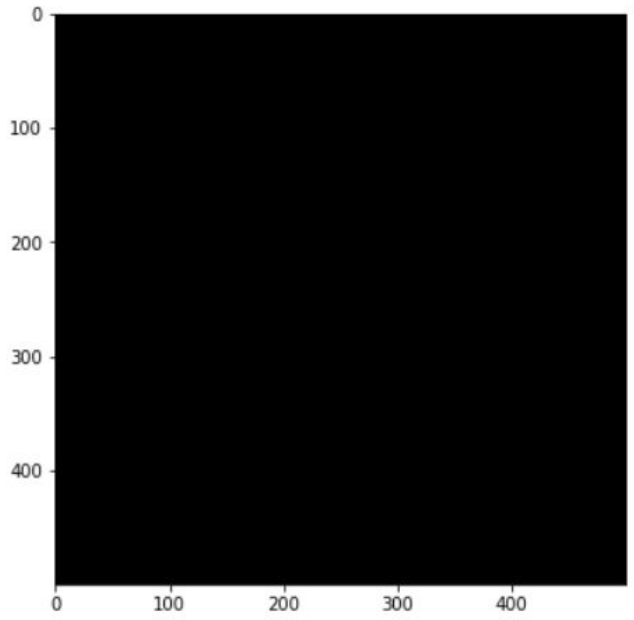
Drawing with Opencv

- Khởi tạo một ảnh màu đen có kích thước 500x500x3.

```
: 1 img = np.zeros([500,500,3])
  2 plt.figure(figsize=(6, 6))
  3 plt.imshow(img)
```

```
: <matplotlib.image.AxesImage at 0x7f8ed2521040>
```

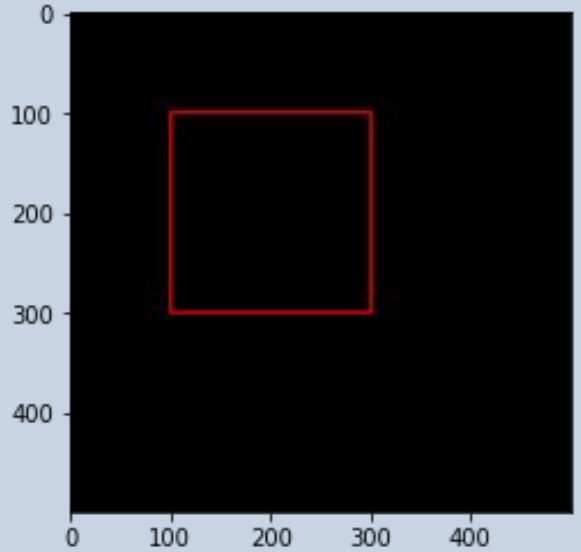


Drawing with Opencv

Vẽ một hình vuông:
Sử dụng hàm cv2.rectangle trong opencv:
Trong đó:

cp : là ảnh đầu vào.
(100, 100) , (300, 300) : là tọa độ điểm top-left và bottom-right.
(255,0,0) : là màu của đường vẽ.
2 : là thickness.

```
:  1 cp = img.copy()
   2 rectang = cv2.rectangle(cp, (100, 100), (300, 300), (255,0,0), 2)
   3 plt.imshow(rectang)
```

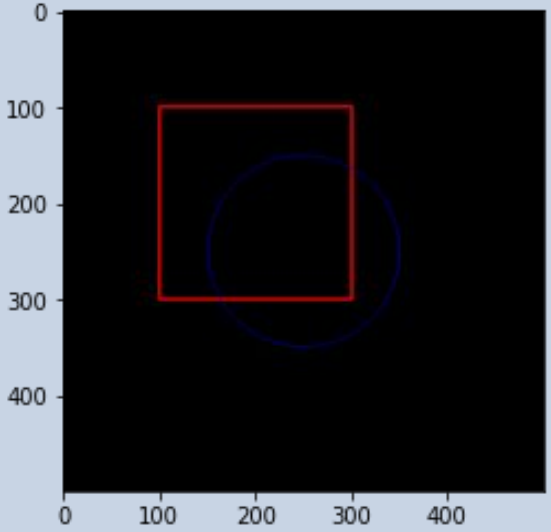


Drawing with Opencv

Vẽ một hình tròn:
Sử dụng hàm cv2.circle trong opencv:
Trong đó:

cp : là ảnh đầu vào.
(250, 250): là tọa độ tâm của đường tròn.
100 (radius): bán kính.
(0,0,255) : là màu của đường vẽ.
1 : là thickness.

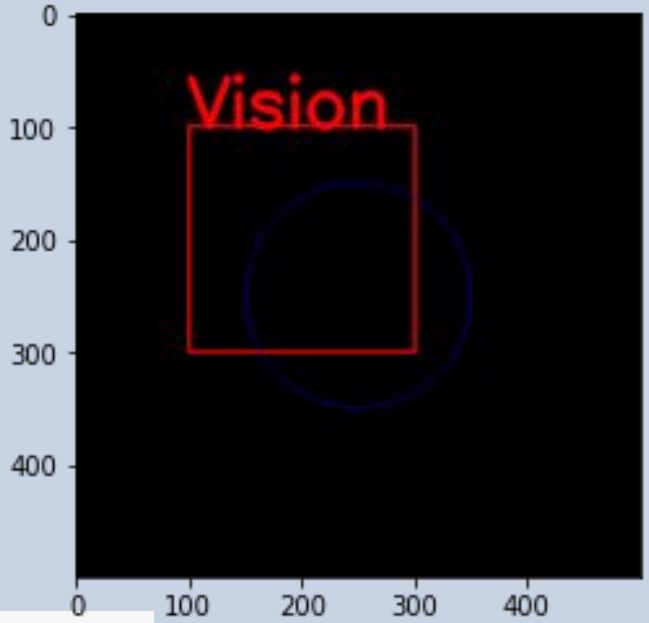
```
cir = cv2.circle(cp, (250, 250), 100, (0, 0, 255), 1)
plt.imshow(cir)
```



Drawing with Opencv

Viết chữ:
Sử dụng hàm cv2.puttext trong opencv:
Trong đó:

- cp: là ảnh đầu vào.
- 'Vision': là input text.
- (100, 100): tọa độ viết chữ.
- font: font style.
- 2: font size.
- (255, 0, 0): là màu của chữ.
- 5: là thickness
- cv2.LINE_AA: là LineType



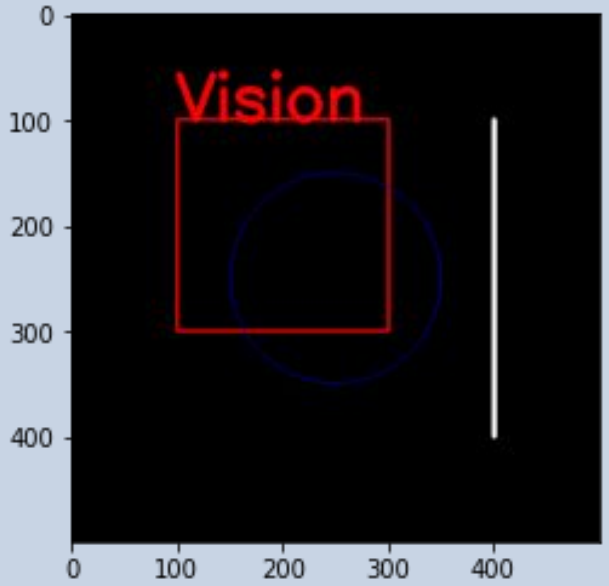

```
font = cv2.FONT_HERSHEY_SIMPLEX
new_img = cv2.putText(cp,'Vision',(100, 100), font, 2,(255,0,0),5,cv2.LINE_AA)
plt.imshow(new_img)
```

Drawing with Opencv

Vẽ đường thẳng:
Sử dụng hàm cv2.line trong opencv:
Trong đó:

cp: là ảnh đầu vào.
(400, 100): tọa độ điểm xuất phát.
(400, 400): tọa độ điểm kết thúc.
(255, 255, 255): là màu của line.
3: là thickness.

```
line = cv2.line(cp, (400, 100), (400, 400), (255, 255, 255), 3)
plt.imshow(line)
```

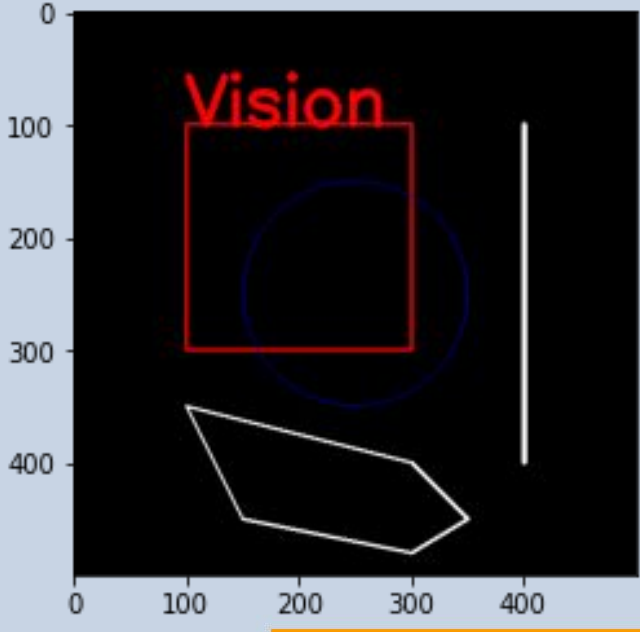


Drawing with Opencv

Vẽ đa giác:
Sử dụng hàm cv2.polyline trong opencv:
Trong đó:

- cp: là ảnh đầu vào.
- pts là tập tọa độ các đỉnh của đa giác.
- True: là True khi đa giác kín.
- (255, 255, 255): là màu của line.
- 2: là thickness.

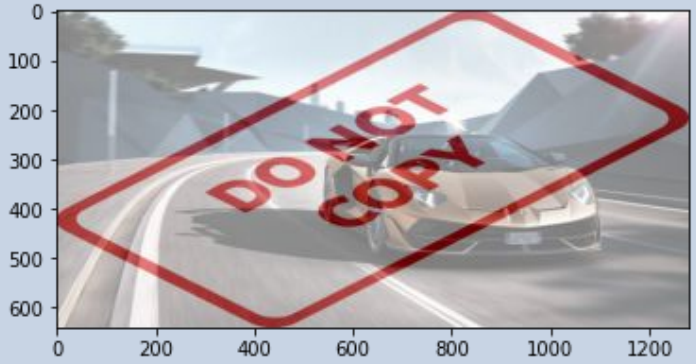
```
pts = np.array([[100, 350], [300, 400],
                [350, 450], [300, 480],
                [150, 450], [100, 350]],
               np.int32)

pts = pts.reshape((-1, 1, 2))
print(pts)

a = cv2.polylines(cp, [pts],
                  True, (125, 125, 125), 2)
plt.imshow(a)
```



Drawing with Opencv

Cộng ảnh:


```
1  dst = cv2.addWeighted(img1, 0.5, img2_resized, 0.5, 0.0)
2  plt.imshow(dst)
```



Drawing with Opencv

Dán ảnh:


```
1 img1[0:small_img.shape[0], 0:small_img.shape[1] ] = small_img
2 plt.imshow(img1)
```

Drawing with Opencv

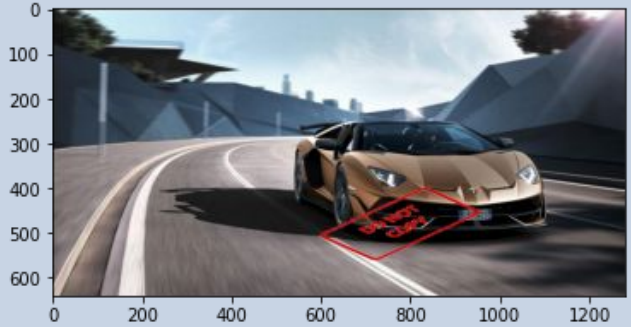
Cắt ảnh:


```
1 roi = img1[400:560, 600:955]
2 plt.imshow(roi)
```

Drawing with Opencv

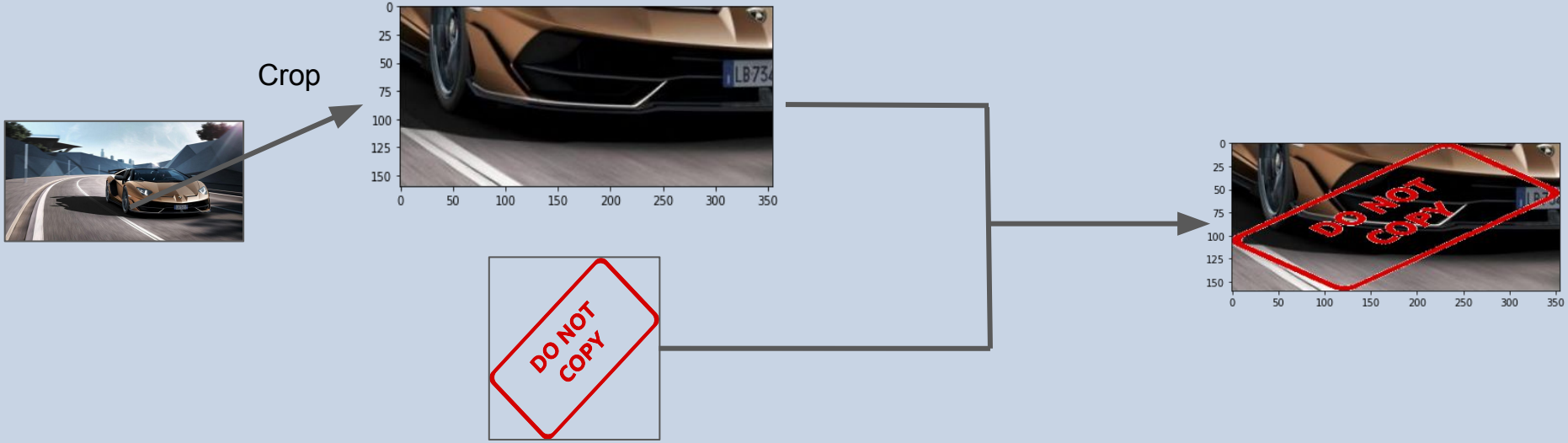
Trộn ảnh:

?



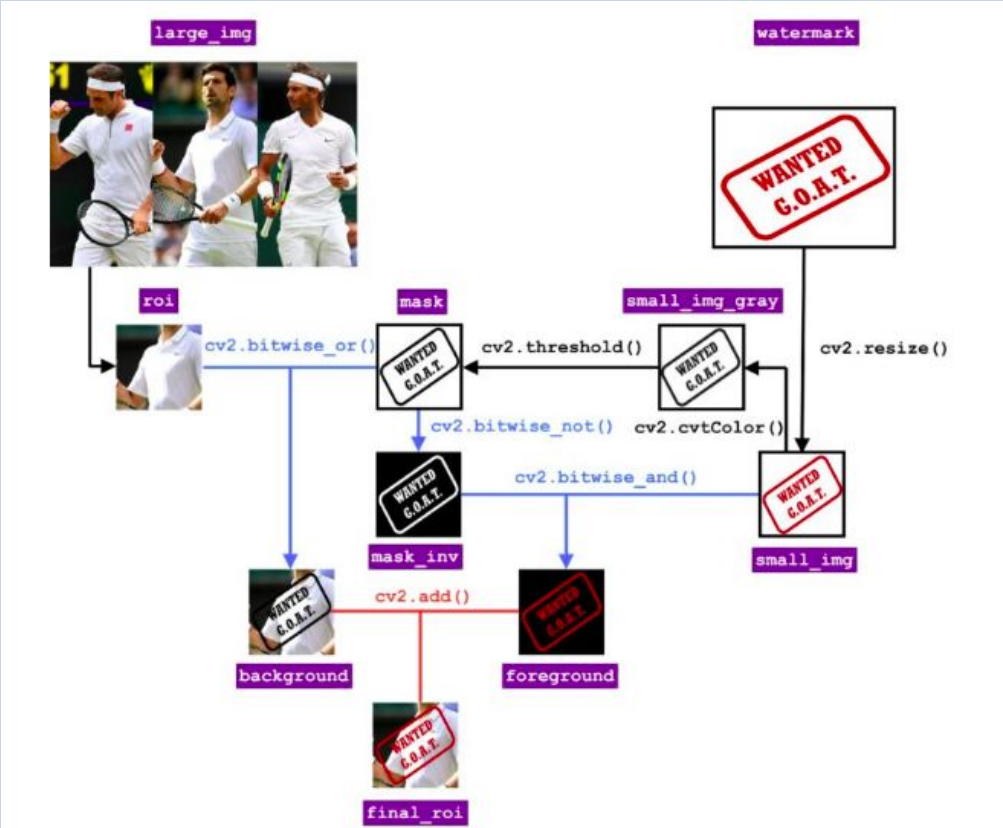
Drawing with Opencv

Trộn ảnh:



Drawing with Opencv

Trộn ảnh:



Drawing with Opencv

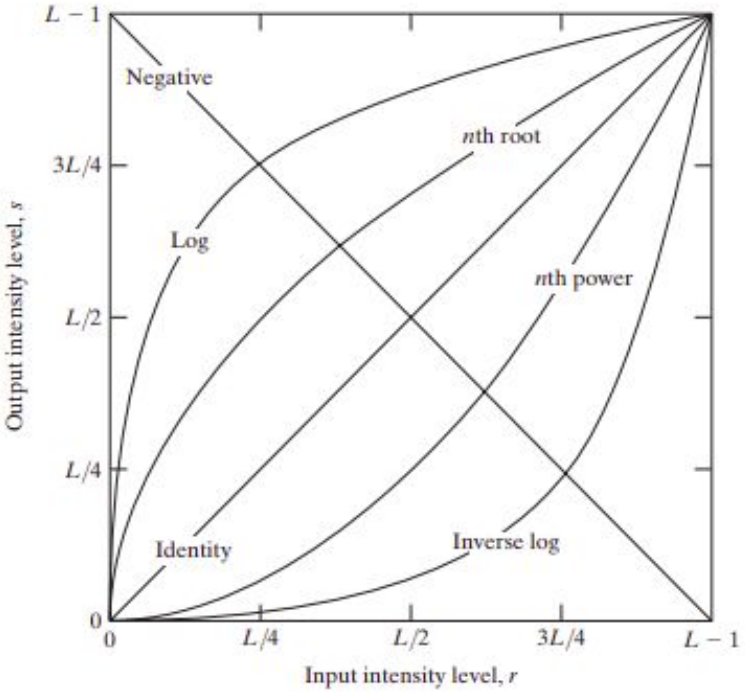
Trộn ảnh:

Intensity Transformations

1. Log Transformations

c = 225 / (log(1 + m))
S = c * log(1 + r)

Trong đó:
- S là cường độ đầu ra
- r >= 0 là cường độ đầu vào của từng pixel
- c là hằng số tỷ lệ
- m là giá trị lớn nhất trong ảnh

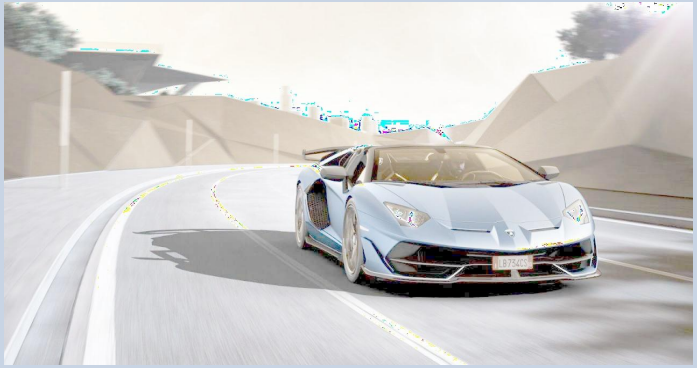


Intensity Transformations

1. Log Transformations

$$c = 225 / (\log(1 + m))$$
$$S = c * \log(1 + r)$$

```
# Open the image.
img = cv2.imread('/home/bau/Downloads/xe.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# Apply log transform.
c = 255/(np.log(1 + np.max(img)))
log_transformed = c * np.log(1 + img)

# Specify the data type.
log_transformed = np.array(log_transformed, dtype = np.uint8)
```



Intensity Transformations

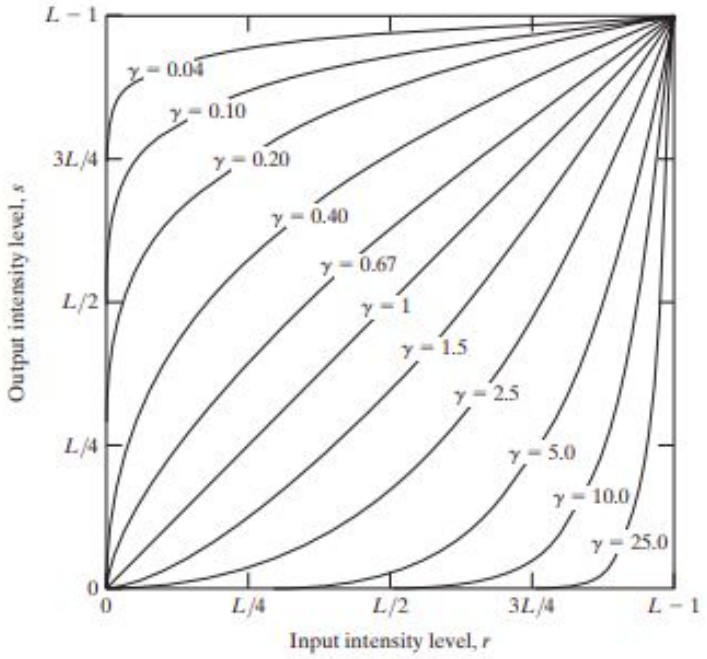
1. Power - Law (Gamma) Transformations

$$s = cr^{\gamma}$$

Trong đó:

c và γ là hằng số dương

$r \geq 0$ là cường độ đầu vào của tường pixel

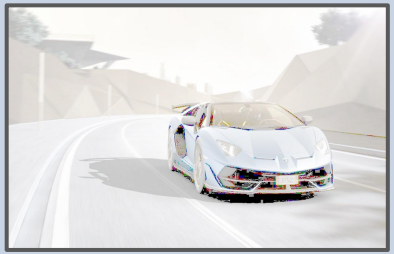


Intensity Transformations

1. Power - Law (Gamma) Transformations

$$s = cr^{\gamma}$$

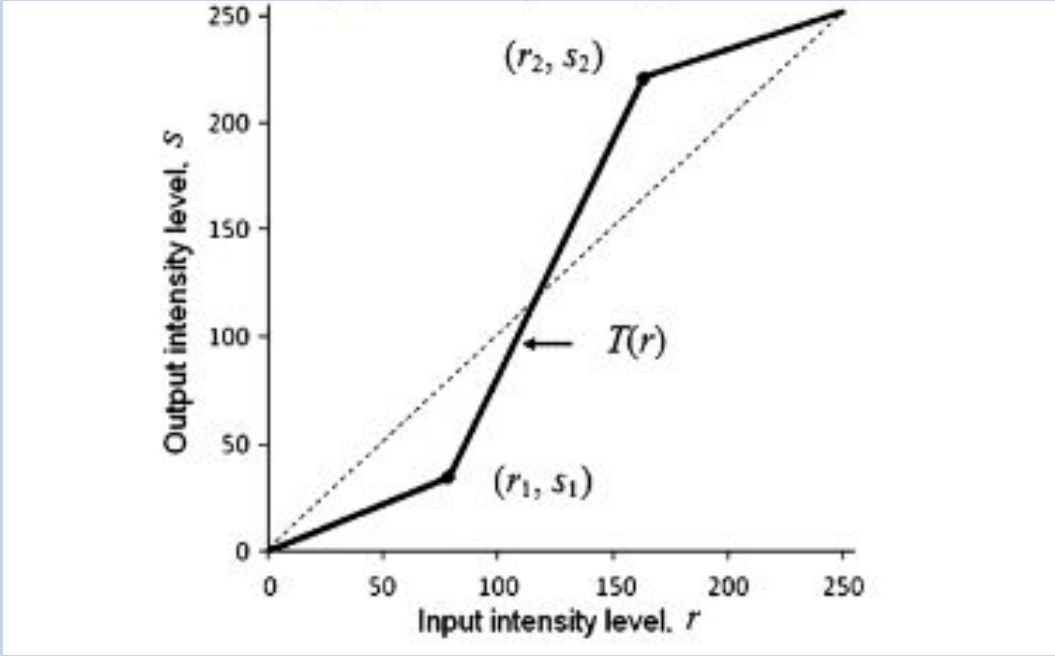
```
i = 1
for gamma in [0.1, 0.5, 1.2, 2.2]:
    # Apply gamma correction.
    gamma_corrected = np.array(255*(img / 255) ** gamma, dtype = 'uint8')
    plt.subplot(2,2,i)
    plt.imshow(gamma_corrected)
    i +=1
```




Intensity Transformations

1. **Piecewise - Linear Transformation (Contrast)**

```
Contrast =  (I_max - I_min)/(I_max + I_min)
```



PHENIKAA UNIVERSITY

Intensity Transformations

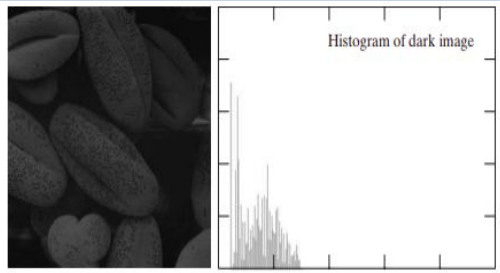
1. Piecewise - Linear Transformation (Contrast)

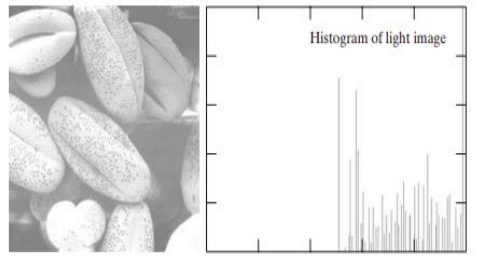
```python
img = cv2.imread('/home/bau/Downloads/xe.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
def pixelVal(pix, r1, s1, r2, s2):
    if (0 <= pix and pix <= r1):
        return (s1 / r1)*pix

    elif (r1 < pix and pix <= r2):
        return ((s2 - s1)/(r2 - r1)) * (pix - r1) + s1

    else:
        return ((255 - s2)/(255 - r2)) * (pix - r2) + s2


# Define parameters.
r1 = 70
s1 = 0
r2 = 140
s2 = 255

# Vectorize the function to apply it to each value in the Numpy array.
pixelVal_vec = np.vectorize(pixelVal)

# Apply contrast stretching.
contrast_stretched = pixelVal_vec(img, r1, s1, r2, s2)
```

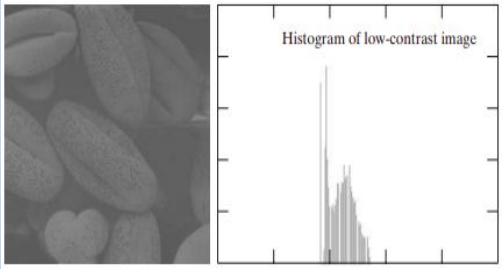
$$\text{Contrast} = \frac{(I_{max} - I_{min})}{(I_{max} + I_{min})}$$

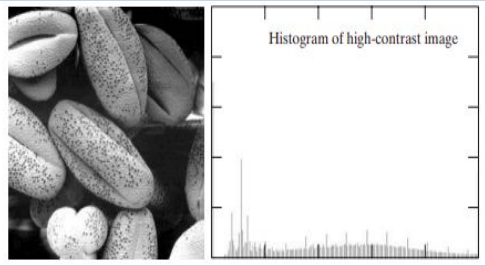
Intensity Transformations

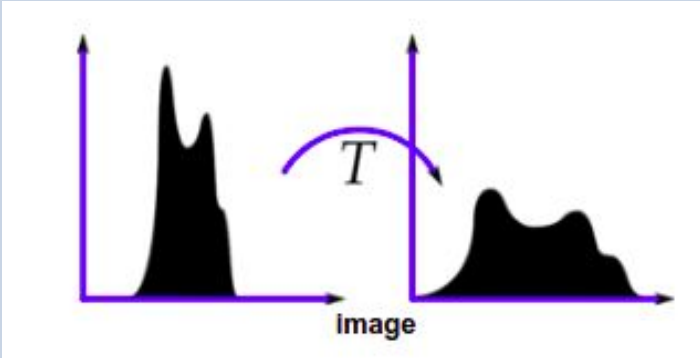
1. Equalize Histogram











PHENIKAA UNIVERSITY

Intensity Transformations

1. Equalize Histogram

```
hist = cv2.calcHist([gray], channels=[0],histSize=[256],ranges=(0,255),mask=None)
plt.plot(hist)

hist2 = cv2.calcHist([hist_eq_gray],channels=[0],histSize=[256],ranges=(0,255),mask=None)
plt.plot(hist2)
```