Module 3

Introduction to Image Processing

▼ What is the purpose of converting color images to grayscale?

- Converting an image to grayscale reduces the number of pixels that need to be processed
- Note: grayscale images are represented in one channel

▼ What is the purpose of augmenting image data using transformations (scaling, rotation, and translation)?

- This technique increases the probability that your model will recognize objects when they appear in any form or shape within your images.
- For example, you can train your model to recognize a puppy in an image. By augmenting this image, you can train your model to recognize the same puppy in an image when it is facing the opposite direction.

▼ What's an example of a color adjustment that reduces noise in an image?

removing a background color in an image

▼ How can you do color conversion in OpenCV?

- cvtColor(input_image, flag) where the flag determines the type of conversion
- e.g. for BGR to Gray conversion, you use the flag COLOR_BGR2GRAY

```
# converts image to grayscale

import cv2
img = cv2.imread('kitten.jpg')
```

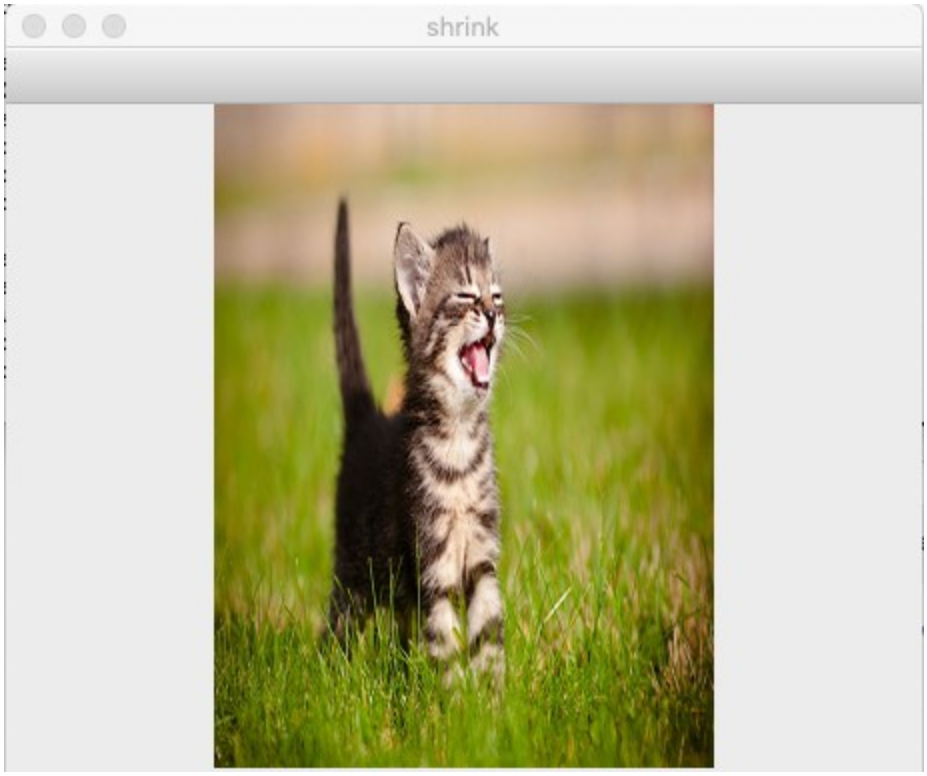
```
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

cv2.imshow('grayscale', gray_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

▼ How can you resize/rescale in OpenCV?

- cv2.resize()
- The size of the image can be specified manually, or you can specify the scaling factor.
- Different interpolation methods are used
  - cv2.INTER_AREA for shrinking
  - cv2.INTER_CUBIC for zooming

```
# resizes the image so that it is 2 times as wide
# it then resizes the image so that it is ½ as wide

import cv2
```

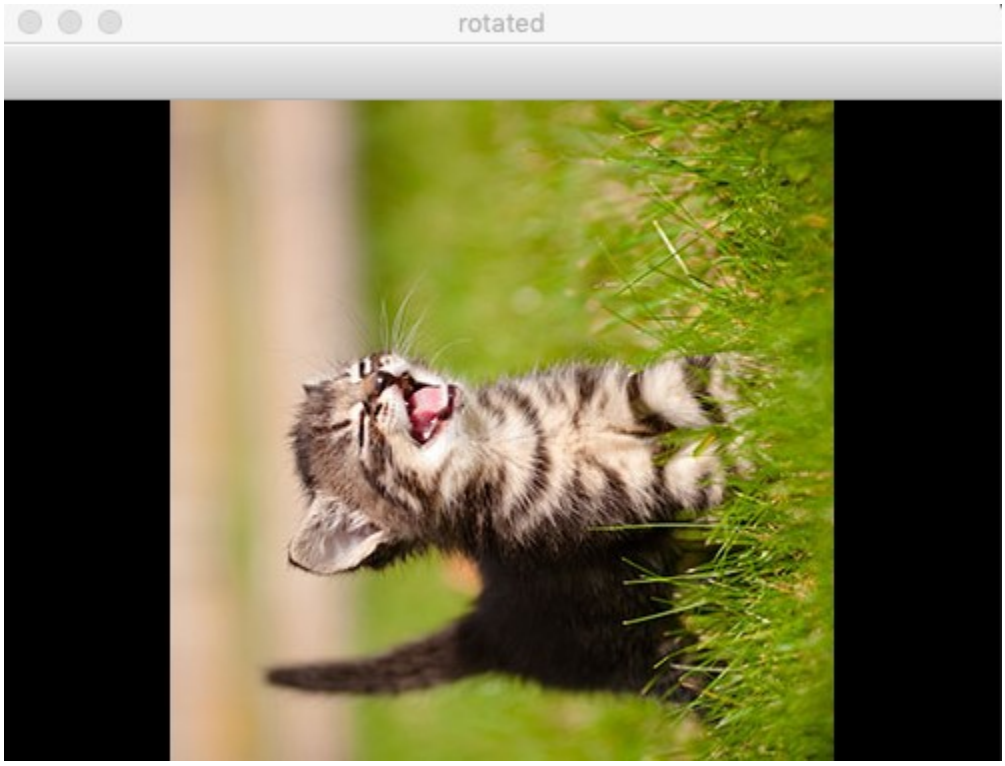
```
img = cv2.imread('kitten.jpg')

height, width = img.shape[:2]
zoom = cv2.resize(img, (2*width, 1*height), interpolation = cv2.INTER_CUBIC)
shrink = cv2.resize(img, (int(0.5*width), 1*height), interpolation = cv2.INTER_AREA)

cv2.imshow('zoom', zoom)
cv2.imshow('shrink', shrink)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



▼ How can you rotate an image in OpenCV?

- OpenCV provides a transformation function, **cv2.warpAffine**, with which you can perform all kinds of transformations.
- For rotation of an image, you must first use **cv2.getRotationMatrix2D** to calculate the transformation matrix and then pass it to **cv2.warpAffine** to perform the transformation on the image.

```
import cv2
img = cv2.imread('kitten.jpg')

rows, cols = img.shape[:2]

R = cv2.getRotationMatrix2D((cols/2, rows/2), 90, 1)
rot_img = cv2.warpAffine(img, R, (cols, rows))

cv2.imshow('rotated', rot_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



▼ How can you translate an image in OpenCV?

- To augment an image using a translation, you must first define the transformation matrix specifying the values you want to shift the image position.
- Next, make it into a Numpy array of type *np.float32* and then pass it into the **cv2.warpAffine()** function to perform the transformation on the image.

```
# translates the kitten image 100 pixels to the right and 50 pixels down

import numpy as np
import cv2
img = cv2.imread('kitten.jpg')

rows, cols = img.shape[:2]

T = np.float32([[1,0,100], [0,1,50]])
trans_img = cv2.warpAffine(img, T, (cols, rows))

cv2.imshow('translated', trans_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```