Reading image

Đọc ảnh từ thiết bị

Sử dụng hàm cv2.imread(

filename: str,

flags:int = …)

```
img = cv2.imread('lamborghini_huracan_sterrato.png')
plt.imshow(img)
```

Reading image

Đổi kênh màu BRG thành RGB

Sử dụng hàm cv2.cvtColor(src: Matlike,code:int)

Hoặc

Đảo ngược kênh màu của ảnh hiện tại

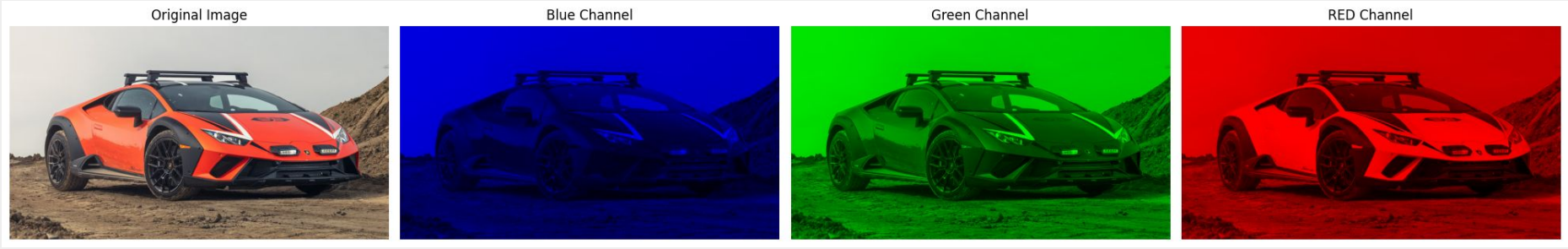
```
img_rgb = cv2.cvtColor(src=img, code=cv2.COLOR_BGR2RGB)
plt.imshow(img_rgb)
```

```
img_rgb = img[:,:,::-1]
plt.imshow(img_rgb)
```

Reading image

Tách các kênh màu từ ảnh

```
red = img_rgb.copy()
green = img_rgb.copy()
blue = img_rgb.copy()

red[:,:,1] = 0
red[:,:,2] = 0

green[:,:,0] = 0
green[:,:,2] = 0

blue[:,:,0] = 0
blue[:,:,1] = 0
```



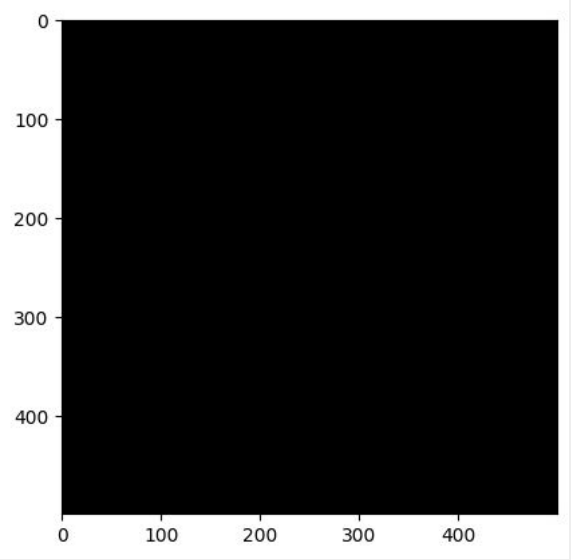
Cropping image

```
img_cropped = img_rgb[200:600, 200:1200]
plt.imshow(img_cropped)
```

Drawing with OpenCV

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

```
img = np.zeros([500, 500, 3], dtype=np.uint8)
plt.imshow(img)
```

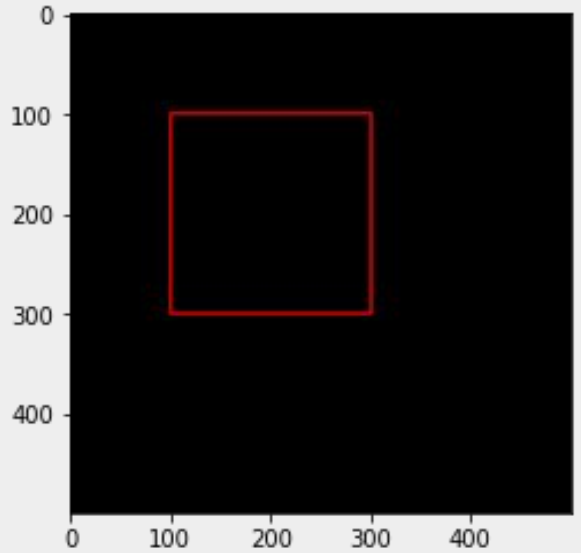


Drawing with OpenCV

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

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

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

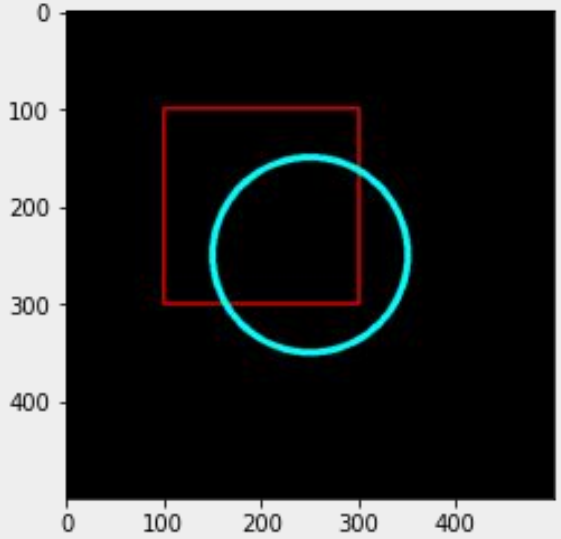


Drawing with OpenCV

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

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

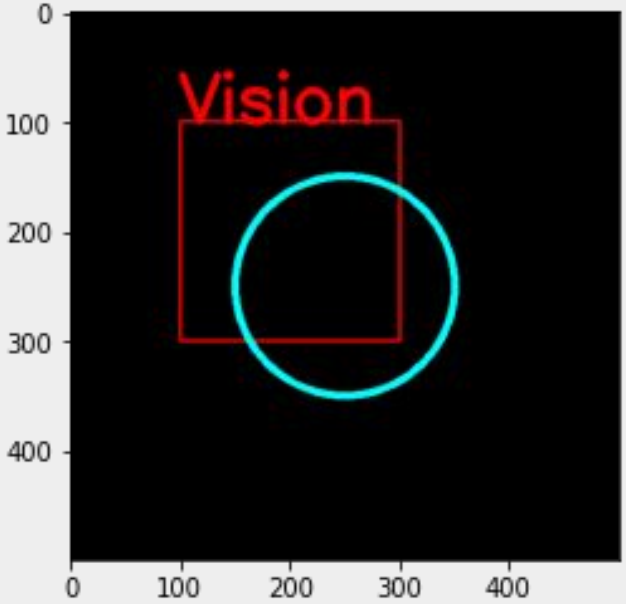
```
1 cir = cv2.circle(cp, (250, 250), 100, (0, 0, 255), 1)
2 plt.imshow(cir)
```



Drawing with OpenCV

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

cp:	ảnh đầu vào
“Vision”:	input text
(100, 100):	tọa độ viết chữ (bottom-left)
font:	font style
2:	font size
(255, 0, 0):	mã màu của chữ (RGB)
5:	thickness
cv2.LINE_AA:	LineType



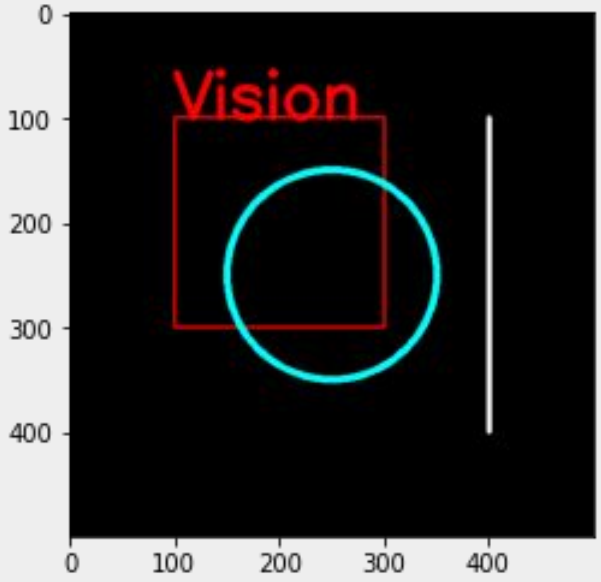

```
font = cv2.FONT_HERSHEY_SIMPLEX
new_img = cv2.putText(cp,'Vision',(100, 100), font, 2,(255,0,0),5,cv2.LINE_AA)
plt.imshow(new_img)
```

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)
```

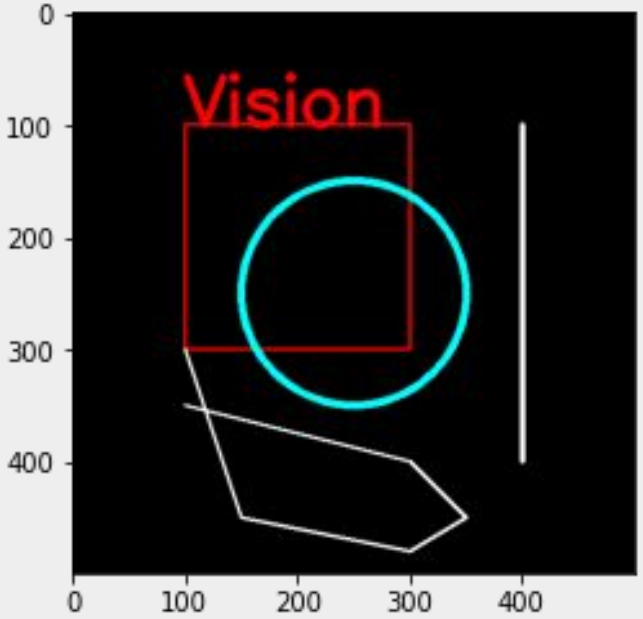


Drawing with OpenCV

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 nối thành 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, 300]],
               np.int32)

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

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



Reading video

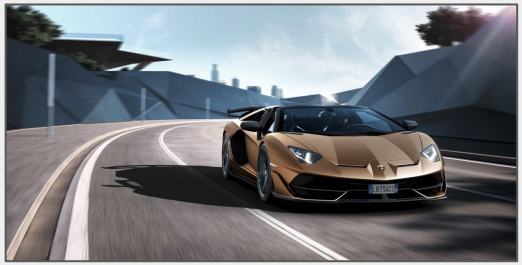
Đọc video bằng OpenCV

- Cấu trúc code thành dạng function, có parameter điều chỉnh tốc độ video
- Thực hiện đọc 1 video bất kỳ, vẽ “Do not copy” lên từng frame của video, sau đó lưu lại dưới dạng .mp4

```python
import cv2

video_path = 'video.mp4' # Path to the video file

cap = cv2.VideoCapture(video_path) # Open the video file

while cap.isOpened():
    ret, frame = cap.read() # Read the frame, ret is True if frame is read correctly
    if not ret:
        break
    cv2.imshow('Frame', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'): # Press 'q' to quit
        break

cap.release()
cv2.destroyAllWindows()
```

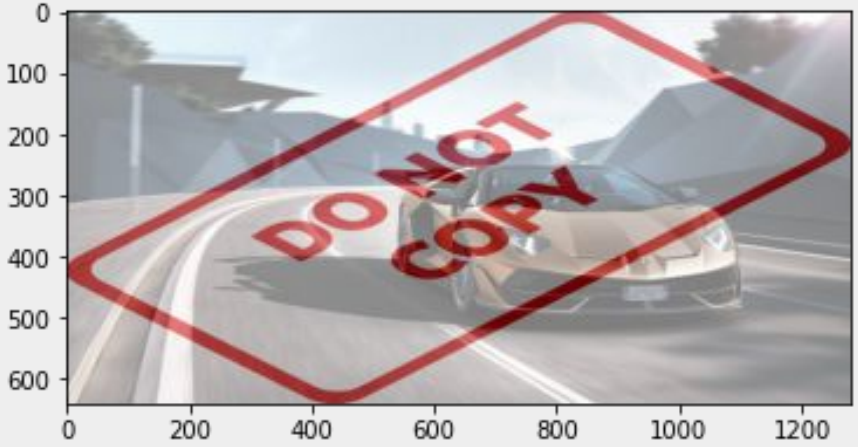
Arithmetic Operations

Cộng ảnh:



+

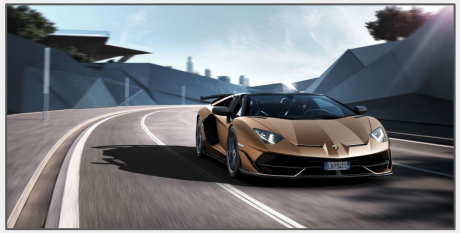
=




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

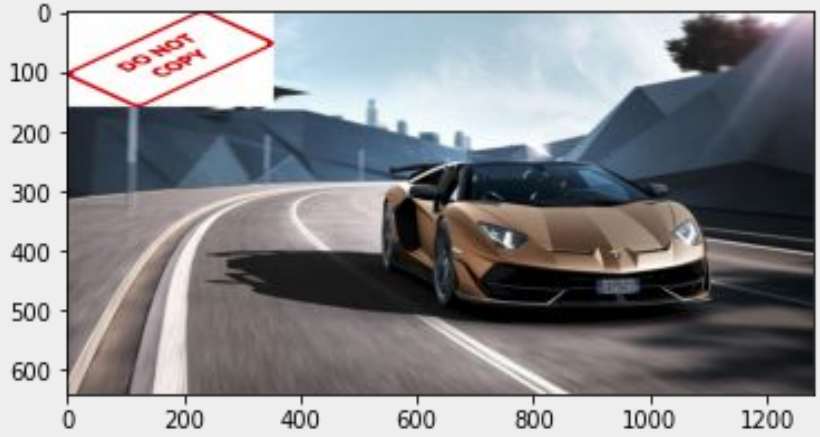
Arithmetic Operations

Dán ảnh:



+

=

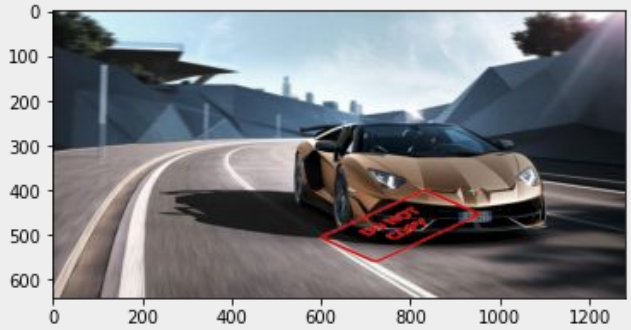



```
1 img1[0:small_img.shape[0], 0:small_img.shape[1] ] = small_img
2 plt.imshow(img1)
```

Arithmetic Operations

Trộn ảnh:

?



Arithmetic Operations

Trộn ảnh:

- Thực hành trộn ảnh dựa vào hướng dẫn theo hình bên.

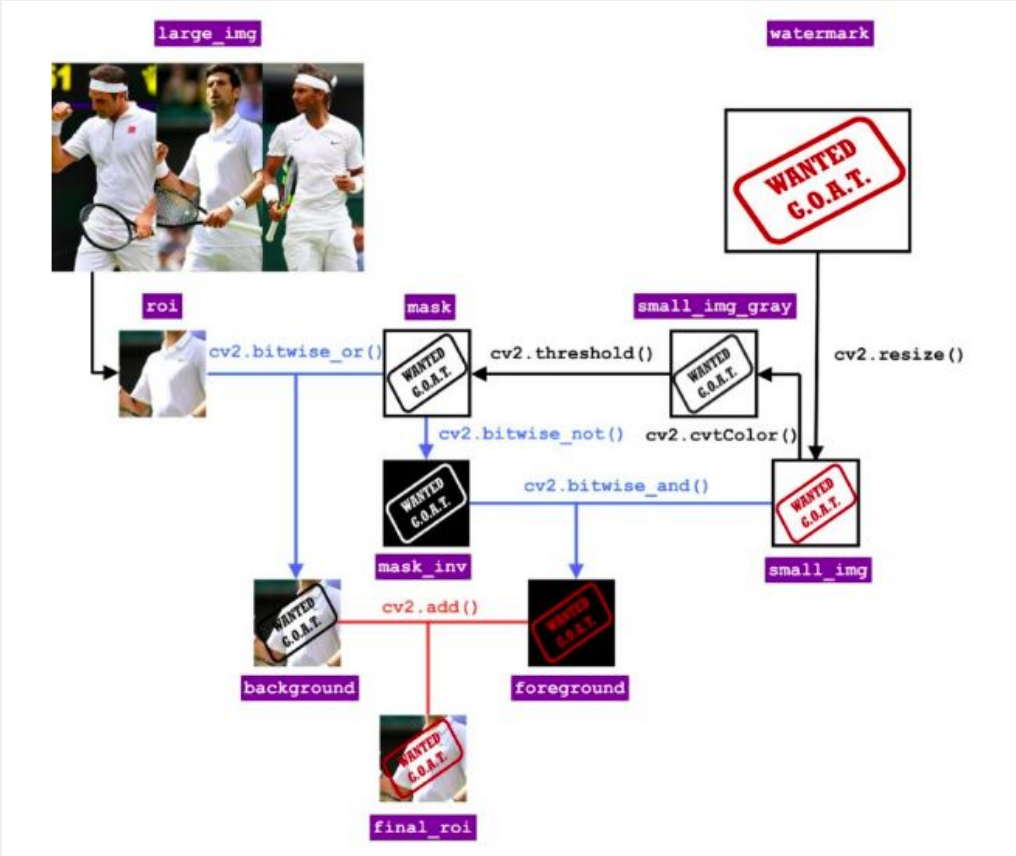
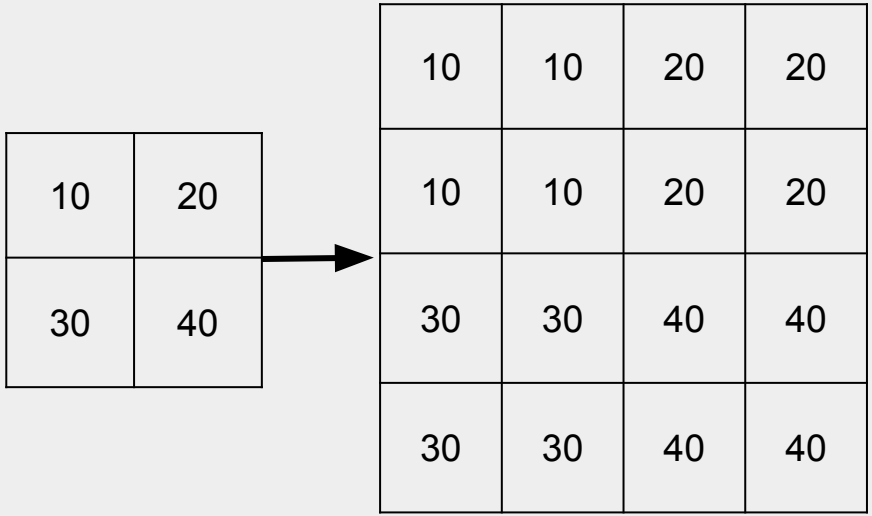


Image Interpolation

Image Scaling: Nearest Neighbor Interpolation




```python
def scaling_nearest(image: np.ndarray, scale: float) -> np.ndarray:
    """
    Scale the image using nearest neighbor interpolation.
    parameters:
        image: numpy.ndarray - an image
        scale: float - scale factor
    return:
        new_image: numpy.ndarray - the scaled image

    """
    return
```

Image Interpolation

Image Scaling: Nearest Neighbor Interpolation using KNN

Bước 1: Tiền xử lý dữ liệu

- Chuẩn hóa tọa độ tương ứng với kích thước mới (X_train)
- Độ sáng của từng pixel (Y_train)

Bước 2: Huấn luyện mô hình KNN

- Thử nghiệm với số lượng n_neighbors tăng dần

Bước 3:

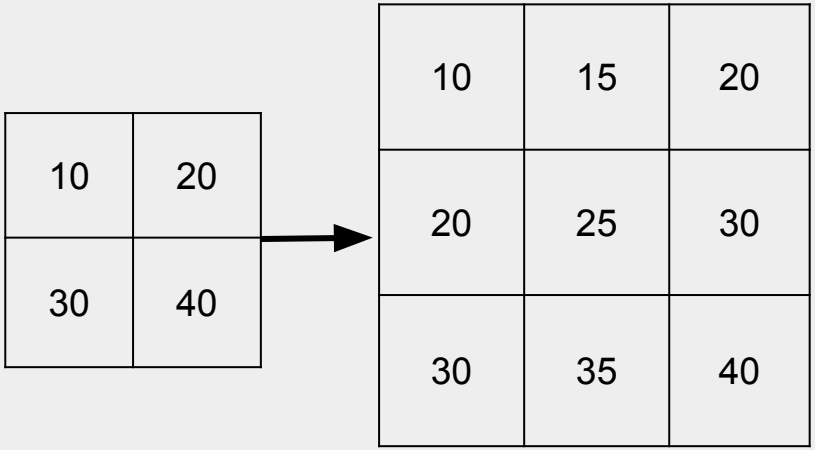
- Tạo ảnh rỗng dựa trên kích thước đã cho
- Thay thế tọa độ từng pixel với giá trị tương ứng của mô hình

```python
def get_intensity_coordinates(img: np.ndarray) -> tuple:
    """
    Get the intensity and coordinates of the image.
    parameters:
        img: numpy.ndarray - an image
    return:
        intensity: numpy.ndarray - the intensity of the image
        coordinates: numpy.ndarray - the normalized coordinates of the image

    """
```

```python
def train_knn_model(coordinates: np.ndarray,
                    intensity: np.ndarray,
                    n_neighbor:int) -> KNeighborsRegressor:
    """
    Train the KNN model.
    parameters:
        coordinates: numpy.ndarray - the normalized coordinates of the image
        intensity: numpy.ndarray - the intensity of the image
    return:
        model: KNeighborsRegressor - the trained KNN model

    """
    model = KNeighborsRegressor(n_neighbors=n_neighbor).fit(coordinates, intensity)
    return model
```

Image Interpolation

Image Scaling: Bilinear interpolation




```python
def scaling_bilinear(image: np.ndarray, scale: float) -> np.ndarray:
    """
    Scale the image using bilinear interpolation.
    parameters:
        image: numpy.ndarray - an image
        scale: float - scale factor
    return:
        new_image: numpy.ndarray - the scaled image

    """
```

$$v(x, y) = ax + by + cxy + d$$

Image Interpolation

Image Scaling with OpenCV

- Thực hiện thay đổi kích thước ảnh với thư viện OpenCV bằng cách sử dụng hàm cv2.resize(src, dsize, fx, fy, interpolation) với interpolation(INTER_NEAREST, INTER_LINEAR)
- So sánh kết quả nhận được của thư viện và của mình.

```
img_resized = cv2.resize(img_rgb, None, fx=0.3, fy=0.3, interpolation=cv2.INTER_LINEAR)
plt.imshow(img_resized)
```

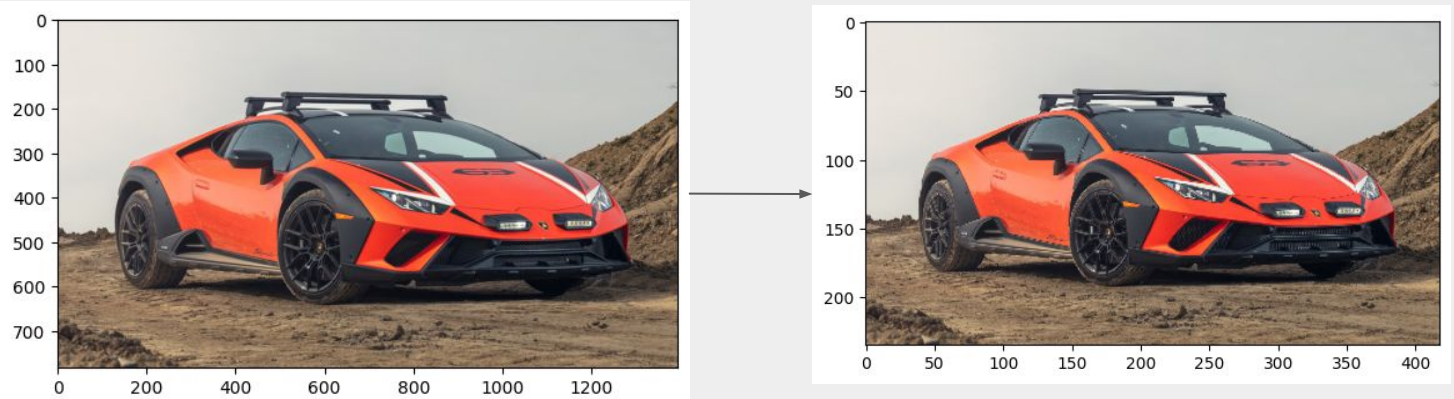


Image Interpolation

Image Rotation: Nearest Neighbor Interpolation

- Thực hiện viết hàm xoay ảnh
- So sánh với thư viện OpenCV với các góc [15, 45, -45, -30] độ
- Điều chỉnh code sao cho khi xoay không bị mất ảnh

$$x = v\cos\theta - w\sin\theta$$
$$y = v\cos\theta + w\sin\theta$$

```python
def rotate_nearest_neighbor(image: np.ndarray, angle: float) -> np.ndarray:
    """
    Rotate the image using nearest neighbor interpolation.
    parameters:
        image: numpy.ndarray - an image
        angle: float - angle in degrees
    return:
        new_image: numpy.ndarray - the rotated image
        """
```

45 degrees

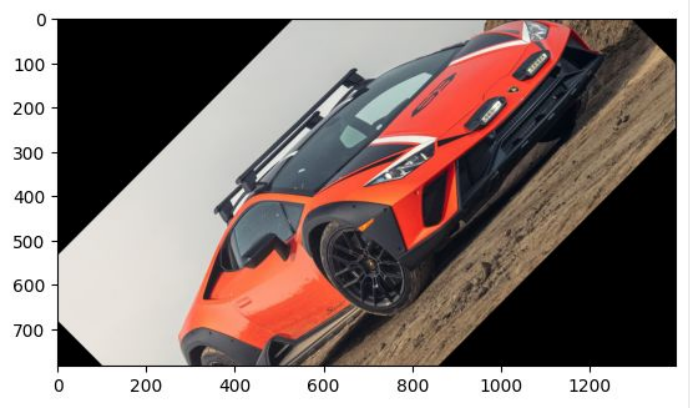


Image Negative

$$s = L - 1 - r$$

s: mức sáng mới

L: số mức sáng của ảnh

r: mức sáng tại pixel hiện tại