Day 6: Image Resizing, Cropping, and Copying

Outcomes:

- Resize images using OpenCV
- Crop images using array slicing
- Copy and modify image regions
- Understand how image dimensions change after resizing and cropping

Resizing an Image

Resizing changes the height and width of an image.

Syntax of `resize()` function:

```
cv2.resize(img, dsize)
```

- `dsize = (width, height)` (not height, width) Exact output size.
- `resize` returns a new image (does not modify the original image)
- `fx`, `fy` are horizontal and vertical scaling factors.

Declaring Exact Dimensions (common usage)

```
resized = cv2.resize(img, (width, height))
```

Example:

```
resized = cv2.resize(img, (300, 300))
```

- Resizes the image array `img`, to exactly 300 pixels in width and 300 pixels in height.

7-resizingImages.py

```python
import cv2 as cv


img = cv.imread("OpenCV-codes/Images/image.png")


#Resizing by Declaring Exact Dimensions
resized = cv.resize(img, (300,400))
cv.imshow("resized image", resized)


cv.waitKey(0)
cv.destroyAllWindows()
```

Resizing Using Scale Factors

```
resized = cv2.resize(img, None, fx=0.5, fy=0.5)
```

- Rescales the width and height to 0.5 times the original dimensions
- i.e., width = original width * 0.5

`fx = 1.0`	Width stays the same
`fx = 0.5`	Width becomes half
`fx = 2.0`	Width becomes double

(Same logic for fy and height.)

Why is `none` required?

- `None` tells OpenCV that there is no fixed output size
- Without `None`, scaling factors are ignored

Note:

- Aspect Ratio is preserved only if `fx == fy`
- `resize()` returns a new image
- Original image remains unchanged

8-resizingImages.py

```python
import cv2 as cv


img = cv.imread("OpenCV-codes/Images/image.png")


#Resizing Using Scale Factors

resized = cv.resize(img, None, fx=0.5, fy=0.3)

cv.imshow("resized image", resized)


cv.waitKey(0)

cv.destroyAllWindows()
```

Cropping an Image

Cropping uses NumPy slicing.

Syntax:

```
cropped = img[y1:y2, x1:x2]
```

(From row y1 to y2, and column x1 to x2)

Parameters:

y1	Starting y coordinate (top)	x1	Starting x coordinate (left)
y2	Ending y coordinate (bottom)	x2	Ending x coordinate (right)

Note:

- `y2` and `x2` are not included
- Order is always `[y,x]` and not `[x,y]`

9-croppingImages.py

```python
import cv2 as cv


img = cv.imread("OpenCV-codes/images/image.png")

cropped = img[0:490, 490:980]


cv.imshow("Image", cropped)


cv.waitKey(0)

cv.destroyAllWindows()
```

Copying an Image

Shallow Copy (NOT recommended)

```
copy1 = img
```

- It creates another reference to the image
- both images point to the same data
- changes affect both images

10-copyingImages.py

```
import cv2 as cv


img = cv.imread("OpenCV-codes/images/image1.png")
copy = img


cv.line(copy, (245,735), (735,245), (0,255,0), 2)
cv.imshow("image",img)


cv.waitKey(0)
cv.destroyAllWindows()
```

Note: Here we can see that any changes made to `copy` are also applied to `img` which is not a good practice.

Proper Copy

```
copy1 = img.copy()
```

- this creates a new image with new memory
- safe to modify

11-copyingImages.py

```
import cv2 as cv


img = cv.imread("OpenCV-codes/images/image1.png")
copy = img.copy()


cv.line(copy, (245,735), (735,245), (0,255,0), 2)
cv.imshow("copy",copy)
cv.imshow("image", img)


cv.waitKey(0)
cv.destroyAllWindows()
```

Note: Here any changes made to `copy` of `img` does not affect the `img` at all. This is proper copying.

Region of Interest (ROI)

A **Region of Interest (ROI)** is a **specific part of an image** that you want to focus on instead of processing the whole image.

In OpenCV, ROI is just a cropped region.

What does ROI mean?

- ROI is **cropping**, just used with a **purpose**.
- Instead of working on the entire image
- You select only the useful area

Example uses:

- face area in face detection
- number plate in **ANPR** (Automatic Number Plate Recognition)
- object bounding box
- eyes, hands, text area

Syntax:

```
roi = img[y1:y2, x1:x2]
```

- Useful for object detection, tracking, editing

ROI vs Cropping

- If you say cropping → you're just extracting
- If you say ROI → you plan to **process only that region**
- Cropping can be saved/ displayed
- ROI is often processed, modified, analysed and written back to the original image.

Note: All ROIs are cropped regions, but not all crops are ROIs.

When to copy an image?

- You want to modify an image but keep the original image
- You want to draw temporary shapes

Note: Always use `img.copy()` when you want an independent image.