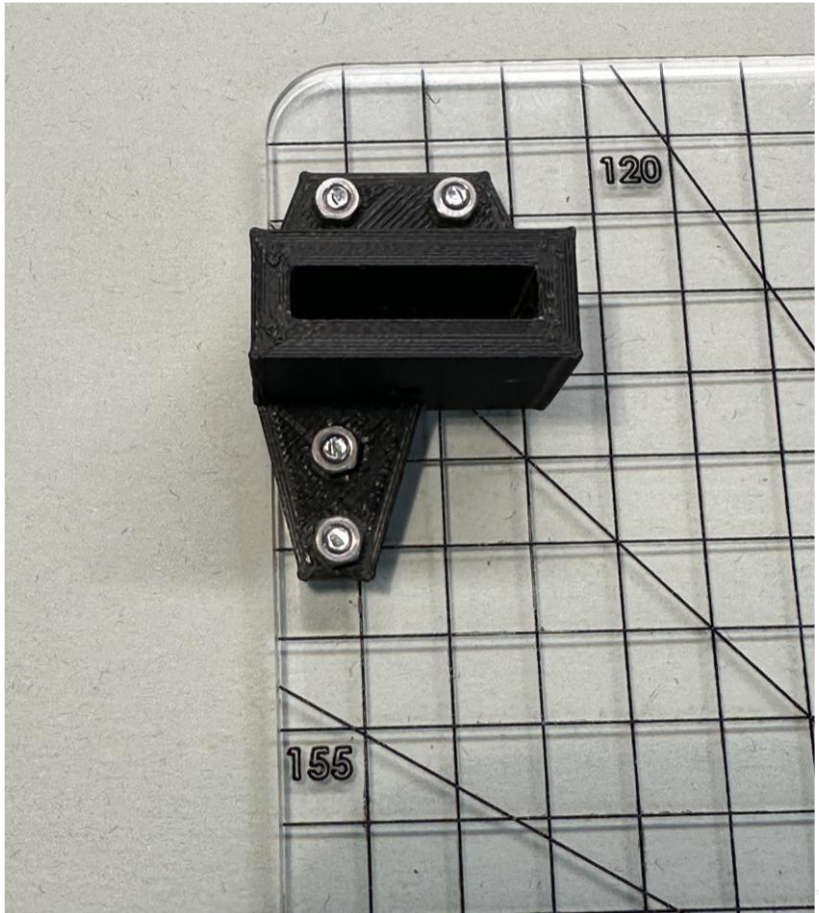
When ready to use, connect the two sections together and wrap the USB cable around the arm pieces.



Tip on Assembling the Camera Stand

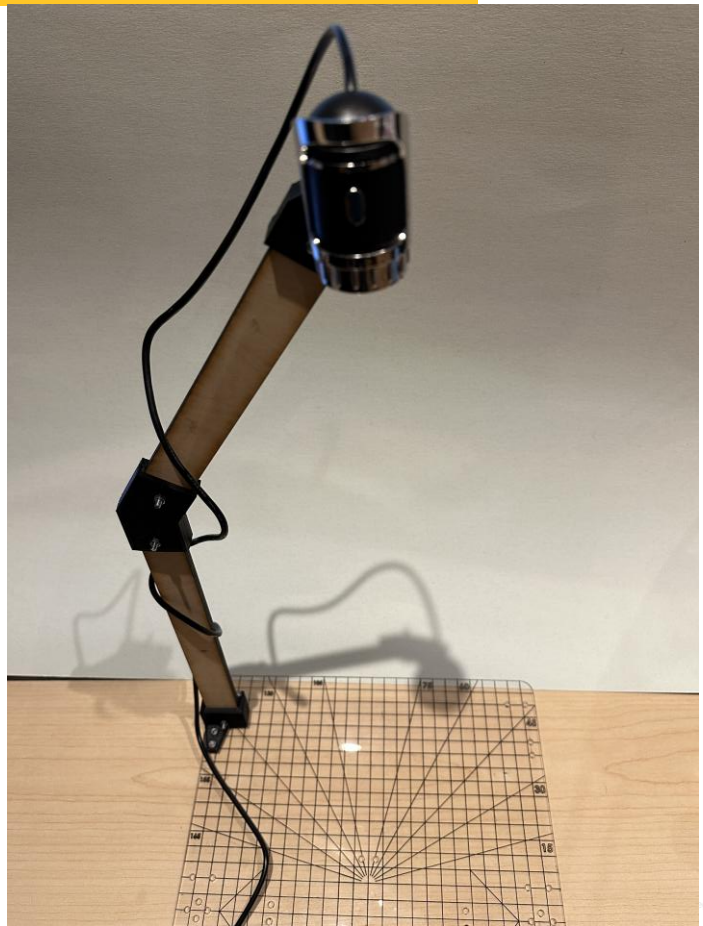
To secure the camera stand to the base, use four M3 x 10 mm screws and nuts in the four holes in the upper left corner of the base plate.

The image shown here is without the assembled stand, so that is easier to show how to attach it.



Tip on Assembling the Camera Stand

If assembled and mounted to the base plate your setup will look similar to what is shown here.



Steps for Adding Open CV via PIP

Steps for Adding Open CV

- Open Command Line (cmd) or Terminal
- Type “python –version” and hit enter/return
    - It should return the version of Python loaded on the computer
- Type “pip install opencv-python” for Main modules package
- or type “pip install opencv-contrib-python” for the Full package (contains both main modules and contrib/extra modules)

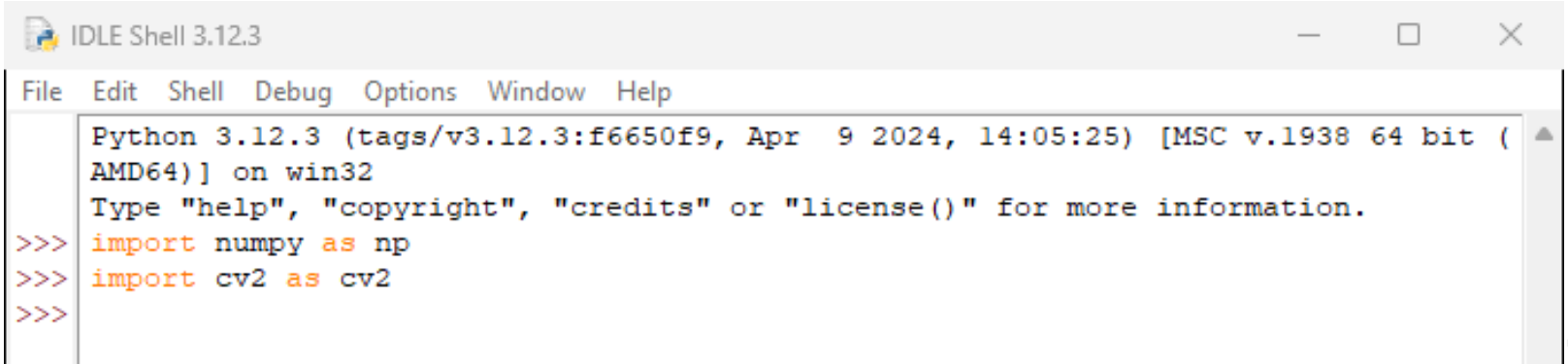
```
PS C:\Users\kwnicho> pip install opencv-python
Collecting opencv-python
  Using cached opencv_python-4.10.0.84-cp37-abi3-win_amd64.whl.metadata (20 kB)
Requirement already satisfied: numpy>=1.21.2 in c:\users\kwnicho\appdata\local\programs\python\python3
12\lib\site-packages (from opencv-python) (2.0.0)
Using cached opencv_python-4.10.0.84-cp37-abi3-win_amd64.whl (38.8 MB)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.10.0.84
```

Steps for Adding Open CV

Now, to check the install of OpenCV

- Open IDLE Python and in the IDLE Shell
- type “import numpy as np” because OpenCV uses part of NumPy
- On the next line, type “import cv2 as cv2”

If you get no errors, then OpenCV has successfully been installed.



Steps for Adding Open CV

Let start with some basic code.

EGR_455_Lab08-Basic_Camera.py - C:/Users/kwnicho/ASU Dropbox/Kevin Nichols/Robotics/0...

File Edit Format Run Options Window Help

```
import numpy as np
import cv2

cap = cv2.VideoCapture(0) ##Set a varible to hold the video from camera #? 0,1,2

while (1):
    _,frame = cap.read()

    ## to show frame picture in a window with a title
    cv2.imshow('LastName-Frame',frame)

    ## a way to break from the loop
    k = cv2.waitKey(5) ## the number is the ms delay to check if

    ## the number '27' is the ASCII control character value for the esc key
    if k == 27:
        break

cv2.destroyAllWindows() ## to close windows python opened
```

Steps for Adding Open CV

Run this basic starting point code.

- If you do not get the camera you want, change the number within the parentheses of “VideoCapture.” Depending on the number of camera connected or part of the computer will determine the range in numbers. (i.e. a built-in camera and USB camera means there are two choices, 0 or 1)
- If the camera is out of focus, so web camera can have there focus adjusted by turning the lens slightly.

Steps for Color Sorting

Steps for Color Sorting

Now, adjust the code to sort and display the results.

When you run the code, to exist the inifinate for loop, press the “**esc**” key.

EGR_455_Lab08-Basic_Camera.py - C:/Users/kwnicho/ASU Dropbox/Kevin Nichols/Robotics/...

File Edit Format Run Options Window Help

```
import numpy as np
import cv2

cap = cv2.VideoCapture(0)

while (1):
    _,frame = cap.read()

    red = frame[:,:,2] ## all the row, all the colmuns, the third layer
    green = frame[:,:,1] ## all the row, all the colmuns, the second layer
    blue = frame[:,:,0] ## all the row, all the colmuns, the first layer


    cv2.imshow('LastName-RGB',frame)
    cv2.imshow('LastName-Red Layer',red)
    cv2.imshow('LastName-Green Layer',green)
    cv2.imshow('LastName-Blue Layer',blue)


    ## a way to break from the loop
    k = cv2.waitKey(5) ## the number is the ms delay to check if

    ## the number '27' is the ASCII control character value for the esc key
    if k == 27:
        break

cv2.destroyAllWindows() ## to close windows python opened
```

Steps for Color Sorting

Notice that you will have on each color layer there is some amount of each colored object. To get it so that you have only one color without some of the other colors, you will need to do what is called image subtraction.

Return to the Python code and modify it so you do color subtraction to find Red Only.

Steps for Color Sorting

*EGR_455_Lab08-Basic_Camera.py - C:/Users/kwnicho/ASU Dropbox/Kevin Nichols/Robotics...

```
cap = cv2.VideoCapture(0)

while (1):
    _,frame = cap.read()

    ## add np.matrix to ensure they are treated as matrics
    red = np.matrix(frame[:,:,2])
    green = np.matrix(frame[:,:,1])
    blue = np.matrix(frame[:,:,0])

    ## make value a 16  bit value that can be positive or negative
    red_only = np.int16(red) - np.int16(green) - np.int16(blue)

    ## to numbers to achive valid matrix ranging from 0 to 255
    red_only[red_only < 0] = 0 ## sets numbers lower than zero to zero
    red_only[red_only > 255] = 255 ## sets numbers greater than 255 to 255

    red_only = np.uint8(red_only) ## to ensure red_only is a unsigned interger

    cv2.imshow('LastName-RGB',frame)
    cv2.imshow('LastName-Red Layer',red)
    cv2.imshow('LastName-Green Layer',green)
    cv2.imshow('LastName-Blue Layer',blue)


    cv2.imshow('LastName-Red Only',red_only)

    ## a way to break from the loop
    k = cv2.waitKey(5) ## the number is the ms delay to check if

    ## the number '27' is the ASCII control character value for the esc key
    if k == 27:
        break

cv2.destroyAllWindows() ## to close windows python opened
```

Ln: 16 Col: 1

Steps for Color Sorting

Now when you run the code, the “Red Only” image should show a bright white where the red object is and everything else is black.

Go back and add code to be able to show the blue and green only.

Image is of the red only with the upcoming finding the center of red object.

Steps for Finding the Center of the Object

Steps for Finding the Center of the Object

The following additional code will find the center of the red only. The center in the column direction of the image is also the ‘X’ pixel value and the center in the row direction is the ‘Y” in units of pixels.

```
27
28      ## how to find the center of red object in 'X' of the carmera image
29      column_sums = np.matrix(np.sum(red_only,0))##'0' to sum the columns
30      column_numbers = np.matrix(np.arange(640))
31      column_mult = np.multiply(column_sums,column_numbers)#element wise multiptation
32      column_total = np.sum(column_mult)
33      total_total = np.sum(np.sum(red_only))# sum of all values in the image
34      red_column_location = column_total/total_total
35
36      print('Red column ("x") location: ',red_column_location)
37
38      ## how to find the center of red object in 'X' of the carmera image
39      row_sums = np.matrix(np.sum(red_only,1))
40      row_sums = row_sums.transpose()
41      row_numbers = np.matrix(np.arange(480))
42      row_mult = np.multiply(row_sums,row_numbers)
43      row_total = np.sum(row_mult)
44      red_row_location = row_total/total_total
45
46      print('Red row ("Y") location: ',red_row_location)
47      red_only = np.uint8(red_only) # to ensure red_only is a unsigned interger
48
```

Steps for Finding the Center of the Object

When you run this code and move a red object to the left of the camera's frame, the printed value will get smaller—approaching zero.

When the red object is moved to the right of the camera's frame, the printed value will get larger—approaching the maximum value of 640.

Use this to modify your code to get the centers of also the blue and green location.

Saving Images and Reopening Images

Saving Images and Reopening Images

To help capature the different images needed for submission, you might want to reopen the image or save it with text. Use the following code to do both.

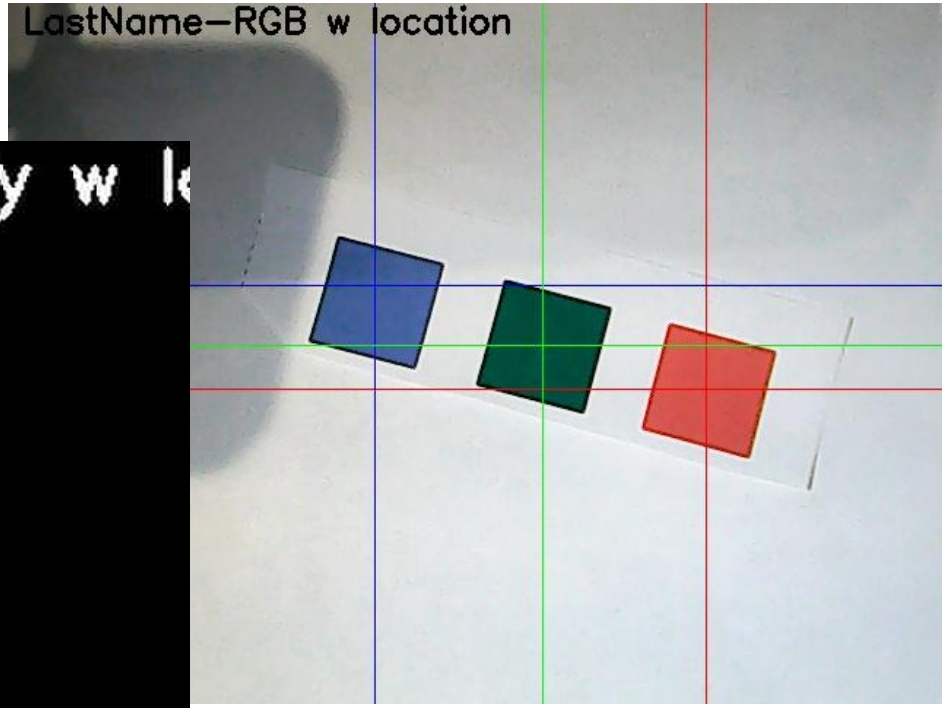
EGR_455_Lab08-Camera_Color_Sorting(full).py - C:\Users\kwnicho\ASU Dropbox\Kevin Nichols\Robotics\03 -

File Edit Format Run Options Window Help

```
import numpy as np
import cv2
import os

cap = cv2.VideoCapture(1)

while (1):
    _,frame = cap.read()

    ## add np.matrix to ensure they are treated as matrics
```

```
    k = cv2.waitKey(5) ## the number is the ms delay to check if

    ## the number '27' is the ASCII control character value for the esc key
    if k == 27:
        break

## out of "for" loop by removing tab indent
cv2.destroyAllWindows() ## to close windows python opened


## Draw the "cross hairs" across the "frame" image as a form of verification of location
frame = cv2.line(frame, (red_x_location, 0), (red_x_location, 480), (0,0,255), thickness)
frame = cv2.line(frame, (0, red_y_location), (640, red_y_location), (0,0,255), thickness)
frame = cv2.line(frame, (blue_x_location, 0), (blue_x_location, 480), (255,0,0), thickness)
frame = cv2.line(frame, (0, blue_y_location), (640, blue_y_location), (255,0,0), thickness)
frame = cv2.line(frame, (green_x_location, 0), (green_x_location, 480), (0,255,0), thickness)
frame = cv2.line(frame, (0, green_y_location), (640, green_y_location), (0,255,0), thickness)
print('Red image size: ',np.shape(red_only))

# Option to re-open images windows to review set position or to capture
cv2.imshow('LastName-RGB',frame)
cv2.imshow('LastName-Red Only',red_only)

## Option to Save images
 # Image directory (if you want a select location, otherwise
 # it is save to the location of the Python code)
directory = r'C:\Users\kwnicho\Desktop\PythonImages'
os.chdir(directory)
# Filename
filename = 'LastName-Red Only w location.jpg'

# Using cv2.imwrite() method to save image
font_color = (255,255,255)
font = cv2.FONT_HERSHEY_SIMPLEX
font_size = 0.8
font_thickness = 2
red_text = 'LastName-Red Only w location'
blue_text = 'LastName-Blue Only w location'
green_text = 'LastName-Green Only w location'
rgb_text = 'LastName-RGB w location'
x,y = 10,20

red_only = cv2.putText(red_only, red_text, (x,y), font, font_size, font_color, font_thickness)
blue_only = cv2.putText(blue_only, blue_text, (x,y), font, font_size, font_color, font_thickness)
green_only = cv2.putText(green_only, green_text, (x,y), font, font_size, font_color, font_thickness)
frame = cv2.putText(frame, rgb_text, (x,y), font, font_size, (0,0,0), font_thickness)

cv2.imwrite(filename, red_only)
cv2.imwrite('LastName-RGB w location.jpg', frame)
cv2.imwrite('LastName-Blue Only w location.jpg', blue_only)
cv2.imwrite('LastName-Green Only w location.jpg', green_only)

Ln: 18
```

Saving Images and Reopening Images



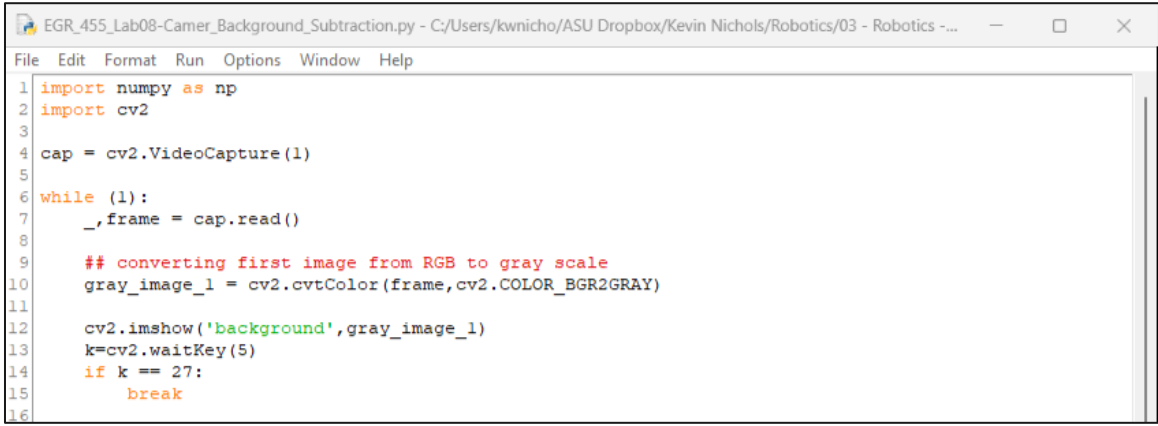
Steps for Background Subtraction

Steps for Background Subtraction

Start by either copying or creating a new python code file.

I will be naming this file something like background_subtraction.

Then edit the “while” loop as shown.



Steps for Background Subtraction

Now that you have a background image that is captured when you hit the “esc” key, copy or add the following second “while” loop.

```
        break

while (1):
    _,frame = cap.read()

    ## converting first image from RGB to gray scale
    gray_image_2 = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)

    cv2.imshow('foreground',gray_image_2)

    # To find the difference between the two ( Note: need to ensure values are
    # signed appropriately and are matrices before subtracting)
    ## since the object could be light or dark, we will take the absolute value of the difference
    Difference = np.absolute(np.matrix(np.int16(gray_image_1))-np.matrix(np.int16(gray_image_2)))

    ## 'cleaning' the numbers to achive valid image matrix ranging from 0 to 255
    Difference[Difference > 255] = 255 ## sets numbers greater than 255 to 255
    #Note: Do not need to address the negative values because that was handled with the absolute

    ## Now to make the Difference matrix an unsigned 8 bit interger
    Difference = np.uint8(Difference)

    cv2.imshow('difference',Difference)

    k=cv2.waitKey(5)
    if k == 27:
        break
```

This loop capture a foreground image and create a difference image.

Steps for Background Subtraction

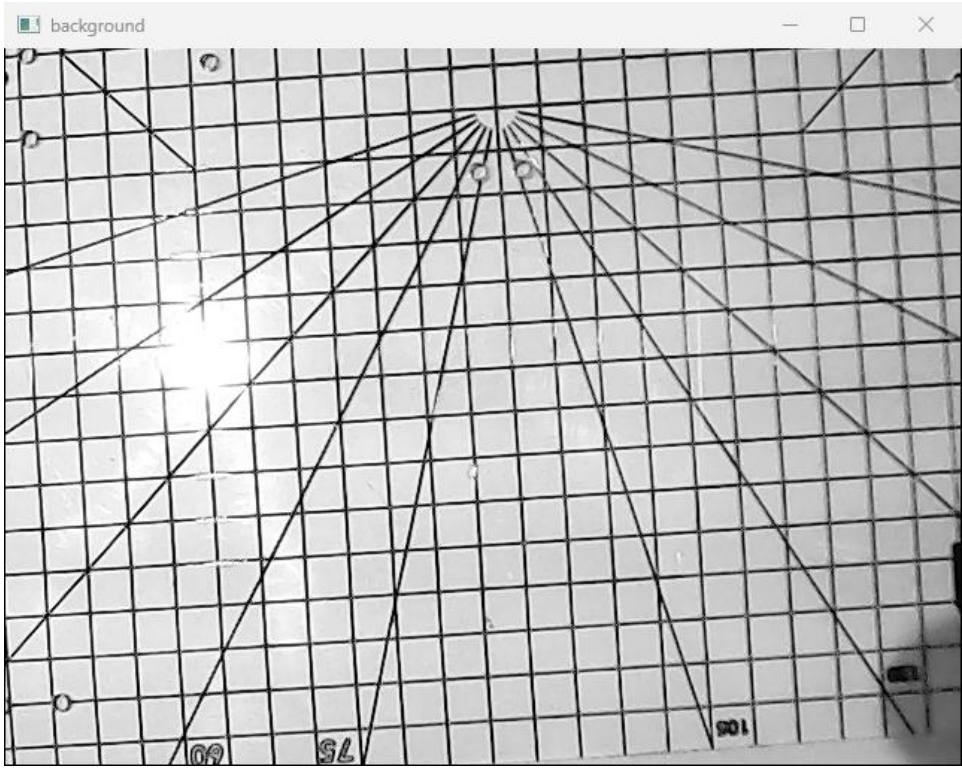
After the user exist out of the second “while,” add the code to find the center of the object.

```
        break

## Now to find the center of the object in 'X' dimension of the carmera image
column_sums = np.matrix(np.sum(Difference,0))##'0' to sum the columns
column_numbers = np.matrix(np.arange(640))
column_mult = np.multiply(column_sums,column_numbers)#element wise multiptation
column_total = np.sum(column_mult)
total_total = np.sum(np.sum(Difference))# sum of all values in the image
difference_column_location = column_total/total_total

print('Object column ("X") location: ',difference_column_location)

## how to find the center of the object in 'Y' of the carmera image
row_sums = np.matrix(np.sum(Difference,1))
row_sums = row_sums.transpose()
row_numbers = np.matrix(np.arange(480))
row_mult = np.multiply(row_sums,row_numbers)
row_total = np.sum(row_mult)
difference_row_location = row_total/total_total

print('Object row ("Y") location: ',difference_row_location)

#########################################################################
# Draw the "cross hairs" across "Difference" image as a form of verification of location
# Line color in BGR code
color = (255, 255, 255)# white
# Line thickness in pixels
thickness = 1

# formating the found locations so that it works with draw line
diff_x_location = np.uint16(difference_column_location)
diff_y_location = np.uint16(difference_row_location)

# First line is the verical line and the second is the horizonital
Difference = cv2.line(Difference, (diff_x_location, 0), (diff_x_location, 480), color, thickness)
Difference = cv2.line(Difference, (0, diff_y_location), (640, diff_y_location), color, thickness)

cv2.imshow('difference',Difference)
```

Steps for Background Subtraction

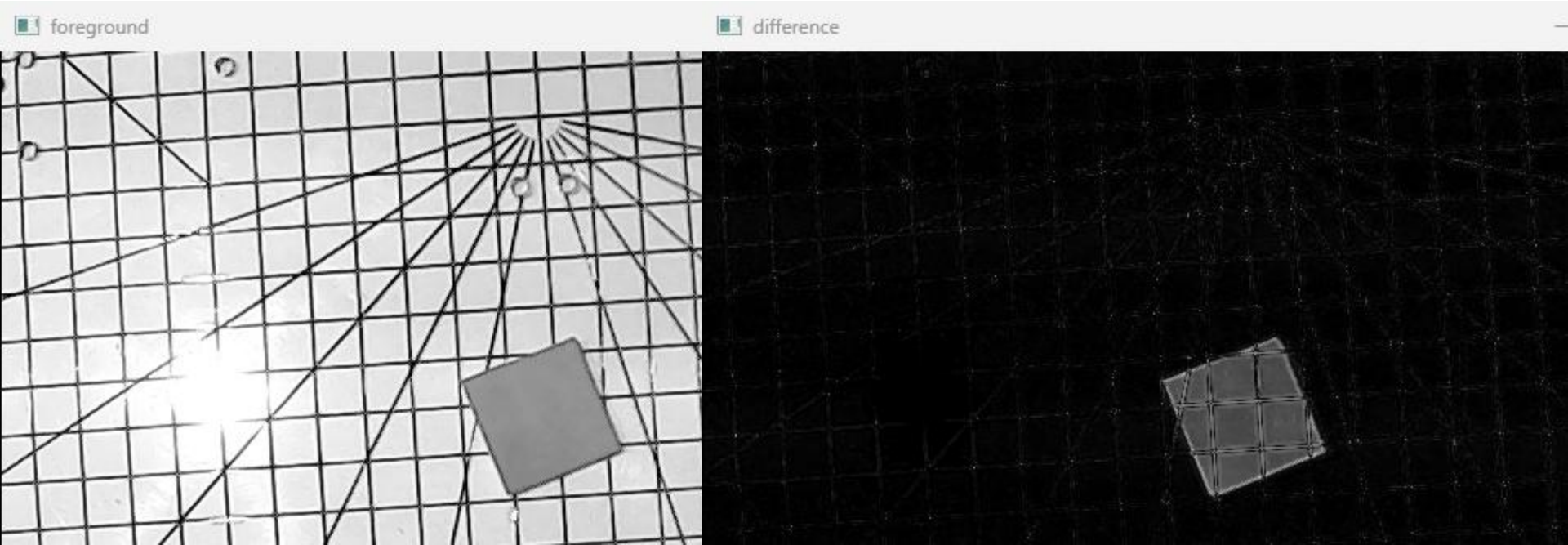
- With the base cleared of objects, run a test of the code module.
- The first shown is the background image, which should look something like this in a gray scale.
- When you have a steady image, hit “esc” to move forward in the code.

** While running, try to keep the camera still to prevent a shaky image and poor results.



Steps for Background Subtraction

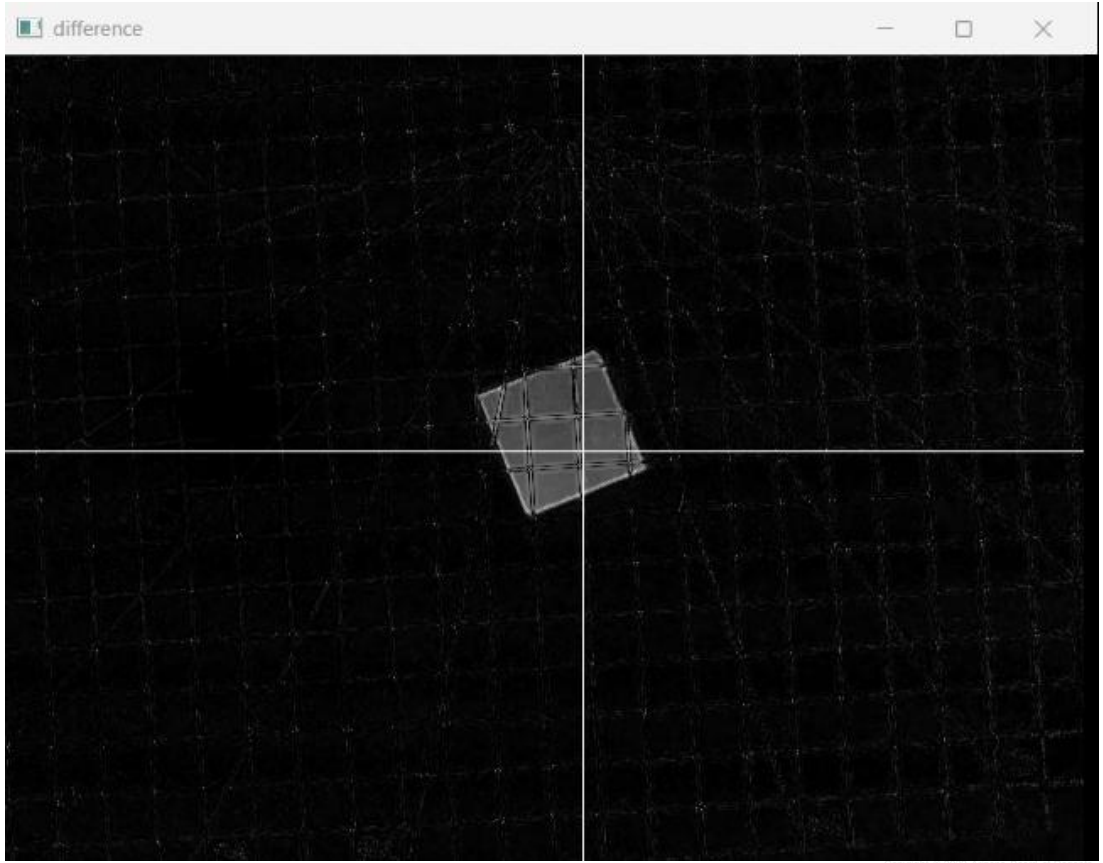
Now place a object into view. Notice the difference image still shows the base's grid marking under the object. There is also some base line still showing.



Steps for Background Subtraction

Now hit “esc” again to break out of the second “while” loop and see the results of finding the center of the object.

Now notice it is not accurately centered on the object.



Steps for Background Subtraction

To fix this, let us now add the method of "Thresholding" to the code/module to convert the image from a gray scale image to a black and white image.

Added

Changed for now BW image

EGR_455_Lab08-Camer_Background_Subtraction.py - C:/Users/kwnicho/ASU Dropbox/Kevin Nichols/Robotics/03 - Robotics - Labs/Lab Codes/Python Codes/EGR_455_Lab08-C...

File Edit Format Run Options Window

```
31      Difference[Difference > 255] = 255 ## sets numbers greater than 255 to 255
32      #Note: Do not need to address the negative values because that was handled with the absolute
33
34      ## Now to make the Difference matrix an unsigned 8 bit interger
35      Difference = np.uint8(Difference)
36      Saved_Diff = Difference
37      cv2.imshow('difference',Saved_Diff)
38
39      ##Thresholding
40      BW=Difference
41      BW[BW<=100]=0 #have to do this first to prevent problems by the following step
42      BW[BW>100]=255
43
44      ## Now to find the center of the object in 'X' dimension of the carmera image
45      column_sums = np.matrix(np.sum(BW,0))##'0' to sum the columns
46      column_numbers = np.matrix(np.arange(640))
47      column_mult = np.multiply(column_sums,column_numbers)#element wise multiptation
48      column_total = np.sum(column_mult)
49      total_total = np.sum(np.sum(BW))# sum of all values in the image
50      BW_column_location = column_total/total_total
51      BW_column_location = np.nan_to_num(BW_column_location, nan=0)
52
53      print('Object column ("X") location: ',BW_column_location)
54
55      ## how to find the center of the object in 'Y' of the carmera image
56      row_sums = np.matrix(np.sum(BW,1))##'1' to sum the rows
57      row_sums = row_sums.transpose()
58      row_numbers = np.matrix(np.arange(480))
59      row_mult = np.multiply(row_sums,row_numbers)
60      row_total = np.sum(row_mult)
61      BW_row_location = row_total/total_total
62      BW_row_location = np.nan_to_num(BW_row_location, nan=0)
63
64      print('Object row ("Y") location: ',BW_row_location)
65
66      ######################################################################
67      # Draw the "cross hairs" across "foreground" image as a form of verification of location
68      # Line color in BGR code
69      color = (255,255,255)# white
70      # Line thickness in pixels
71      thickness = 1
72
73      # formating the found locations so that it works with draw line
74      diff_x_location = np.uint16(BW_column_location)
75      diff_y_location = np.uint16(BW_row_location)
76
77      # First line is the verical line and the second is the horizonital
78      BW = cv2.line(BW, (diff_x_location, 0), (diff_x_location, 480), color, thickness)
79      BW = cv2.line(BW, (0, diff_y_location), (640, diff_y_location), color, thickness)
80
81      cv2.imshow('Difference w/ Center & Threshold',BW)
82
```

Steps for Background Subtraction

To fix this, let us now add the method of "Thresholding" to the code/module to convert the image from a gray scale image to a black and white image.

Changed for now BW row and Column location and draw lines on the BW image

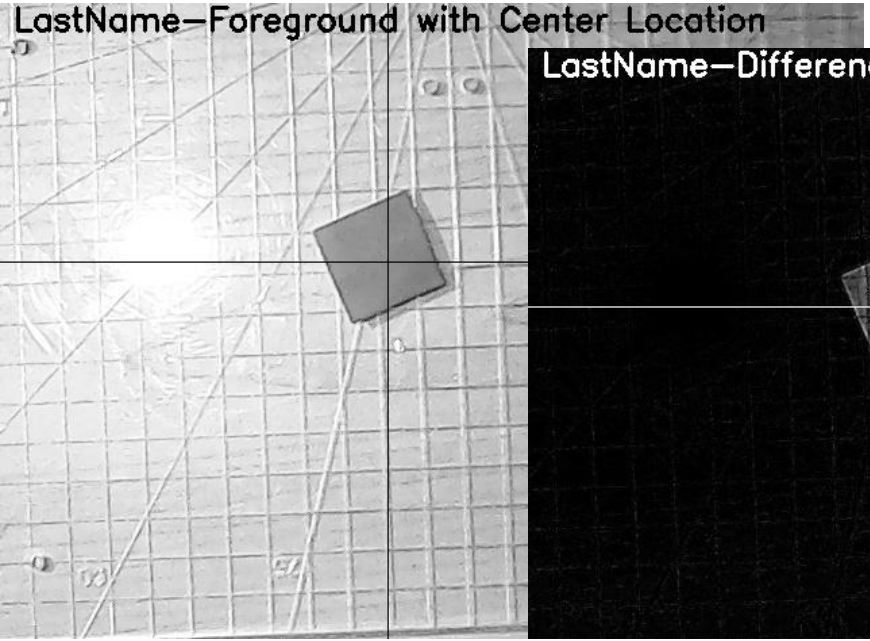
EGR_455_Lab08-Camer_Background_Subtraction.py - C:/Users/kwnicho/ASU Dropbox/Kevin Nichols/Robotics/03 - Robotics - Labs/Lab Codes/Python Codes/EGR_455_Lab08-C...

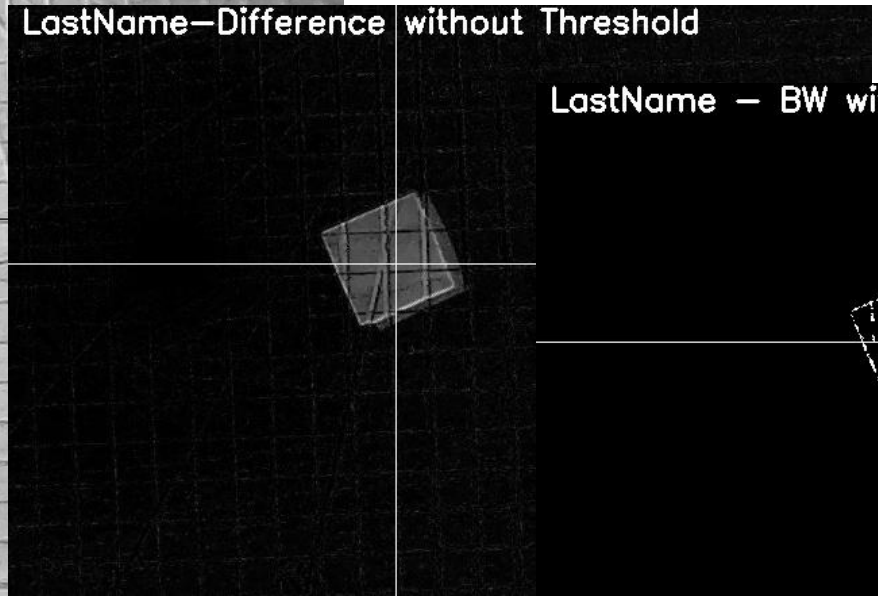
File Edit Format Run

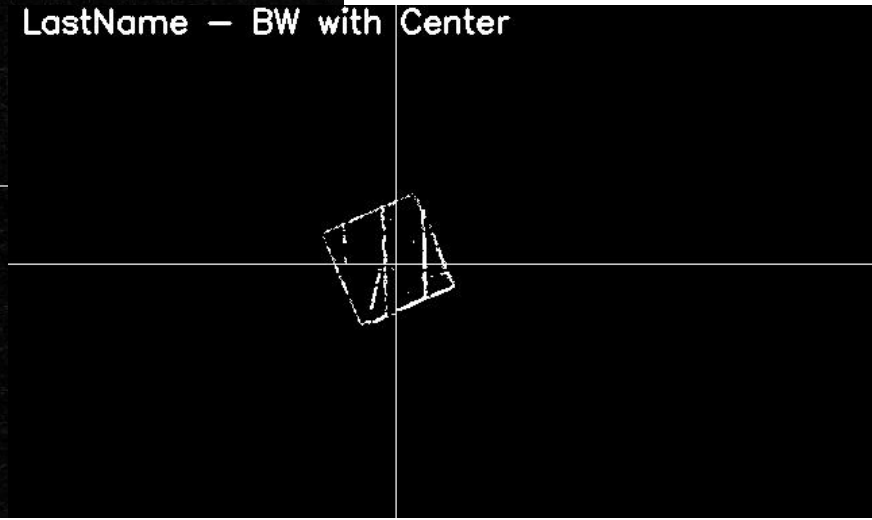
```
    column_numbers = np.matrix(np.arange(640))
    column_mult = np.multiply(column_sums,column_numbers)#element wise multiptation
    column_total = np.sum(column_mult)
    total_total = np.sum(np.sum(BW))# sum of all values in the image
    BW_column_location = column_total/total_total
    BW_column_location = np.nan_to_num(BW_column_location, nan=0)

    print('Object column ("X") location: ',BW_column_location)

    ## how to find the center of the object in 'Y' of the carmera image
    row_sums = np.matrix(np.sum(BW,1))##'1' to sum the rows
    row_sums = row_sums.transpose()
    row_numbers = np.matrix(np.arange(480))
    row_mult = np.multiply(row_sums,row_numbers)
    row_total = np.sum(row_mult)
    BW_row_location = row_total/total_total
    BW_row_location = np.nan_to_num(BW_row_location, nan=0)

    print('Object row ("Y") location: ',BW_row_location)

    ########################################################################
    # Draw the "cross hairs" across "foreground" image as a form of verification of location
    # Line color in BGR code
    color = (255,255,255)# white
    # Line thickness in pixels
    thickness = 1

    # formating the found locations so that it works with draw line
    diff_x_location = np.uint16(BW_column_location)
    diff_y_location = np.uint16(BW_row_location)

    # First line is the verical line and the second is the horizonital
    BW = cv2.line(BW, (diff_x_location, 0), (diff_x_location, 480), color, thickness)
    BW = cv2.line(BW, (0, diff_y_location), (640, diff_y_location), color, thickness)

    cv2.imshow('Difference w/ Center & Threshold',BW)

    k=cv2.waitKey(5)
    if k == 27:
        break

## out of "for" loop by removing tab indent
```

Steps for Background Subtraction

Now when you run the module/code you with have results similar to these.







Steps for Background Subtraction

Or like this with a M3 nut from your kit.

Steps for Background Subtraction

To save the difference image without Tredholding. I had to save and then later read the image back in.

```
39
40      ## Now to make the Difference matrix an unsigned 8 bit interger
41      Difference = np.uint8(Difference)
42      Saved_Diff = Difference
43      cv2.imwrite('LastName-Difference.jpg', Saved_Diff)
44      cv2.imshow('difference',Saved_Diff)
45
46      ##Thresholding
47      BW=Difference
```

The other images were handle similar to the Color Sorting saving of images.

```
95      if k == 27:
96          break
97
98  ## out of "for" loop by removing tab indent
99  cv2.destroyAllWindows() ## to close windows python opened
100
101 Saved_Diff = cv2.imread('LastName-Difference.jpg')
102
103 Saved_Diff = cv2.line(Saved_Diff, (diff_x_location, 0), (diff_x_location, 480), color, th
104 Saved_Diff = cv2.line(Saved_Diff, (0, diff_y_location), (640, diff_y_location), color, th
105
106 gray_image_2 = cv2.line(gray_image_2, (diff_x_location, 0), (diff_x_location, 480), (0,0,
107 foreground = cv2.line(gray_image_2, (0, diff_y_location), (640, diff_y_location), (0,0,0)
```

Steps for Homogenous Coordinate Transformation Between Camera and Base

Homogenous Coordinate Transformation

Converting from Pixels to Centimeters

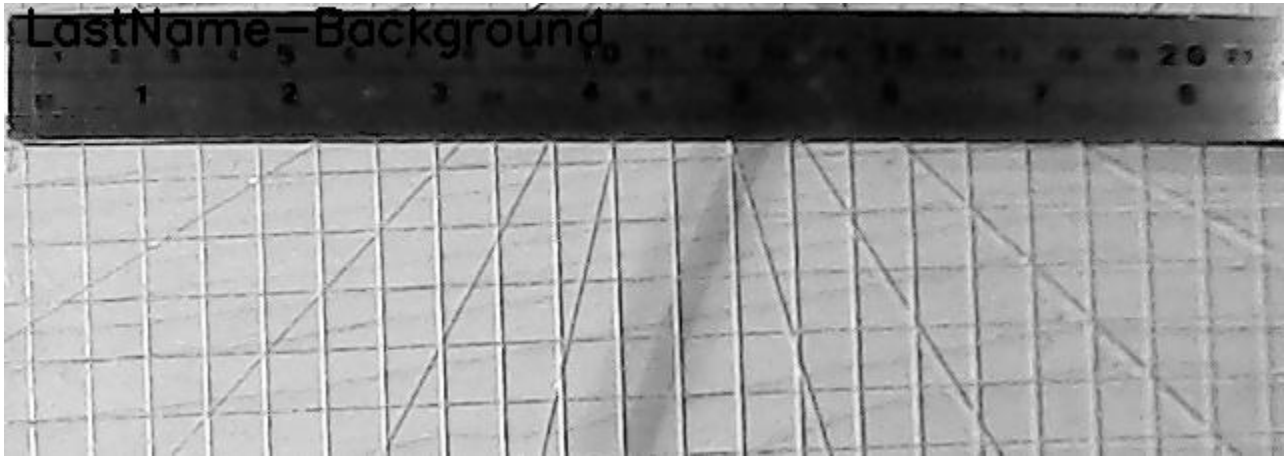
Till now, the ‘X” and ‘Y’ location has been in units of pixels. You will need these coordinates in units of centimeters to be able to complete the Homogenous Transformation.

Start by saving your Background subtraction module/code as something like Camera_Pixel_to_Centimeters.

Homogenous Coordinate Transformation

Converting from Pixels to Centimeters

Run the newly saved module/code. During the background stage section, measure across the viewable window. Aligning one end so the zero is just at the edge the ruler is parallel to the top as shown here. Note where the other edge goes out of view and get that measurement. **This time** I got a measurement of about 21.6 cm.



Homogenous Coordinate Transformation

Converting from Pixels to Centimeters

Now with the measure, you can add the following couple of lines of code to the module.

EGR_455_Lab08-Camer_Pixel_2_CM.py - C:/Users/kwnicho/ASU Dropbox/Kevin Nichols/Robotics/03 - Robotics - Labs/Lab Codes/Python Code

File Edit Format Run Options Window Help

```
1  import numpy as np
2  import cv2
3  import os
4
5  directory = r'C:\Users\kwnicho\Desktop\PythonImages'
6  os.chdir(directory)
7
8  cap = cv2.VideoCapture(1)
9
10 cm_to_pixel=21.6/640 #if measured across image
11
12 # While loop to capture background image
13 while (1):
14     _,frame = cap.read()
15
16     ## converting first image from RGB to gray scale
```

```
53     column_numbers = np.matrix(np.arange(640))
54     column_mult = np.multiply(column_sums,column_numbers)#element wise multiptation
55     column_total = np.sum(column_mult)
56     total_total = np.sum(np.sum(BW))# sum of all values in the image
57     BW_column_location = column_total/total_total
58     BW_column_location = np.nan_to_num(BW_column_location, nan=0)
59
60     X_Location = BW_column_location*cm_to_pixel
61
62     print('Object "X" location (cm): ',X_Location)
63
64     ## how to find the center of the object in 'Y' of the carmera image
65     row_sums = np.matrix(np.sum(BW,1))##'1' to sum the rows
66     row_sums = row_sums.transpose()
67     row_numbers = np.matrix(np.arange(480))
68     row_mult = np.multiply(row_sums,row_numbers)
69     row_total = np.sum(row_mult)
70     BW_row_location = row_total/total_total
71     BW_row_location = np.nan_to_num(BW_row_location, nan=0)
72
73     Y_Location = BW_row_location*cm_to_pixel
74
75     print('Object "Y" location (cm): ',Y_Location)
76
77     ########################################################################
78     # Draw the "cross hairs" across "foreground" image as a form of verification of locat
```

Homogenous Coordinate Transformation

Converting from Pixels to Centimeters

Additionally, you with add it so the measurements are also printed on the image by adding the following lines of code.

*EGR_455_Lab08-Camer_Pixel_2_CM.py - C:/Users/kwnicho/ASU Dropbox/Kevin Nichols/Robotics/03 - Robotics - Labs/Lab Codes/Python Codes/EGR_455_Lab08-Camer_

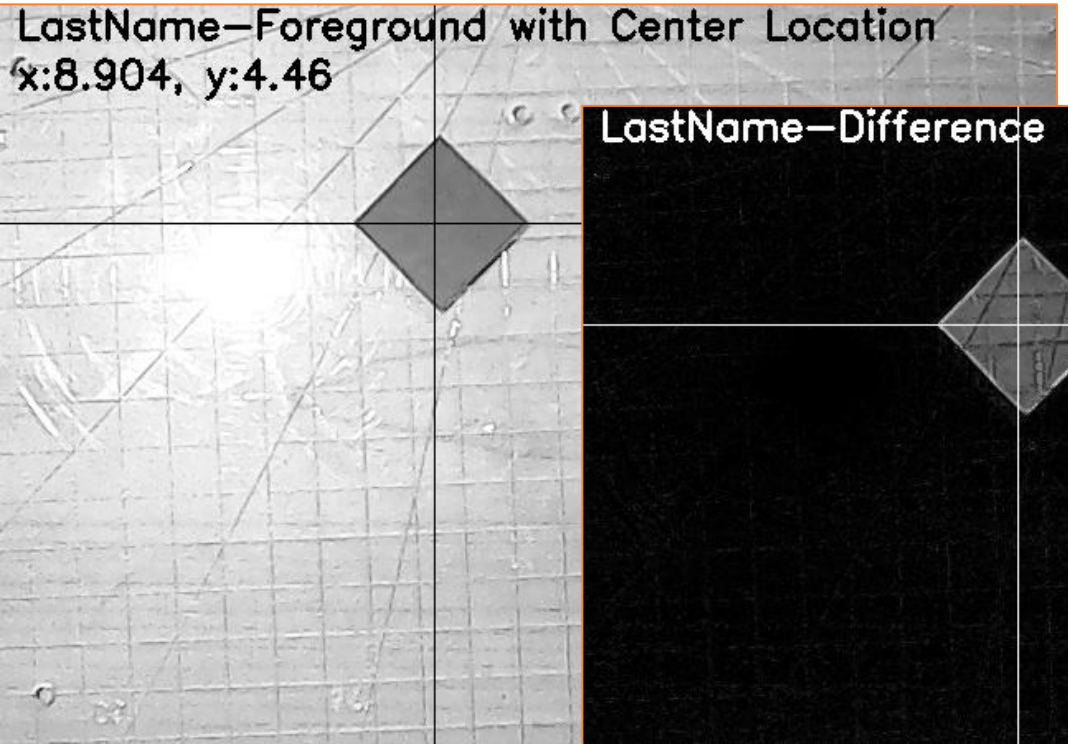
File Edit Format Run Options Window Help

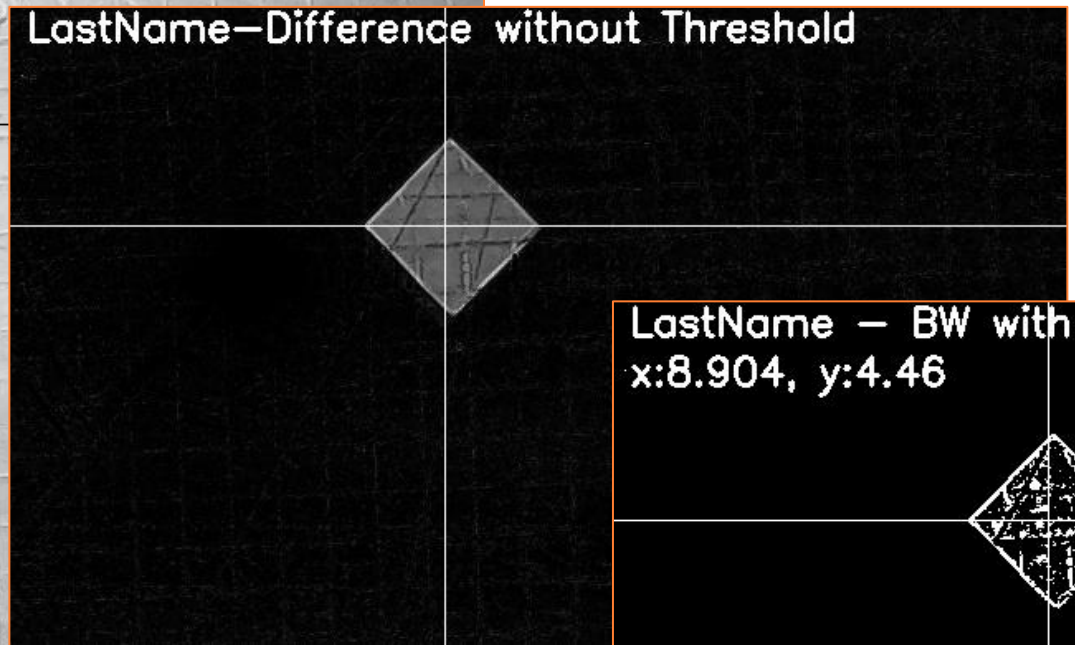
```
    ########################################################################
    # Draw the "cross hairs" across "foreground" image as a form of verification of location
    # Line color in BGR code
    color = (255,255,255)# white
    # Line thickness in pixels
    thickness = 1

    # formating the found locations so that it works with draw line
    diff_x_location = np.uint16(BW_column_location)
    diff_y_location = np.uint16(BW_row_location)

    # First line is the verical line and the second is the horizonital
    BW = cv2.line(BW, (diff_x_location, 0), (diff_x_location, 480), color, thickness)
    BW = cv2.line(BW, (0, diff_y_location), (640, diff_y_location), color, thickness)

    foreground = cv2.line(gray_image_2, (diff_x_location, 0), (diff_x_location, 480), (0,0,0), thickness)
    foreground = cv2.line(foreground, (0, diff_y_location), (640, diff_y_location), (0,0,0), thickness)


    unit_Text = f"x:{X_Location}, y:{Y_Location} "
    font = cv2.FONT_HERSHEY_SIMPLEX
    font_size = 0.8
    font_thickness = 2
    x,y = 10,20
    BW = cv2.putText(BW, unit_Text, (x,50), font, font_size, (255,255,255), font_thickness)

    cv2.imshow('Difference w/ Center & Threshold',BW)

    foreground = cv2.putText(foreground, unit_Text, (x,50), font, font_size, (0,0,0), font_thickness)

    cv2.imshow('Foreground w/ Center & Threshold',foreground)

    k=cv2.waitKey(5)
    if k == 27:
        break
```

Homogenous Coordinate Transformation

Converting from Pixels to Centimeters

With those changes, you should get the similar results as shown here.





Homogenous Coordinate Transformation

Converting from Pixels to Centimeters

Now, if you tried to check how accurate these converted measurements are, you with find it tricky once you place a rule into view.

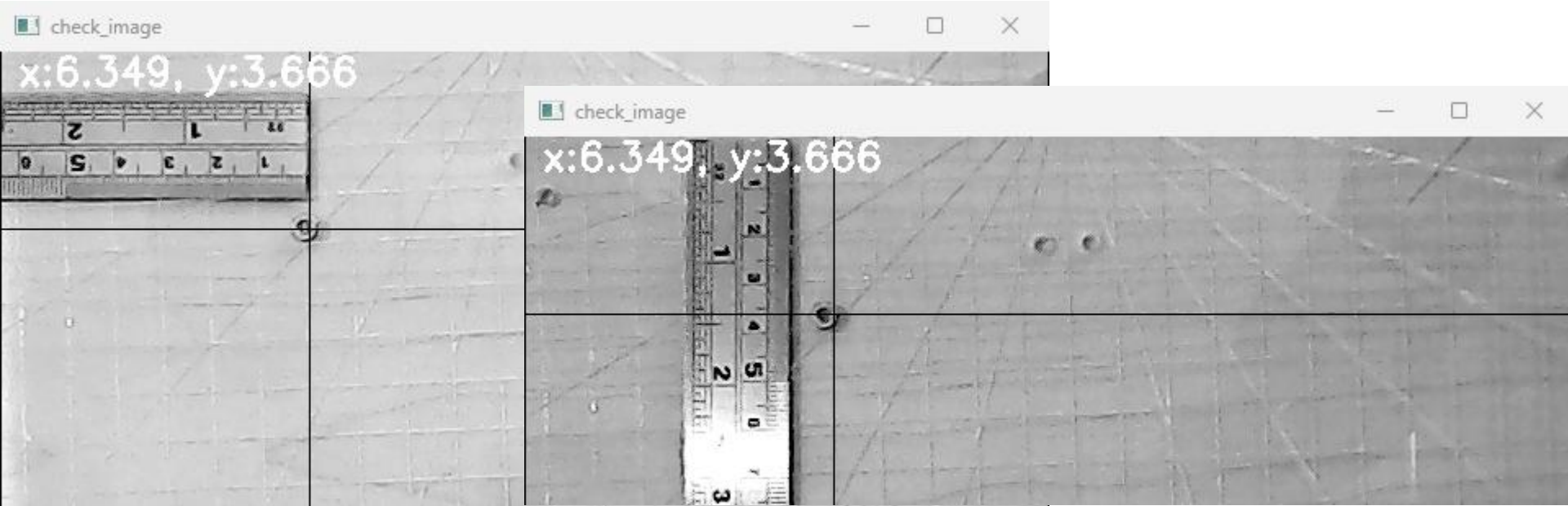
I found adding the following “while” loop helpful.

```
106
107     cv2.imshow('Difference w/ Center & Threshold',BW)
108
109     foreground = cv2.putText(foreground, unit_Text, (x,50), font, font_size, (0,0,0), font_thickness)
110
111     cv2.imshow('Foreground w/ Center & Threshold',foreground)
112
113     k=cv2.waitKey(5)
114     if k == 27:
115         break
116
117 # While loop to check object measured position
118 while (1):
119     _,frame = cap.read()
120
121     ## converting first image from RGB to gray scale
122     check_image = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
123
124     check_image = cv2.line(check_image, (diff_x_location, 0), (diff_x_location, 480), (0,0,0), thickness)
125     check_image = cv2.line(check_image, (0, diff_y_location), (640, diff_y_location), (0,0,0), thickness)
126     check_image = cv2.putText(check_image, unit_Text, (x,y), font, font_size, (0,0,0), font_thickness)
127
128     cv2.imshow('check_image',check_image)
129     k=cv2.waitKey(5)
130     if k == 27:
131         break
132
134 ## out of "for" loop by removing tab indent
135 cv2.destroyAllWindows() ## to close windows python opened
```

Homogenous Coordinate Transformation

Converting from Pixels to Centimeters

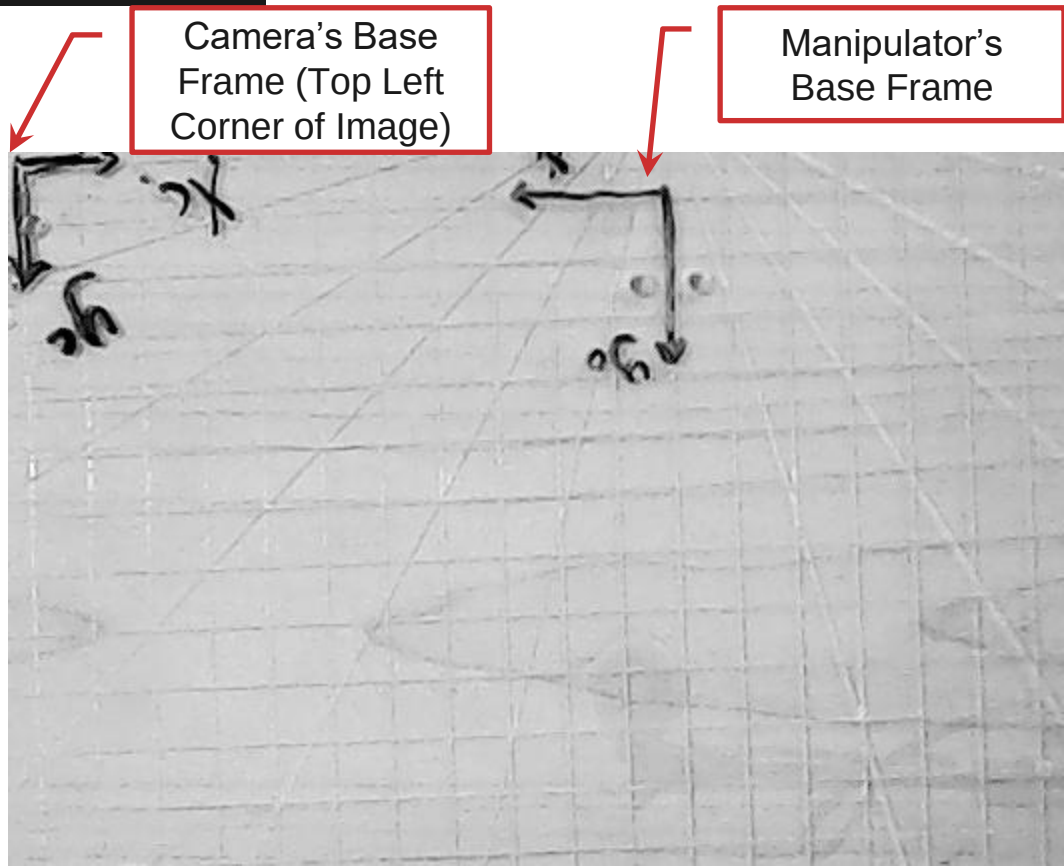
With that third “while” loop I can now measure without the measure moving. Additionally, if I bump the object, I still have the cross-hair lines as reference.



Homogenous Coordinate Transformation

Setting Up the Homogenous Transframation Matrix

Now you need to set-up a Homogenous Transformation Matrix that changes the camera's 'X' and 'Y' location to a location in reference to the manipulator's base coordinate frame.

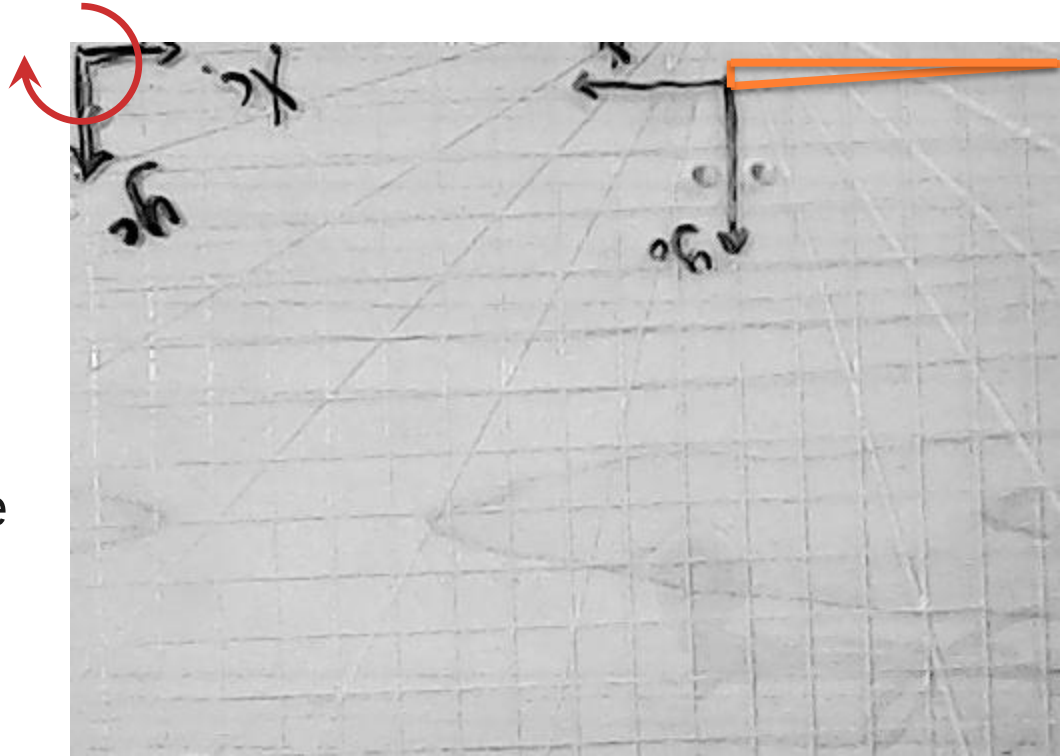


Homogenous Coordinate Transformation

Setting Up the Homogenous Transframation Matrix

One thing that is needed is the rotational matrix that describes the differences between the two coordinate frames.

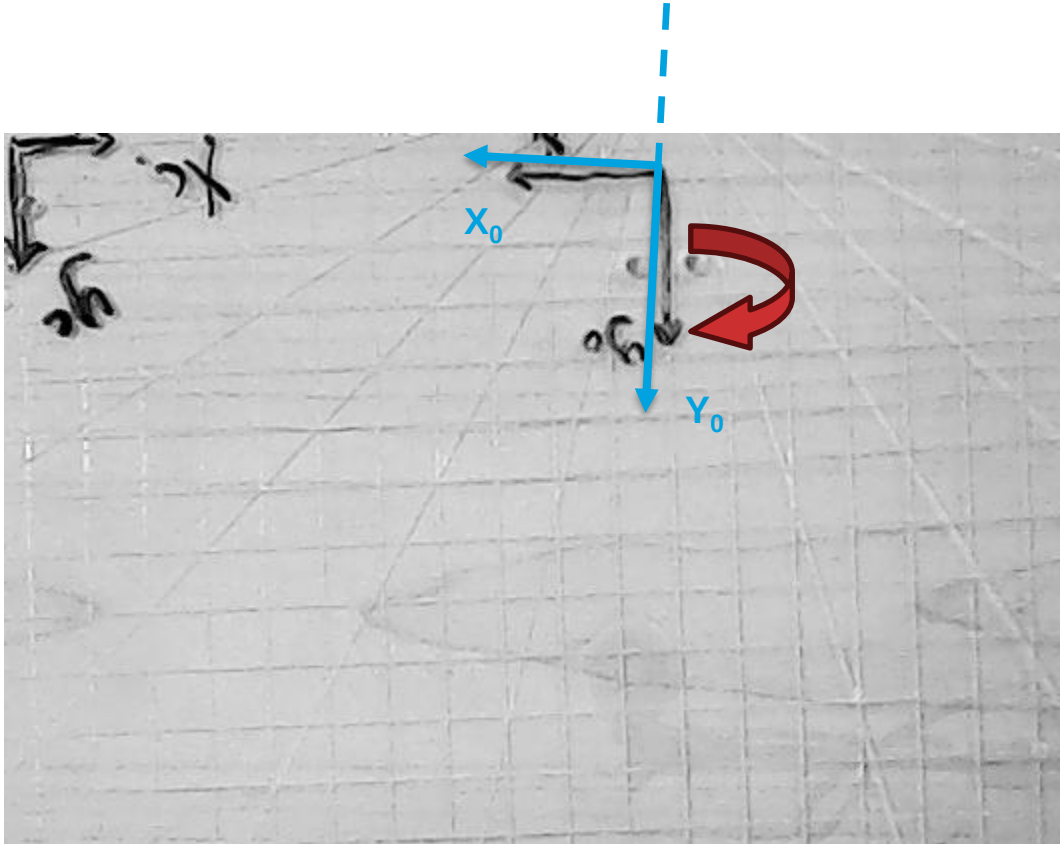
- It will take two rotation in this set-up to get the coordinate frame orientated the same.
- First rotation can be about the ‘Z’ axis 4° to get the ‘X_0’ axis aligned with Camera’s ‘X”



Homogenous Coordinate Transformation

Setting Up the `Matrix

- The second rotation can be about the ‘Y’ axis 180° to get the ‘X’ and ‘Z’ axis in the same direction.



Homogenous Coordinate Transformation

Setting Up the Homogenous Transframation Matrix

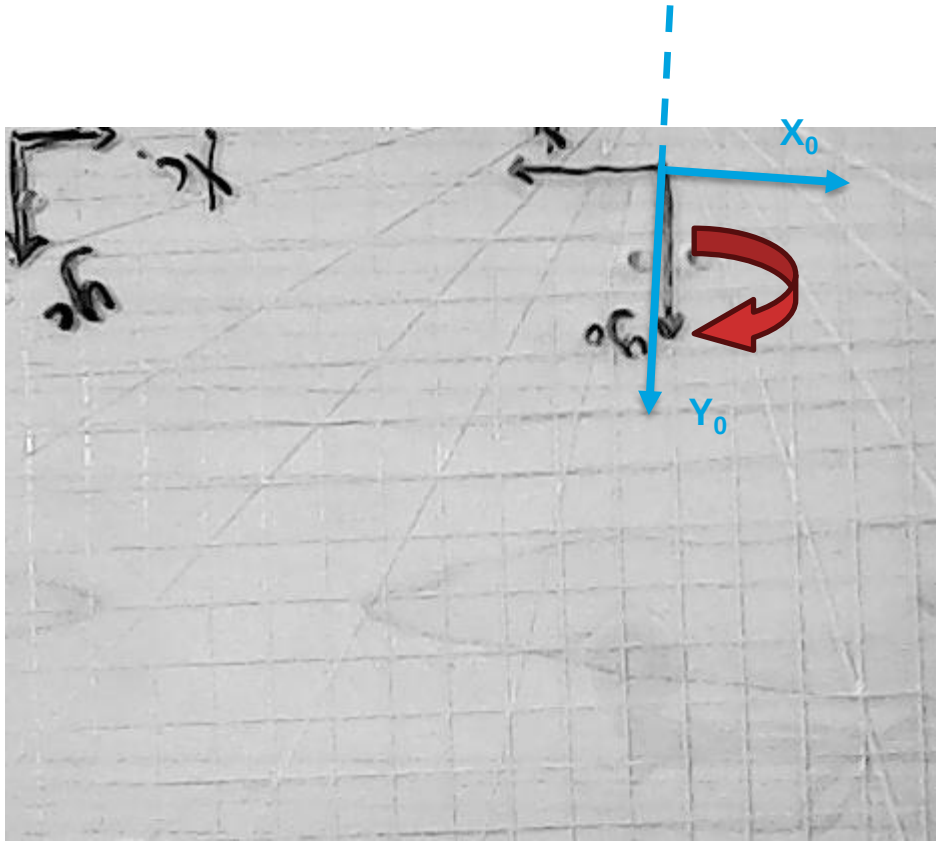
$$R_C^0 = R_z R_x$$

$$R_z = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\theta_z = 4°$$

$$R_y = \begin{bmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) \\ 0 & 1 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix}$$

$$\theta_y = 180°$$



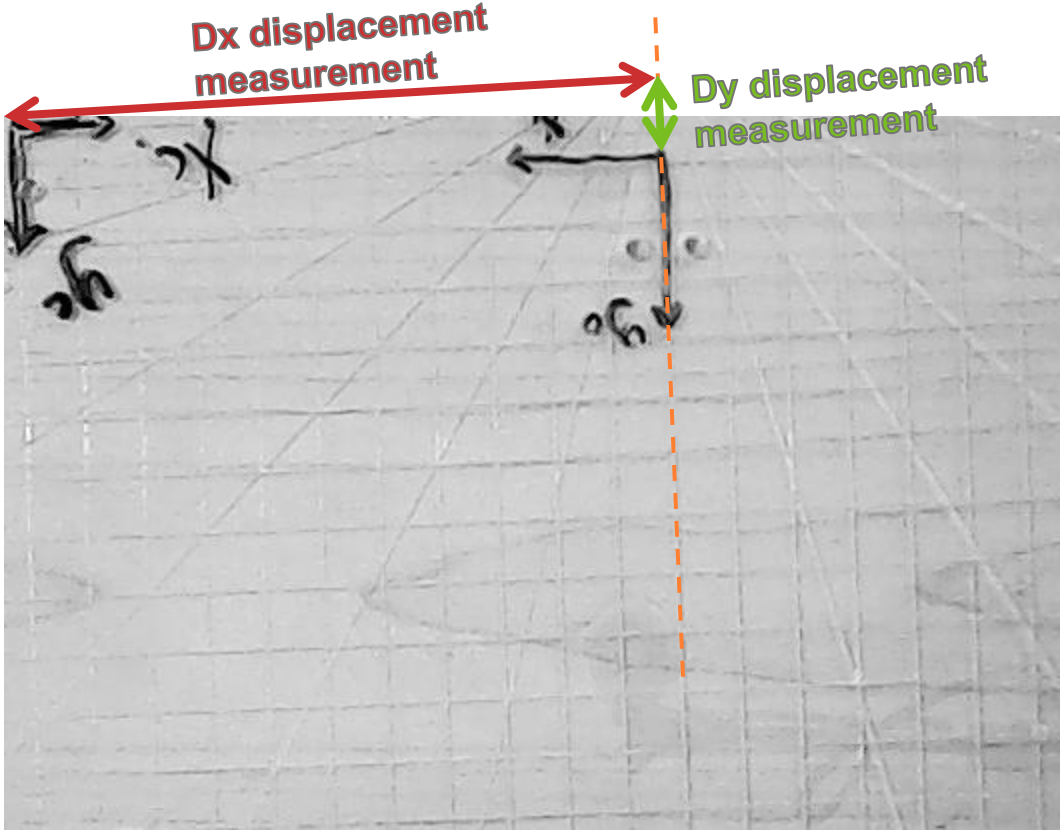
Homogenous Coordinate Transformation

Setting Up the Homogenous Transframation Matrix

The next thing that is needed is the displacement vector.

Dx = distance from the Manipulator's origin, along the X-axis to it interception of the Camera's Y-axis

Dy = distance from the Manipulator's origin, along the Y-axis to it interception of the Camera's X-axis



Homogenous Coordinate Transformation

Setting Up the Homogenous Transframation Matrix

Take the last module code where we did the pixel to centimeter conversion and save it to something like “Homogenous.”

Now we will add this information into the top of this code module.

EGR_455_Lab08-Homogenous.py - C:/Users/kwnicho/ASU Dropbox/Kevin Nichols/Robotics/03 - Robotics - Labs/Lab Codes/Python Codes/EGR_455_La

File Edit Format Run Options Window Help

```
import numpy as np
import cv2
import os

directory = r'C:\Users\kwnicho\Desktop\PythonImages'
os.chdir(directory)

cap = cv2.VideoCapture(1)

cm_to_pixel=21.6/640 #if measured across image

Theta_Z = -2 #in degrees, rotation about Z axis

Theta_Y = np.pi #or 180 degrees, rotation about Z axis
Theta_Z = (Theta_Z/180.0)*np.pi #Z rotation now in radians

R_z = [[np.cos(Theta_Z), -np.sin(Theta_Z), 0],[np.sin(Theta_Z), np.cos(Theta_Z), 0],[0, 0, 1]]
R_y = [[np.cos(Theta_Y), 0, np.sin(Theta_Y)],[0, 1, 0],[-np.sin(Theta_Y), 0, np.cos(Theta_Y)]]
R0_C = np.dot(R_z, R_y)

Dx = 7.4
Dy = -0.9
Dz = 0.0

D0_C=[[Dx],[Dy],[Dz]]
H0_C = np.concatenate((R0_C,D0_C),1)
H0_C = np.concatenate((H0_C, [[0, 0, 0, 1]]),0)

# While loop to capture background image
while (1):
    _,frame = cap.read()

    ## converting first image from RGB to gray scale
    gray_image_1 = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
```

Homogenous Coordinate Transformation

Setting Up the Homogenous Transframation Matrix

The next thing is to convert the camera dimensions into manipulator dimensions by adding this little bit of code in Red. I then modified the code in the green and blue to print the result.

Now, it is time to test.

Content changed

```
91      Y_Location = BW_row_location*cm_to_pixel
92
93      # Transform Coordinates
94      PC = [[X_Location],[Y_Location],[0],[1]]
95      P0 = np.dot(H0_C,PC)
96      X_BaseFrame = P0[0]
97      Y_BaseFrame = P0[1]
98
99      X_BaseFrame = np.round(X_BaseFrame,3)
100     Y_BaseFrame = np.round(Y_BaseFrame,3)
101     print('Object "X" location (cm): ',X_BaseFrame)
102     print('Object "Y" location (cm): ',Y_BaseFrame)
103
104     ########################################################################
105     # Draw the "cross hairs" across "foreground" image as a form of verification of
106     # Line color in BGR code
107     color = (255,255,255)# white
108     # Line thickness in pixels
109     thickness = 1
110
111     # formating the found locations so that it works with draw line
112     diff_x_location = np.uint16(BW_column_location)
113     diff_y_location = np.uint16(BW_row_location)
114
115     # First line is the verical line and the second is the horizonital
116     BW = cv2.line(BW, (diff_x_location, 0), (diff_x_location, 480), color, thickness
117     BW = cv2.line(BW, (0, diff_y_location), (640, diff_y_location), color, thickness
118
119     foreground = cv2.line(gray_image_2, (diff_x_location, 0), (diff_x_location, 480
120     foreground = cv2.line(foreground, (0, diff_y_location), (640, diff_y_location),
121
122
123     unit_Text = f"x:{X_BaseFrame}, y:{Y_BaseFrame}"
124     font = cv2.FONT_HERSHEY_SIMPLEX
125     font_size = 0.8
```

Move down and content changed

Homogenous Coordinate Transformation

Setting Up the Homogenous Transframation Matrix