VNU HCMC-UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

PROJECT REPORT

APPLIED MATHEMATICS AND STATISTICS

IMAGE PROCESSING

Lecturers:	**Vu Quoc Hoang**
Practical Instructor:	**Phan Thi Phuong Uyen**
	Nguyen Van Quang Huy
	Le Thanh Tung
Student:	**Bui Quang Thanh - 20127329**

Table of Contents

FUNCTION LIST

Function	Complete
Adjust the brightness	100%
Adjust the contrast	100%
Flip image (Horizontal - Vertical)	100%
Convert RGB image to grayscale image	100%
Stack 2 photos of the same size	100%
Blur image	100%
Crop the picture with a circle frame	100%
Crop the image with the frame as 2 diagonal ellipses	100%

ABTRACT

Nowadays, image processing became very important especially in real-time where the results of real-time image processing failures can be severe; therefore, the study and research in methods of real-time image processing are of extreme significance. The main contribution of this paper is to provide an overview of the current state of real-time image processing research (Applications), the relevant techniques, and methods.

INTRODUCTION

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too.

We have two Types of methods used for image processing: (1) Analog Image Processing: Here we process analog signals that have two-dimensional representation only, where the images are modified by electrical signals, e.g., television images. (2) Digital Image Processing: Here we represent the image by a matrix of pixels where the image contains a set of elements that need to be processed, we can do that by many libraries and algorithms.

In this project, it requires more image processing based on the knowledge learned and done through project 1.

Perform some basic image processing functions as follows:

1. Change the brightness of the photo.
2. Change the contrast.
3. Convert RGB image to grayscale image.
4. Flip the image (horizontal - vertical).
5. Stack 2 photos of the same size.
6. Blur the image.

7 & 8 Crop photo by frame

FUNCTION

1. Support Function

def openAndReadImage(nameImg):

Input: nameImg (name of image which entered by user)

Output: image (numpy.ndarray)

Purpose: read image and return image (numpy.ndarray)

def iamgePreProcess(image): pre-processing of the image

Input: image: numpy.ndarray (is the matrix of the image)

Output:

- flat_image: the matrix of the image after reshape (numpy.ndarray)
- width: the width of image (int)
- height: the height of image (int)
- n_channels: 3 (int)

def check(pixel): handle pixel when pixel > 255 or pixel < 0

Input: pixel (float)

Output: pixel (float)

Purpose: Function will handle if pixel > 255 or pixel < 0

def output(result,width,height,n_channel,name,input,save=False):

Input:

- result: (numpy.ndarray) matrix after processing
- width: (int) the width of image
- height: (int) the height of image
- n_channel: (int) a red, green, and blue channel
- name: (string) name of image
- input: (string) string which entered by user
- save: (boolean) to confirm “Do you save the image or not?”

Output: display image to screen if save=True users can save image

2. Adjust the brightness

Implementation idea

def adjustBrightness(image, brightness): adjust the brightness of image

Input:

- image: numpy.ndarray (is the matrix of the image)
- brightness:int (number entered by user)

Output:

- result: numpy.ndarray (the matrix of the image after add brightness)

Make a reshape of the image: image.reshape(width* height,channels).

Iterate through each pixel r to get 3 colors corresponding to R G B. To increase the brightness of the image, add the 3 colors that have just been taken in turn for brightness. (brightness can be positive or negative). Each addition checks the validity of the color using the check(pixel) function.

Here to adjust the brightness for the image, we will fill in the adjustment brightness in the range from [-255,255] depending on the level to be adjusted

3.Adjust the contract

Implementation idea

def adjustContract(image,contrast):

Input:

- image: numpy.ndarray (is the matrix of the image)
- contract:int (number entered by user)

Output:

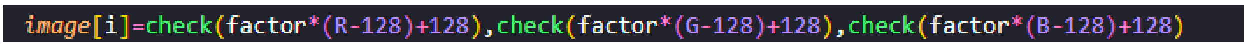
- result: numpy.ndarray (the matrix of the image after changing the constract)

Make a reshape of the image: image.reshape(width* height,channels).

Iterate through each pixel r to get 3 colors corresponding to R G B. Calculate a contrast correction factor which is given by the following formula:

$$F = \frac{259(C + 255)}{255(259 - C)}$$

With C is contract which entered by user. The next step is to perform the actual contrast adjustment itself.



```
image[i]=check(factor*(R-128)+128),check(factor*(G-128)+128),check(factor*(B-128)+128)
```

3. Flip image (Horizontal - Vertical)

Implementation idea

def flip_Image(image,input):

Input:

- image: numpy.ndarray (is the matrix of the image)
- input: string (If user want flip horizontal or vertical)

Output:

- result: numpy.ndarray (is the matrix of the image after flip using numpy)

Use the support library to be able to flip photos.

4. Convert RGB image to grayscale image

Implementation idea

def changeGrayScale(image):

Input:

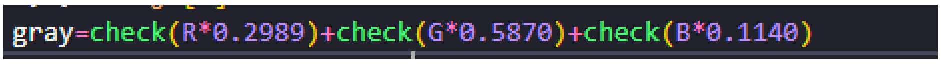
- image: numpy.ndarray (is the matrix of the image)

Output:

- result: numpy.ndarray (the matrix of the image after grayscale)

Make a reshape of the image: image.reshape(width* height,channels).

Iterate through each pixel r to get 3 colors corresponding to R G B.Calculate gray color using formula:
New grayscale image = ((0.2989 * R) + (0.5870 * G) + (0.1140 * B)).



```
gray=check(R*0.2989)+check(G*0.5870)+check(B*0.1140)
```

Using function check(pixel) to check pixel when it > 255 or it <0.
Applying this equation to the image → image[i]=gray,gray,gray.

5. Stack 2 photos of the same size

Implementation idea

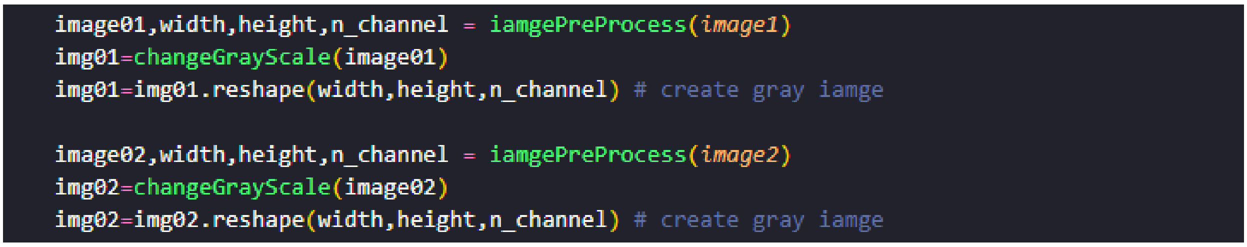
def plus2Image(image1,image2,alpha) :
Input:

- image1: numpy.ndarray (the matrix of the image01)
- image2: numpy.ndarray (the matrix of the image02)
- alpha : float (the number that entered by user)

Output:

- result: numpy.ndarray (the matrix of the image after plus 2 image)

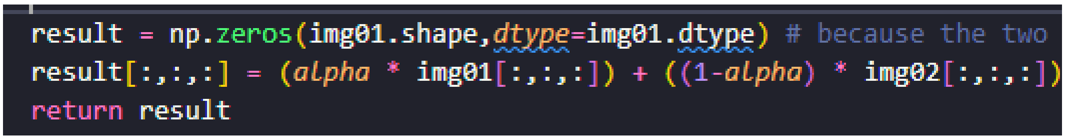
The first step is to convert to grayscale based on the grayscale conversion function installed above.(applied image01 and image02).



```
    image01,width,height,n_channel = iamgePreProcess(image1)
    img01=changeGrayScale(image01)
    img01=img01.reshape(width,height,n_channel) # create gray iamge

    image02,width,height,n_channel = iamgePreProcess(image2)
    img02=changeGrayScale(image02)
    img02=img02.reshape(width,height,n_channel) # create gray iamge
```

First create an image the same size as 1 of the 2 photos. Based on the alpha value just entered, add 2 matrices with the corresponding alpha value. An image with a higher alpha value will be seen more clearly.



```
  result = np.zeros(img01.shape,dtype=img01.dtype) # because the two
  result[:,:,:] = (alpha * img01[:,:,:]) + ((1-alpha) * img02[:,:,:])
  return result
```

6. Blur image

Implementation idea

def box_blur (img,height,width) :
Input:

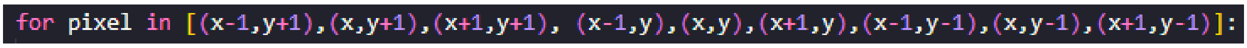
- img: numpy.ndarray (the matrix of the image)
- height: int (the height of the image)
- width: int (the width of the image)

Output:

- result: numpy.ndarray (the matrix of the image after bluring image)

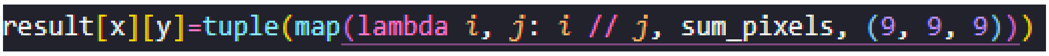
First create a resulting image by copying from the input image → result = img.copy().

In this function, there are 2 loops. Using two loops traversing the entire image, initialize a tuple sum_pixels to sum the color values of the old image by traversing the surrounding 9 elements in turn



```
for pixel in [(x-1,y+1),(x,y+1),(x+1,y+1), (x-1,y),(x,y),(x+1,y),(x-1,y-1),(x,y-1),(x+1,y-1)]:
```

then at each x,y coordinates of the new image assigned the new colors are calculated by dividing the sum_pixels by 9 .



```
result[x][y]=tuple(map(lambda i, j: i // j, sum_pixels, (9, 9, 9)))
```

7.Crop the picture with a circle frame

Implementation idea

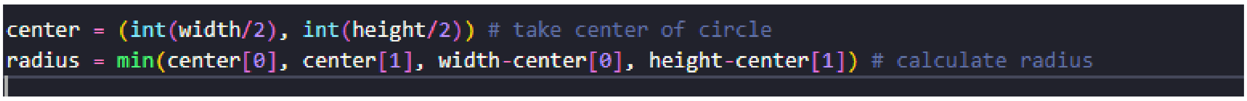
The idea is to create a circular mask, then apply that mask to the original image to create a new image based on the drawn mask.

def crop_Circle (name,save) :
Input:

- name: string (name of the image)
- save: string (to confirm save image)

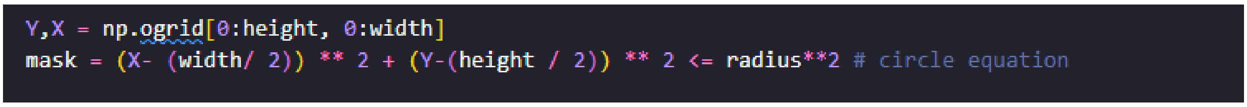
Output: display or save image after cropping

Calculate the values needed to create a circle:



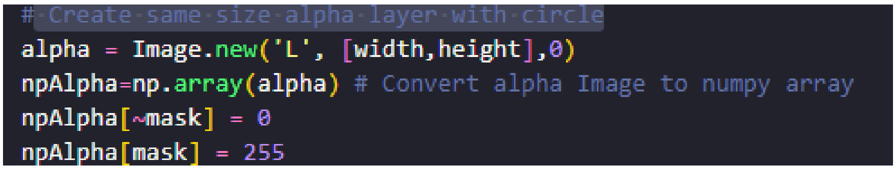
```
center = (int(width/2), int(height/2)) # take center of circle
radius = min(center[0], center[1], width-center[0], height-center[1]) # calculate radius
```

Use the numpy library (ogrid) to get the array values from 0 → height , 0 → width → Y , X
Proceed to create a circle through the equation of the circle:

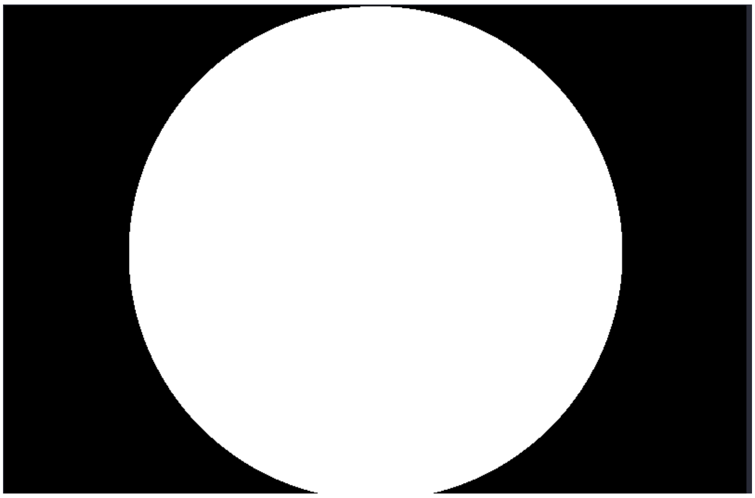


```
Y,X = np.ogrid[0:height, 0:width]
mask = (X- (width/ 2)) ** 2 + (Y-(height / 2)) ** 2 <= radius**2 # circle equation
```

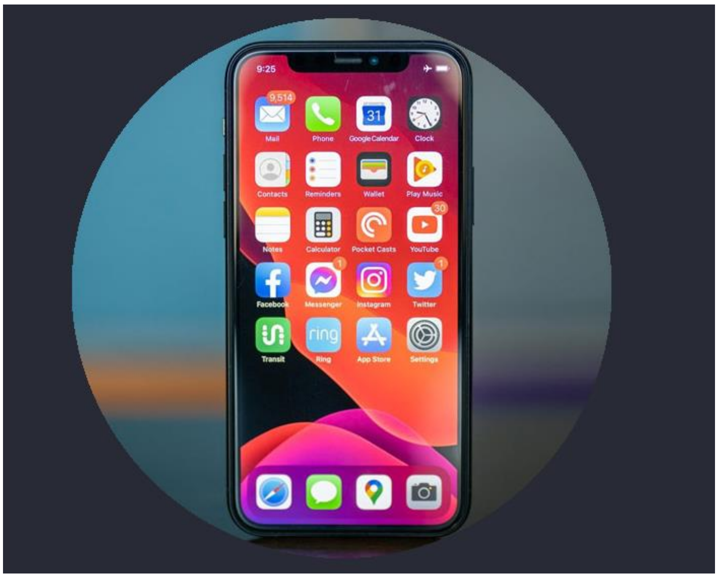
Create same size alpha layer with circle



```
# Create same size alpha layer with circle
alpha = Image.new('L', [width,height],0)
npAlpha=np.array(alpha) # Convert alpha Image to numpy array
npAlpha[~mask] = 0
npAlpha[mask] = 255
```



Add alpha layer to RGB using np.dstack(img,npAlpha)



8. Crop the image with the frame as 2 diagonal ellipses

Implementation idea

The idea is to create a circular mask, then apply that mask to the original image to create a new image based on the drawn mask.

def crop_2_Ellipses_Cross (name,save) :

Input:

- name: string (name of the image)
- save: string (to confirm save image)

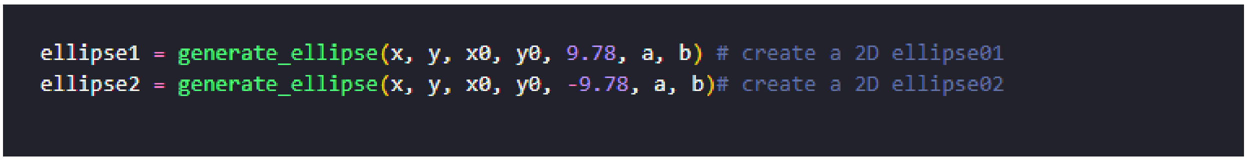
Output: display or save image after cropping

Iinitialize coordinates of x,y based on size of image:

x = np.linspace(-1.2, 1.2, width)

y = np.linspace(-1.2, 1.2, height)[:,None]

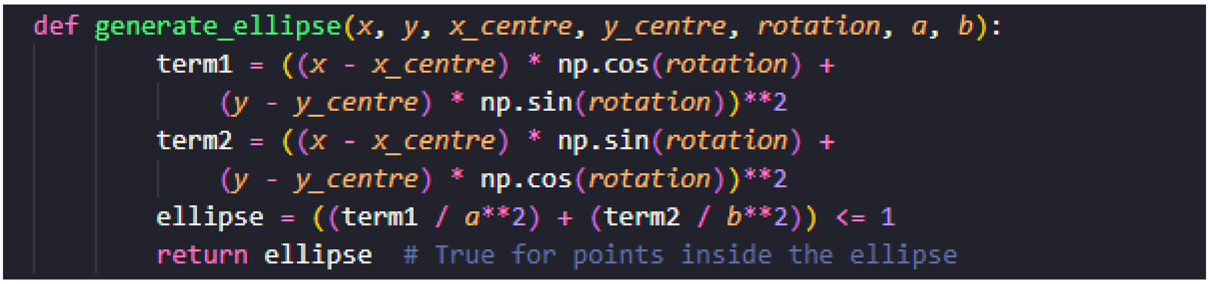
Create two masks that are two ellipses with 9.79 is rotation



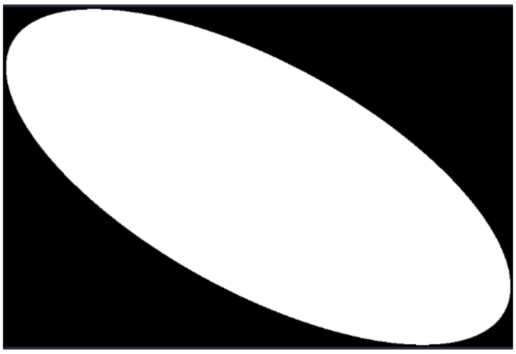
```
ellipse1 = generate_ellipse(x, y, x0, y0, 9.78, a, b) # create a 2D ellipse01
ellipse2 = generate_ellipse(x, y, x0, y0, -9.78, a, b)# create a 2D ellipse02
```

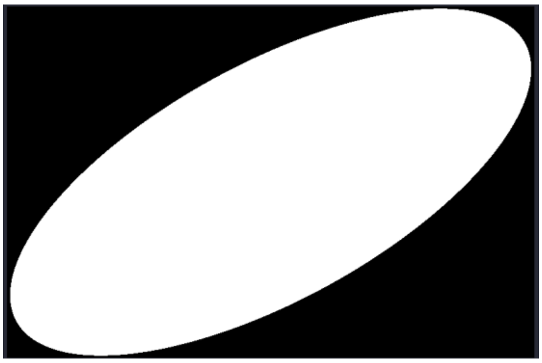
To create an oblique ellipse, you need to follow the formula:

$$\frac{((x-h)\cos(A)+(y-k)\sin(A))^2}{a^2}+\frac{((x-h)\sin(A)-(y-k)\cos(A))^2}{b^2}=1,$$

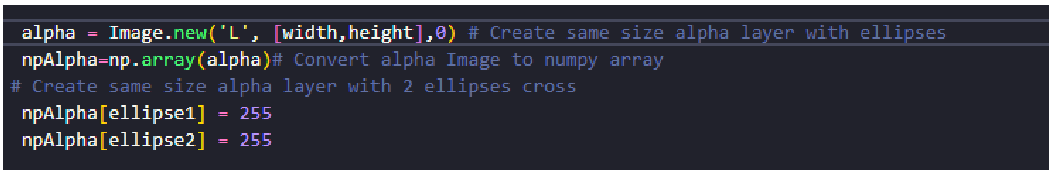


```
def generate_ellipse(x, y, x_centre, y_centre, rotation, a, b):
        term1 = ((x - x_centre) * np.cos(rotation) +
            (y - y_centre) * np.sin(rotation))**2
        term2 = ((x - x_centre) * np.sin(rotation) +
            (y - y_centre) * np.cos(rotation))**2
        ellipse = ((term1 / a**2) + (term2 / b**2)) <= 1
        return ellipse  # True for points inside the ellipse
```

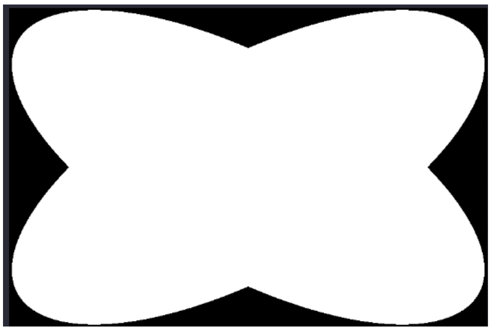




Create same size alpha layer with 2 ellipses cross



```
 alpha = Image.new('L', [width,height],0) # Create same size alpha layer with ellipses
 npAlpha=np.array(alpha)# Convert alpha Image to numpy array
# Create same size alpha layer with 2 ellipses cross
 npAlpha[ellipse1] = 255
 npAlpha[ellipse2] = 255
```

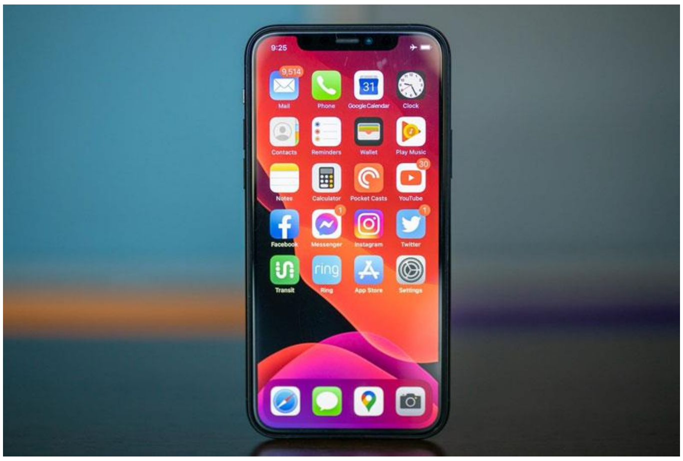


Add alpha layer to RGB using np.dstack(img,npAlpha)

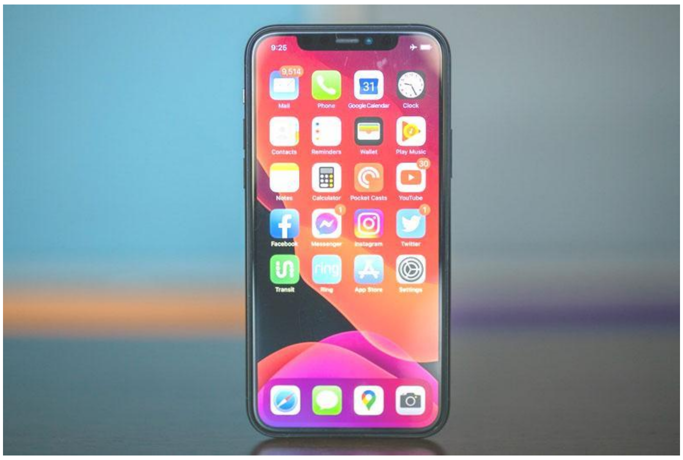
Because JPG does not support transparency - RGBA means Red, Green, Blue, Alpha - Alpha is transparency so in function number 7 and number 8 must use PNG format image to support transparency.

TESTCASE

1.Adjust the brightness

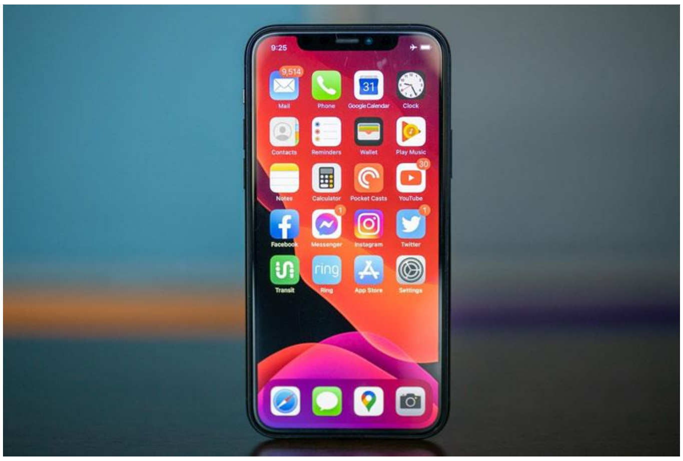


Original

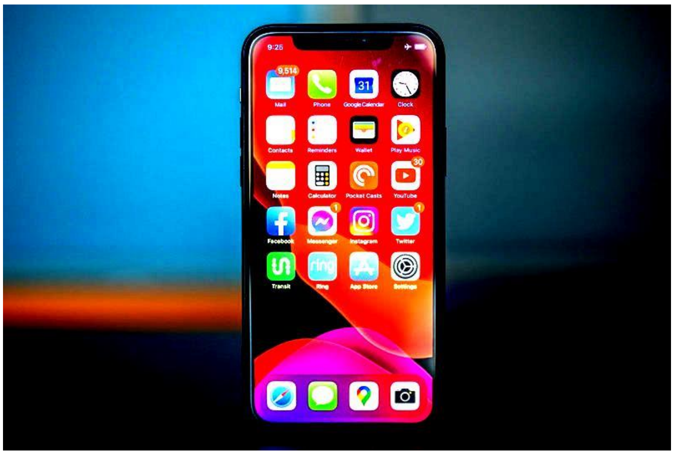


Brightness = 50

2.Adjust the contrast

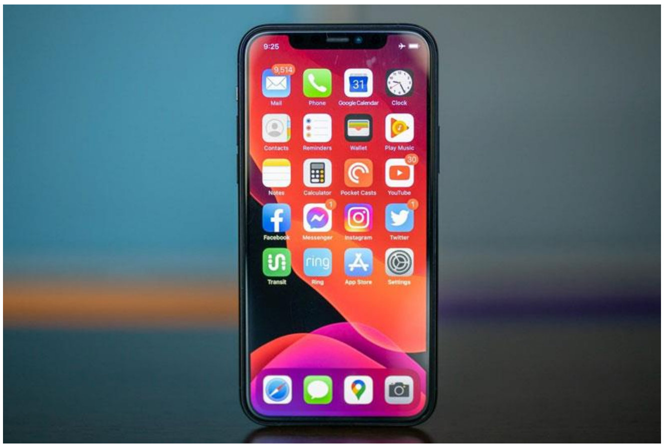


Original

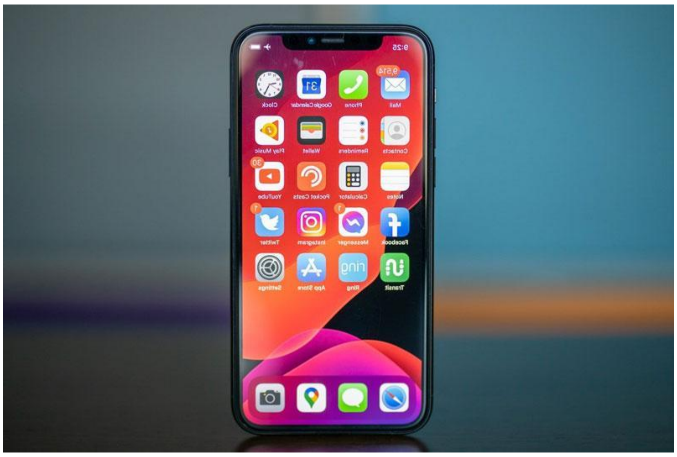


Contrast = 128

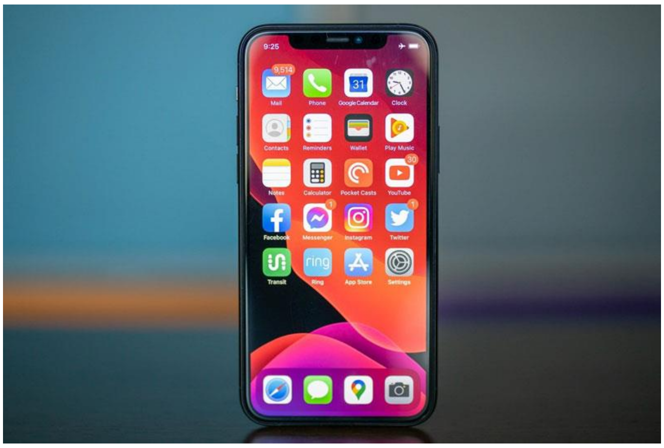
3. Flip image (Horizontal - Vertical)



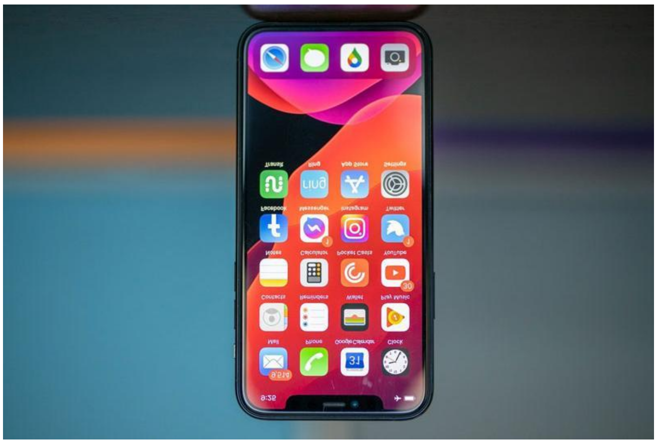
Original



Horizontal

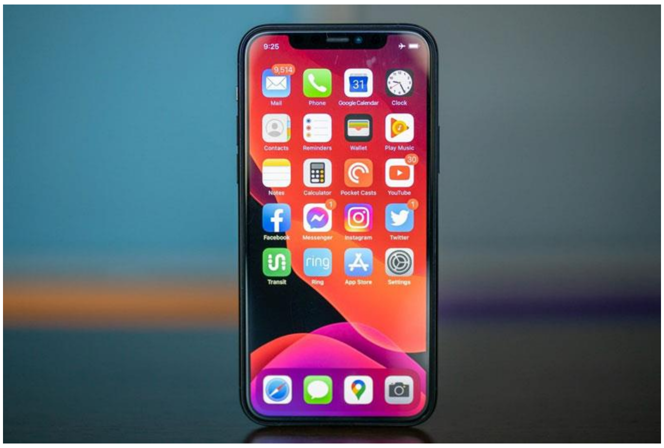


Original

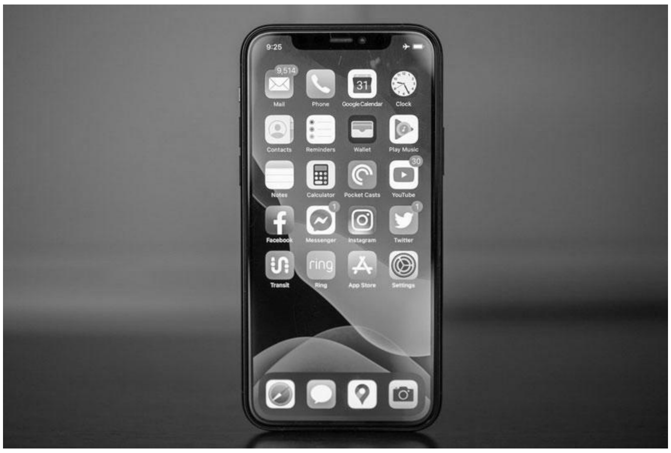


Vertical

4.Convert RGB image to grayscale image

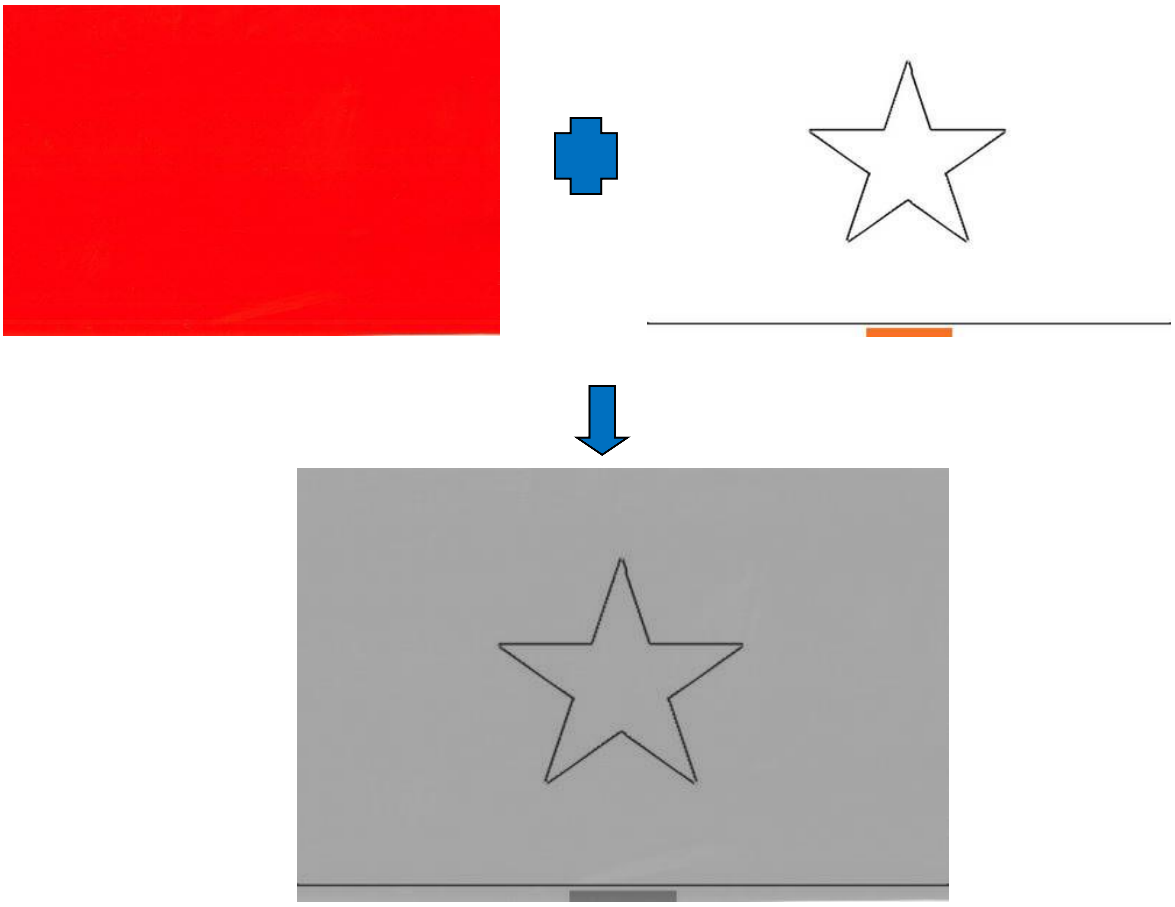


Original



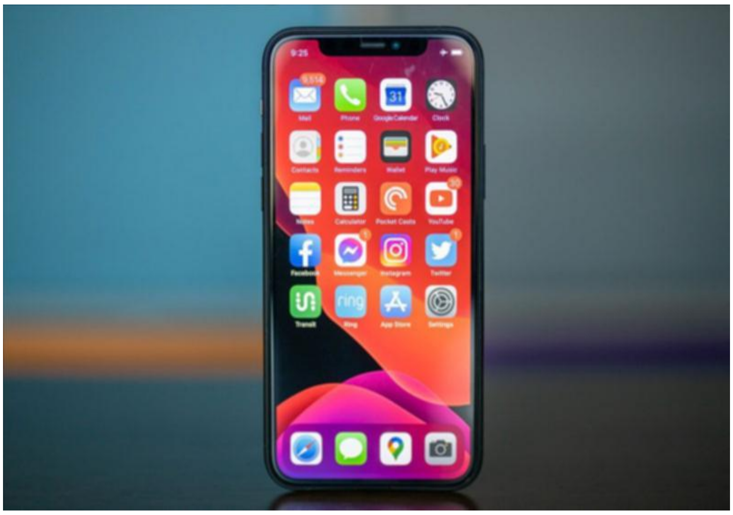
Grayscale

5.Stack 2 photos of the same size



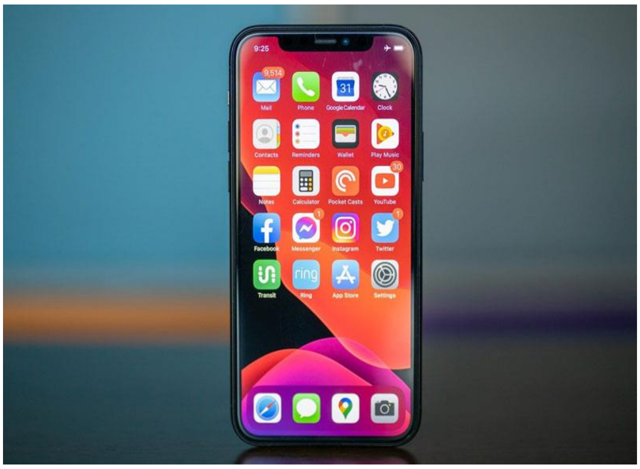
6.Blur image

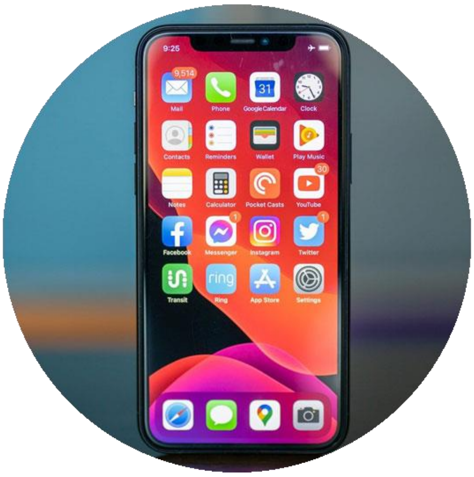
Original



Blur image

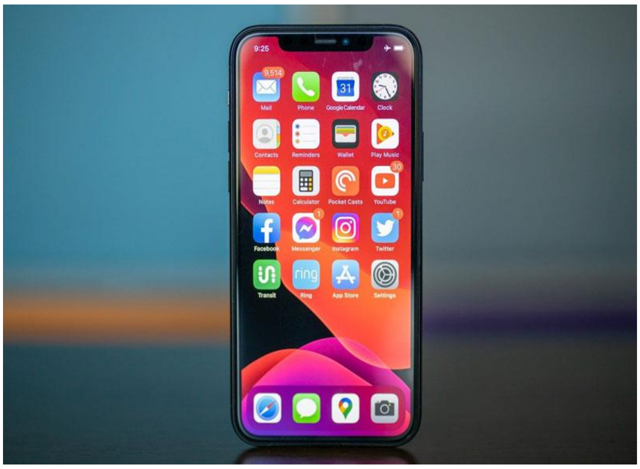
7.Crop the picture with a circle frame

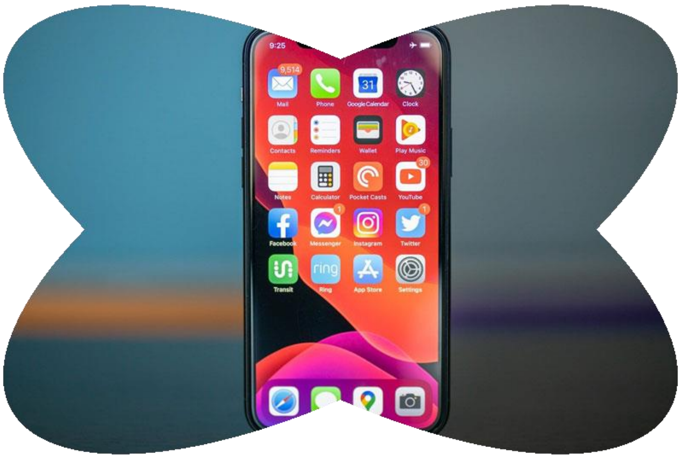




Original

8.Crop the image with the frame as 2 diagonal ellipses





Original

REFERENCES

https://note.nkmk.me/en/python-opencv-numpy-alpha-blend-mask/#:~:text=Complex%20alpha%20blending%20and%20masking%20with%20NumPy,-In%20the%20example&text=Gradation%20images%20can%20be%20generated%20using%20NumPy.&text=It%20can%20be%20composited%20by,value%20of%20the%20gradation%20image.&text=It%20is%20also%20easy%20if%20you%20want%20to%20mask%20with%20another%20image.

https://www.dfstudios.co.uk/articles/programming/image-programming-algorithms/image-processing-algorithms-part-5-contrast-adjustment/

https://github.com/BenyaminZojaji/image_processing

https://stackoverflow.com/questions/51486297/cropping-an-image-in-a-circular-way-using-python

https://stackoverflow.com/questions/25050899/producing-an-array-from-an-ellipse

https://www.analyticsvidhya.com/blog/2021/05/image-processing-using-numpy-with-practical-implementation-and-code/

https://en.wikipedia.org/wiki/Box_blur