Image Processing using OpenCV

@July 6, 2022

Documentation of Project:

OpenCV is a library of programming functions mainly aimed at real-time computer vision. It is widely used for image processing and analysis. In this week's video, I plan to go through the introduction of opencv in images, but going through simple functions of the documentation of the image processing techniques such as displaying, the color schemes, corners and edges and then finally conclude with a coordinates system. I will create time-lapses of the creation of the code that I have prepared yesterday and then explain each part individually.

Presentation:

5 minute video

Outline -

1. Opening an image using OpenCV
2. Color Space
3. Corner Point Detection
4. Coordinates on an Image

Script -

Let's start by opening a basic image using opencv in image

write first part of code and open image

Color Space are a way to represent the color channels present in the image that gives the image that particular hue. There are several different color spaces and each has its own significance. RGB is the default space, however, OpenCV actually stores color in the BGR format. Let's write the code to see how we can segregate the colors.

write code and show original image and then either blue/red/green

Now, let's try to mark the corners of significant figures on an image.

write code and show output

First, we read the image and can change it into a gray-scale image using the `cv2.cvtColor` function. Then, using the `goodFeaturesToTrack` , we can retrieve the corners and then round them using the numpy function `np.int0` . Then, we create a loop in order to iterate over all the points and draws a circle over it.

Now, let’s find the coordinates of the mouses cursor on the image using event attributes.

write code and show output and then explain code

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# 1. Displaying Image on the screen
# img = cv2.imread("cat.png", cv2.IMREAD_COLOR)
# cv2.imshow("Cat Image", img)
# cv2.waitKey(0)
# cv2.destroyAllWindows()

# 2. Color Space
# image = cv2.imread('RGB.png')
# B, G, R = cv2.split(image)
#
# cv2.imshow("original", image)
# cv2.waitKey(0)

# cv2.imshow("blue", B) # or G or R
# cv2.waitKey(0)

# 3. Corner Point Detection on Images
# img = cv2.imread('shapes.png')

# original image -> gray scale image
# gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# detect corners with the goodFeaturesToTrack
# corners = cv2.goodFeaturesToTrack(gray, 27, 0.01, 10)
# corners = np.int0.(corners)
#
# for i in corners:
#     x, y = i.ravel()
#     cv2.circle(img, (x, y), 2, 255, -1)
#
# plt.imshow(img), plt.show()

# 4. Coordinates of an image using event attributes
# def click_event(event, x, y, flags, params):
#     # checking for button click
#     if event == cv2.EVENT_LBUTTONDOWN:
#         # displaying the coordinates on the console
#         print('(',x, ', ', y,')')
#         # displaying the coordinates on the image
#         font = cv2.FONT_HERSHEY_SIMPLEX
#         fontSize = 1
#         coordinates = (x,y)
#         color = (0, 0, 255)
#         width = 2
#         cv2.putText(img, str(x) + ',' + str(y), coordinates, font, fontSize, color, width)
#         cv2.imshow('image', img)
#
# img = cv2.imread('cat.png')
# cv2.imshow('image', img)
# # setting mouse handler for the image and calling the click_event() function
# cv2.setMouseCallback('image', click_event)
# cv2.waitKey(0)
# cv2.destroyAllWindows()
```

Resources Used -

- https://www.geeksforgeeks.org/reading-image-opencv-using-python/
- https://www.geeksforgeeks.org/color-spaces-in-opencv-python/
- https://www.geeksforgeeks.org/python-detect-corner-of-an-image-using-opencv/

Code -

1. Opening file -

```
import cv2

# To read image from disk, we use cv2.imread function, in below method,
img = cv2.imread("cat.png", cv2.IMREAD_COLOR)

# Creating GUI window to display an image on screen first Parameter is windows title (should be in string format)
# Second Parameter is image array
cv2.imshow("image", img)

# To hold the window on screen, we use cv2.waitKey method
# Once it detected the close input, it will release the control
# To the next line
# First Parameter is for holding screen for specified milliseconds
# It should be positive integer. If 0 pass an parameter, then it will
# hold the screen until user close it.
cv2.waitKey(0)

# It is for removing/deleting created GUI window from screen
# and memory
cv2.destroyAllWindows()
```

2. Color Space -

```
# Color Space
image = cv2.imread('RGB.png')
B, G, R = cv2.split(image)

cv2.imshow("original", image)
cv2.waitKey(0)

# cv2.imshow("blue", B) # or G or R
# cv2.waitKey(0)
```

3. Corner Point Detection -

```
# Corner Point Detection on Images
# read the image
img = cv2.imread('shapes.png')

# convert image to gray scale image
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# detect corners with the goodFeaturesToTrack function.
corners = cv2.goodFeaturesToTrack(gray, 27, 0.01, 10)
corners = np.int0(corners)
```

```
# we iterate through each corner,
# making a circle at each point that we think is a corner.
for i in corners:
    x, y = i.ravel()
    cv2.circle(img, (x, y), 3, 255, -1)

plt.imshow(img), plt.show()
```

4. Coordinates -

```
def click_event(event, x, y, flags, params):
    # checking for left mouse clicks
    if event == cv2.EVENT_LBUTTONDOWN:
        # displaying the coordinates on the Shell
        print('(',x, ', ', y,')')

        # displaying the coordinates on the image window
        font = cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(img, str(x) + ',' + str(y), (x, y), font, 1, (255, 0, 0), 2)
        cv2.imshow('image', img)

    # checking for right mouse clicks
    if event == cv2.EVENT_RBUTTONDOWN:
        # displaying the coordinates on the Shell
        print('(',x, ', ', y,')')

        # displaying the coordinates on the image window
        font = cv2.FONT_HERSHEY_SIMPLEX
        b = img[y, x, 0]
        g = img[y, x, 1]
        r = img[y, x, 2]
        cv2.putText(img, str(b) + ',' + str(g) + ',' + str(r),(x, y), font, 1, (255, 255, 0), 2)
        cv2.imshow('image', img)

# driver function
if __name__ == "__main__":
    # reading the image
    img = cv2.imread('cat.png')

    # displaying the image
    cv2.imshow('image', img)

    # setting mouse handler for the image and calling the click_event() function
    cv2.setMouseCallback('image', click_event)

    # wait for a key to be pressed to exit
    cv2.waitKey(0)

    # close the window
    cv2.destroyAllWindows()
```

Final Code -

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# 1. Displaying Image on the screen
# img = cv2.imread("cat.png", cv2.IMREAD_COLOR)
```

```
# cv2.imshow("Cat Image", img)
# cv2.waitKey(0)
# cv2.destroyAllWindows()

# 2. Color Space
# image = cv2.imread('RGB.png')
# B, G, R = cv2.split(image)
#
# cv2.imshow("original", image)
# cv2.waitKey(0)

# cv2.imshow("blue", B) # or G or R
# cv2.waitKey(0)

# 3. Corner Point Detection on Images
# img = cv2.imread('shapes.png')

# original image -> gray scale image
# gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# detect corners with the goodFeaturesToTrack
# corners = cv2.goodFeaturesToTrack(gray, 27, 0.01, 10)
# corners = np.int0(corners)
#
# for i in corners:
#     x, y = i.ravel()
#     cv2.circle(img, (x, y), 2, 255, -1)
#
# plt.imshow(img), plt.show()

# 4. Coordinates of an image using event attributes
# def click_event(event, x, y, flags, params):
#     # checking for button click
#     if event == cv2.EVENT_LBUTTONDOWN:
#         # displaying the coordinates on the console
#         print('(',x, ', ', y,')')
#         # displaying the coordinates on the image
#         font = cv2.FONT_HERSHEY_SIMPLEX
#         fontSize = 1
#         coordinates = (x,y)
#         color = (0, 0, 255)
#         width = 2
#         cv2.putText(img, str(x) + ',' + str(y), coordinates, font, fontSize, color, width)
#         cv2.imshow('image', img)
#
# img = cv2.imread('cat.png')
# cv2.imshow('image', img)
# # setting mouse handler for the image and calling the click_event() function
# cv2.setMouseCallback('image', click_event)
# cv2.waitKey(0)
# cv2.destroyAllWindows()
```