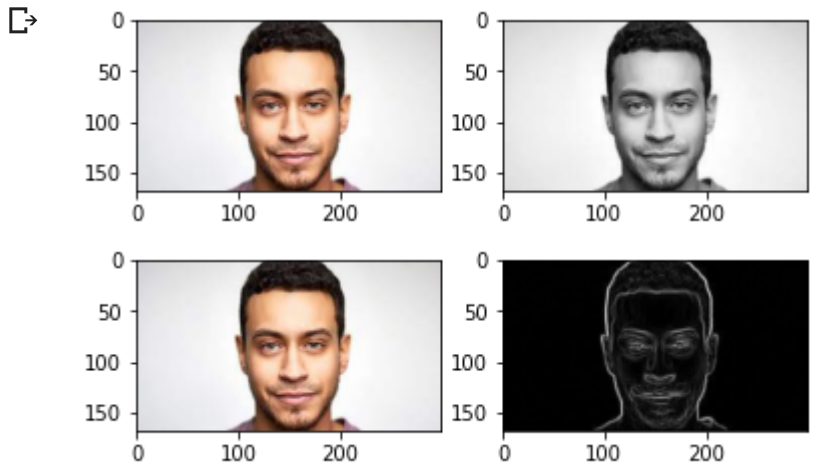
```
#oct 2022
from matplotlib.image import imread
import matplotlib.pyplot as plt
import numpy as np

#import image
image_name = "face.jpg"
image_input = imread(image_name)


#convert color into gray scale
grayscale_image = np.dot(image_input,[1,1,1])//3
grayscale_image = grayscale_image/255

#display orig and gray image
fig1 = plt.figure(1)
ax1, ax2 = fig1.add_subplot(121), fig1.add_subplot(122) #2 grids of 1 by 1
ax1.imshow(image_input) #show orig image
ax2.imshow(grayscale_image, cmap=plt.get_cmap('gray')) #show gray second
fig1.show()


#using sobel


#matrices associated with the Sobel filter
Gx = np.array([[1, 0, -1], [2, 0, -2], [1, 0, -1]])
Gy = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]])
"""
     _          _                  _          _
     |           |                 |            |
     | 1   0  -1 |                 |  1   2   1 |
Gx = | 2   0  -2 |    and     Gy = |  0   0   0 |
     | 1   0  -1 |                 | -1  -2  -1 |
     |_         _|                 |_          _|
"""

[rows,cols] = np.shape(grayscale_image) #gray image shape
sobel_image =  np.zeros(shape=(rows, cols))  # output image array (all elements are 0)


#traverses pixles - x and y direction #compute output
for i in range(rows - 2):
    for j in range(cols - 2):
        gx = np.sum(np.multiply(Gx, grayscale_image[i:i + 3, j:j + 3]))  # x direction
        gy = np.sum(np.multiply(Gy, grayscale_image[i:i + 3, j:j + 3]))  # y direction
        sobel_image[i + 1, j + 1] = np.sqrt(gx ** 2 + gy ** 2)  # calculate the "hypotenuse"



#display orig and sobel image
```

```
fig2 = plt.figure(2)
ax1, ax2 = fig2.add_subplot(121), fig2.add_subplot(122)
ax1.imshow(image_input)
ax2.imshow(sobel_image, cmap=plt.get_cmap('gray'))
fig2.show()
```



sources

https://coderspacket.com/sobel-edge-detection-from-scratch-using-python

https://github.com/adamiao/sobel-filter-tutorial/blob/master/sobel_from_scratch.py

```
#a3
#oct 2022
#final project.#v4
#assumption: numbers and hands can stay in the picture. for example the digit 0 is part of th
#images were split and filled in by hand - assumes images were properly filled in

import cv2 as cv
import numpy as np
from google.colab.patches import cv2_imshow #since google collab cant use cv2.imshow
from PIL import Image


#min bounding box
def boundBox(image):

    threshold = 100     #determines cutoff for valid black pixel, decrease to tighten boundin
    img = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
    height = img.shape[0]
    width = img.shape[1]

    top_row = height - 1    #initialize as max height to insure the value gets change
    bottom_row = 0  #initialize as min height " " " "
    left_col = width - 1     #initialize as max width " " " "
    right_col = 0   #initialize as min width " " " "
```

```
        for x in range(height):
            for y in range(width):

                if(img[x][y] < threshold):  #if the pixel is black
                    if(x < top_row):
                        top_row = x     #this will be the top of the image
                    if(x > bottom_row):
                        bottom_row = x  #bottom of image
                    if(y < left_col):
                        left_col = y    #left side of image
                    if(y > right_col):
                        right_col = y   #right side of image

        new_height = bottom_row - top_row
        new_width = right_col - left_col
        new_img = np.ones((new_height, new_width, 3), np.uint8)
        new_img = cv.cvtColor(new_img, cv.COLOR_BGR2GRAY)

        if(new_height > 0 and new_width > 0):
            for x in range(new_height):
                for y in range(new_width): #go through the new image (bounding box image
                    pixel = img[x+top_row][y+left_col] #get the corresponding pixel value in og i
                    new_img[x][y] = pixel #set to new image

        return new_img


    # Get the pixel from the given image #get colours
    def get_pixel(image, i, j):
        # Inside image bounds?
        width, height = image.size
        if i > width or j > height:
          return None

        # Get Pixel
        pixel = image.getpixel((i, j))
        return pixel


    # count num of black pixles
    def counter(img):
      count_black= 0
      for i in range (img.size[0]):
        for j in range (img.size[1]):
          if get_pixel(img, i,j) == 0:#(0,0,0): #if its black pixle
            count_black +=1

      return count_black
```

```
#main
x,y,z=[],[],[]
x_percent, y_percent, z_percent = [],[],[]

imageNames = ['zero.jpg','one.jpg','two.jpg', 'three.jpg','four.jpg','five.jpg','six.jpg','se

for imageName in imageNames:

  #read image
  img = cv.imread(imageName) #orig image as an array

  #display image
  print("orig pic",imageName)
  cv2_imshow(img)

  #find min bounding box
  bound = boundBox(img)
  img = bound
  img_orig = Image.fromarray(img)

  #1-   The aspect ratio. (This is the first feature x). The aspect ratio of an image is the
  height = img.shape[0]
  width = img.shape[1]
  print("calculate aspect ratio\n",width,":",height,"\n")

  s = width,":",height
  x.append( s)
  x_percent.append( round(width/height,3))
  #x.append( width,":",height) #x[i] = width,":",height


  #2-   The ratio of the left half black pixels area to the rectangle that has the image. (Th
  #3-   The ratio of the right half black pixels area to the rectangle that has the image. (T

  # Cut the image into left and right half
  width_cutoff = width // 2
  s1 = img[:, :width_cutoff]
  s2 = img[:, width_cutoff:]

  print("left pic")
  cv2_imshow(s1)
  print("right pic")
  cv2_imshow(s2)

  img_from_arr_left  = Image.fromarray(s1)
  img_from_arr_right  = Image.fromarray(s2)

  #count black pixels
  count_black_orig = counter(img_orig)#(img)
  #print("number of black pixles in orig pic", count_black_orig)
```

```
    count_black_left =  counter(img_from_arr_left)#(s1)
    print("number of left black pixles: all black pixles\n", count_black_left,":", count_black_
    s = count_black_left,":", count_black_orig
    y.append( s)
    y_percent.append( round(count_black_left/count_black_orig,3))

    count_black_right =   counter(img_from_arr_right)
    print("number of right black pixles: all black pixles\n", count_black_right,":", count_blac
    s = count_black_right,":", count_black_orig
    z.append( s)
    z_percent.append( round(count_black_right/count_black_orig,3))

print("\n X Y Z \n")
print(x)
print(y)
print(z)

print("\n X Y Z as percents \n")
print(x_percent)
print(y_percent)
print(z_percent)
```

```
orig pic zero.jpg
```

```
calculate aspect ratio
 35 : 67

left pic
```

```
right pic
```

```
number of left black pixles: all black pixles
 301 : 462
number of right black pixles: all black pixles
 161 : 462
orig pic one.jpg
```

```
calculate aspect ratio
 34 : 68

left pic
```

```
right pic
```

```
number of left black pixles: all black pixles
 316 : 707
number of right black pixles: all black pixles
 391 : 707
orig pic two.jpg
```

```
calculate aspect ratio
 31 : 68

left pic
```

right pic

number of left black pixles: all black pixles
 289 : 649
number of right black pixles: all black pixles
 360 : 649
orig pic three.jpg

calculate aspect ratio
 46 : 68

left pic

right pic

number of left black pixles: all black pixles
 619 : 909
number of right black pixles: all black pixles
 290 : 909
orig pic four.jpg

calculate aspect ratio
 50 : 68

left pic

right pic

number of left black pixles: all black pixles
 272 : 808
number of right black pixles: all black pixles

```
 536 : 808
orig pic five.jpg
```

```
calculate aspect ratio
 62 : 69

left pic
```

```
right pic
```

```
number of left black pixles: all black pixles
 504 : 928
number of right black pixles: all black pixles
 424 : 928
orig pic six.jpg
```

```
calculate aspect ratio
 33 : 69

left pic
```

```
right pic
```

```
number of left black pixles: all black pixles
 238 : 769
number of right black pixles: all black pixles
 531 : 769
orig pic seven.jpg
```

```
calculate aspect ratio
 37 : 69

left pic
```

```
right pic
```

```
number of left black pixles: all black pixles
 152 : 749
number of right black pixles: all black pixles
 597 : 749
orig pic eight.jpg
```

```
calculate aspect ratio
 46 : 77

left pic
```

```
right pic
```

```
number of left black pixles: all black pixles
 300 : 872
number of right black pixles: all black pixles
 572 : 872
orig pic nine.jpg
```

a3 sources

https://gist.github.com/Integralist/4ca9ff94ea82b0e407f540540f1d8c6c

http://2017.compciv.org/guide/topics/python-nonstandard-libraries/pillow.html

https://stackoverflow.com/questions/72623020/calculate-the-pixel-ratio-of-two-halves-of-an-image

https://stackoverflow.com/questions/45384968/how-to-cut-an-image-vertically-into-two-equal-sized-images

https://www.codementor.io/@isaib.cicourel/image-manipulation-in-python-du1089j1u

https://thispointer.com/convert-a-numpy-array-to-an-image-in-python/#:~:text=The%20Approach%20to%20convert%20NumPy,using%20the%20save()%20method

.

```
[(35, ':', 67), (34, ':', 68), (31, ':', 68), (46, ':', 68), (50, ':', 68), (62, ':', 69
```

```
import cv2 as cv
import numpy as np

def boundBox(image):

    threshold = 100     #determines cutoff for valid black pixel, decrease to tighten boundin
    img = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
    height = img.shape[0]
    width = img.shape[1]

    top_row = height - 1    #initialize as max height to insure the value gets change
    bottom_row = 0  #initialize as min height " " " "
    left_col = width - 1     #initialize as max width " " " "
    right_col = 0   #initialize as min width " " " "

    for x in range(height):
        for y in range(width):

            if(img[x][y] < threshold):  #if the pixel is black
                if(x < top_row):
                    top_row = x     #this will be the top of the image
                if(x > bottom_row):
                    bottom_row = x  #bottom of image
                if(y < left_col):
                    left_col = y    #left side of image
                if(y > right_col):
                    right_col = y   #right side of image

    new_height = bottom_row - top_row
    new_width = right_col - left_col
    new_img = np.ones((new_height, new_width, 3), np.uint8)
    new_img = cv.cvtColor(new_img, cv.COLOR_BGR2GRAY)

    if(new_height > 0 and new_width > 0):
        for x in range(new_height):
            for y in range(new_width): #go through the new image (bounding box image
                pixel = img[x+top_row][y+left_col] #get the corresponding pixel value in og i
                new_img[x][y] = pixel #set to new image

    return new_img

from google.colab.patches import cv2_imshow #since google collab cant use cv2.imshow
#from bounding_box import *
```

```
import numpy as np
import cv2 as cv

zero = cv.imread("zero.jpg")
#print(type(zero))
#zero = cv2_imread("zero.jpg")
zero_bound = boundBox(zero)
#cv.imshow("Zero", zero_bound)
cv2_imshow( zero_bound)
cv.waitKey(0)
```

Colab paid products - Cancel contracts here

✓ 1s completed at 11:51 PM