شماره فصل: ۴	تعداد جلسات مورد نیاز: ۲ جلسه
اهداف: – آشنایی با پردازش‌های مقدماتی تصویر (استخراج کانال‌های تصویر، تبدیل به سطح خاکستری، چرخش تصویر، انتقال تصویر، ...)	

استخراج کانال‌های تصویر:

```
## Extract individual Blue (B), Green (G), and Red (R) channels of an BGR
image ##
import cv2
img=cv2.imread('E:/UOK/UOK_ Presentations/UOK_Digital Image
Processing/Python/Images/Fruits.jpeg')
#extract Blue (B) channel
Blue_channel = img[:,:,0]
#extract Green (G) channel
Green_channel = img[:,:,1]
#extract Red (R) channel
Red_channel = img[:,:,2]
#Show images
cv2.imshow('BGR image',img)
cv2.imshow('Blue channel',Blue_channel)
cv2.imshow('Green channel',Green_channel)
cv2.imshow('Red channel',Red_channel)
cv2.waitKey(0)
####
```

تبدیل تصویر رنگی به تصویر سطح خاکستری:

```
## Convert BGR to Gray scale [Gray=(0.114*B)+(0.587*G)+(0.299*R)] ##
import cv2
img=cv2.imread('E:/UOK/UOK_ Presentations/UOK_Digital Image
Processing/Python/Images/Fruits.jpeg')
imgGray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) #Convert BGR color space to
Gray space
cv2.imshow('BGR image',img)
cv2.imshow('Gray image',imgGray)
cv2.waitKey(0)
####
```

دسترسی به ابعاد تصویر و تعداد کانال‌های آن:

```
## Find the shape and size of an image ##
import cv2
img=cv2.imread('E:/UOK/UOK_ Presentations/UOK_Digital Image
Processing/Python/Images/Fruits.jpeg')
imgGray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) #Convert BGR color space to
Gray space
print(img.shape) #For a BGR image it returns (number of Rows, number of
Cloumns, number of Channels)
print(imgGray.shape) #For a 1 channel image it returns (number of Rows,
number of Cloumns)
print(img.size) #For a BGR image it returns number of Rows * number of
Cloumns * number of Channels
```

```
print(imgGray.size) #For a 1 channel image it returns number of Rows *
number of Cloumns
####
```

دسترسی به نوع کلاس داده تصویر:

```
## Find the data type of an image ##
import cv2
img=cv2.imread('E:/UOK/UOK_ Presentations/UOK_Digital Image
Processing/Python/Images/Fruits.jpeg')
print(img.dtype)
####
```

دسترسی به مقدار ارزش پیکسل‌ها در یک موقعیت معین:

```
## Access the pixel values at a desire location##
import cv2
img=cv2.imread('E:/UOK/UOK_ Presentations/UOK_Digital Image
Processing/Python/Images/Fruits.jpeg')
imgGray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) #Convert BGR color space to
Gray space
(B, G, R) = img[20, 50] # Access the pixel's B, G, and R values located at
Row(y)=20, Column(x)=50
Gray_value = imgGray[20, 50] # Access the pixel's gray value located at
Row(y)=20, Column(x)=50
print('Pixel at the desire location - Blue: {}, Green: {}, Red:
{}'.format(B, G, R))
print('Pixel at the desire location - Gray: {}'.format(Gray_value))
####
```

تغییر مقدار ارزش پیکسل‌ها در یک موقعیت معین:

```
## Change pixel values at a desire location##
import cv2
img=cv2.imread('E:/UOK/UOK_ Presentations/UOK_Digital Image
Processing/Python/Images/Fruits.jpeg')
# Access the pixel located at Row(y)=20, Column(x)=50
(B, G, R) = img[20, 50]
print('Pixel at the desire location - Blue: {}, Green: {}, Red:
{}'.format(B, G, R))
# update the pixel at (20, 50) and set it to red
img[20, 50] = (0, 0, 255)
(B, G, R) = img[20, 50]
print('Pixel at the desire location - Blue: {}, Green: {}, Red:
{}'.format(B, G, R))
####
```

برش یک ناحیه مدنظر از تصویر:

```
## Cropping a desire region of an image##
import cv2
img=cv2.imread('E:/UOK/UOK_ Presentations/UOK_Digital Image
Processing/Python/Images/Fruits.jpeg')
cropped_img = img[80:280, 250:700] #Cropping the image at the desire region
(from Row y to yy and from column x to xx)
cv2.imshow('Original image',img)
cv2.imshow('Cropped image',cropped_img)
cv2.waitKey(0)
####
```

تغییر اندازه تصویر:

```
## Image resizing ##
import cv2
img=cv2.imread('E:/UOK/UOK_ Presentations/UOK_Digital Image
Processing/Python/Images/Fruits.jpeg')
scale_percent = 60  #Percent of original size for resizing
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height) #Size of resized image
resized_img = cv2.resize(img, dim, interpolation=cv2.INTER_AREA) #Resize
image
print('Original dimensions : ', img.shape)
print('Resized dimensions : ', resized_img.shape)
cv2.imshow("Original image", img)
cv2.imshow("Resized image", resized_img)
cv2.waitKey(0)
####
```

چرخش تصویر:

```
## Image rotation ##
import cv2
img=cv2.imread('E:/UOK/UOK_ Presentations/UOK_Digital Image
Processing/Python/Images/Fruits.jpeg')
[height, width] = img.shape[:2]
center = (width / 2, height / 2) #Dividing height and width by 2 to get the
center of the image
rotate_matrix = cv2.getRotationMatrix2D(center, 45, 1) #Using
cv2.getRotationMatrix2D(iamage center coordinate, rotation angle, scale) to
get the rotation matrix. Positive degrees rotate the image counterclockwise
and vice versa.
rotated_img = cv2.warpAffine(img, rotate_matrix, (width, height)) #Rotate
the image using cv2.warpAffine (original image, rotation matrix, size of
rotated image (width, height))
cv2.imshow('Original image', img)
cv2.imshow('Rotated image', rotated_img)
cv2.waitKey(0)
####
```

چرخش تصویر و نمایش آن به صورت کامل با کمک جعبه مرزی (Bounding box):

```
## Image rotation with bounding box ##
import cv2
import imutils
img=cv2.imread('E:/UOK/UOK_ Presentations/UOK_Digital Image
Processing/Python/Images/Fruits.jpeg')
rotated_img = imutils.rotate(img, 45) #Image rotation without boundig box.
Positive degrees rotate the image counterclockwise and vice versa.
rotated_img_bound = imutils.rotate_bound(img, -45) #Image rotation with
boundig box. Negative degrees rotate the image counterclockwise and vice
versa.
cv2.imshow('Original image', img)
cv2.imshow('Rotated image without bounding box', rotated_img)
cv2.imshow('Rotated image with bounding box', rotated_img_bound)
cv2.waitKey(0)
####
```

انتقال تصویر:

```
## Image translation ##
import cv2
import numpy as np
img=cv2.imread('E:/UOK/UOK_ Presentations/UOK_Digital Image
Processing/Python/Images/Fruits.jpeg')
[height, width] = img.shape[:2]
[tx, ty] = width / 4, height / 4 #Get tx and ty values for translation. You
can specify any value of your choice.
#Create the translation matrix using tx and ty, it is a NumPy array.
translation_matrix = np.array([
    [1, 0, tx],
    [0, 1, ty]
], dtype=np.float32)
translated_img = cv2.warpAffine(img, translation_matrix, (width, height))
#Apply the translation to the image using cv2.warpAffine(soriginal image,
translationmatrix, size of translated image(width, height))
cv2.imshow('Original image', img)
cv2.imshow('Translated image', translated_img)
cv2.waitKey(0)
####
```

نرمالسازی مقادیر ارزش پیکسل‌های تصویر:

```
## Normalizing the pixel values of an image ##
import cv2
import numpy as np
img=cv2.imread('E:/UOK/UOK_ Presentations/UOK_Digital Image
Processing/Python/Images/Fruits.jpeg')
norm = np.zeros((img.shape[0],img.shape[1])) # Creating an empty matrix
with the same size of original image
norm_img = cv2.normalize(img,norm,0,1,cv2.NORM_MINMAX,cv2.CV_32F) #
Normalizing the image between [0, 1]
(B, G, R) = img[20, 50] # Access the pixel located at Row(y)=20,
Column(x)=50 in the orginal image
print('Pixel at the desire location in the orginal image - Blue: {}, Green:
{}, Red: {}'.format(B, G, R))
(B, G, R) = norm_img[20, 50] # Access the pixel located at Row(y)=20,
Column(x)=50 in the normalized image
print('Pixel at the desire location in the normalized image - Blue: {},
Green: {}, Red: {}'.format(B, G, R))
####
```

تغییر کلاس داده تصویر:

```
## Convert the image format from unsigned int8 (uint8) to single precision
float and vice versa##
import cv2
import numpy as np
img=cv2.imread('E:/UOK/UOK_ Presentations/UOK_Digital Image
Processing/Python/Images/Fruits.jpeg')
print (img.dtype)
(B, G, R) = img[20, 50] # Access the pixel located at Row(y)=20,
Column(x)=50 in the orginal image
print('Pixel at the desire location in the orginal image - Blue: {}, Green:
{}, Red: {}'.format(B, G, R))
img_float = np.float32(img)# Convert to float32 (for double precision float
use float64)
print (img_float.dtype)
```

```
(B, G, R) = img_float[20, 50] # Access the pixel located at Row(y)=20,
Column(x)=50 in the float type image
print('Pixel at the desire location in the float image - Blue: {}, Green:
{}, Red: {}'.format(B, G, R))
img_uint8 = np.uint8(img_float)# Convert to uint8
print (img_uint8.dtype)
(B, G, R) = img_uint8[20, 50] # Access the pixel located at Row(y)=20,
Column(x)=50 in the uint8 type image
print('Pixel at the desire location in the uint8 image - Blue: {}, Green:
{}, Red: {}'.format(B, G, R))
####
```