به نام خدا

تمرین سری صفر- درس مبانی بینایی کامپیوتر

سید محمد علی نخاری- شماره دانشجویی: 99521496

- در حل سوالات این تمرین از سایت های مختلف جهت آشنایی با توابع مختلف کتابخانه پایتون استفاده شد. از جمله این سایت ها میتوان به numpy,geeks for geeks و ... اشاره کرد.

سوال اول) در ابتدا و در بخش import، کتابخانه های مورد نیاز در حل این سوال را اضافه کرده و آن را در بخش جدایی قرار میدهیم.

الف) در این قسمت عکس گفته شده را از درایو به وسیله کتابخانه opencv خوانده و با تابع cv2_imshow آن را نمایش میدهیم.(تمامی مسیر های مورد نیاز برای دسترسی به فایل های گفته شده از روی google drive برداشته شده است). حال در قسمت بعد سعی میکنیم عکس خوانده شده بوسیله کتابخانه opencv را با کتابخانه matplotlib نمایش دهیم اما همانطور که در خروجی مشاهده میشود بخش های آبی و قرمز عکس به هم ریخته و به درستی نشان داده نشده اند.
این موضوع به دلیل آن است که کتابخانه opencv کانال های عکس خوانده شده را به صورت BGR در نظر گرفته در حالیکه کتابخانه matplotlib عکس ها را به صورت RGB میخواند.
برای حل این مشکل کافی است که هنگام نمایش عکس خوانده شده توسط کتابخانه opencv، کانال های آن را معکوس کنیم.

ب) خروجی img.shape یک تاپل سه تایی میباشد که عضو اول آن نشان دهنده مقدار ارتفاع و مقدار دوم عرض عکس را نشان میدهد. عضو آخر آن هم تعداد کانال های آن عکس را نشان میدهد که در اینجا سه کانال قرمز، آبی و سبز داریم.

ج) در این قسمت در ابتدا با استفاده از کتابخانه os پوشه ای جدید در مسیر گفته شده میسازیم و سپس بر روی تمامی عکس های موجود پیمایش کرده و آن هارا با استفاده از کتابخانه opencv خوانده و با استفاده از همان کتابخانه آن هارا به عکس سیاه سفید تبدیل کرده و در پوشه ساخته

شده میریزیم. در نهایت یک فایل متنی به نام هر عکس ساخته و ابعاد عکس را در آن نگهداری میکنیم.

سوال دوم) در این سوال قرار است که از میان پنج عکس موجود هر کدام را به قسمت های 16*16 تبدیل کرده و از میان این قسمت های آن هایی را که همه پیکسل هایشان بزرگتر از صفر هستند را درون آرایه ای ذخیره کنیم. در آخر آرایه ای خواهیم داشت که شامل پنج عضو بوده و هر عضو یک آرایه 1100 عضوی از عکس های 16*16 میباشد.

در پایان به صورت رندوم از میان هرکدام از پنج عضو آرایه بالا چهار عکس را به صورت رندوم انتخاب کرده و همه را به هم متصل کرده و بوسیله کتابخانه cv2_imshow نمایش میدهیم.

پیاده سازی تابع resize: برای پیاده سازی این تابع از کتابخانه opencv استفاده میکنیم که در ورودی عکس اصلی و سایز مورد نظر را میدهیم و در خروجی، عکس تغییر سایز یافته را خروجی میگیریم.

پیاده سازی تابع crop: طبق همانچیزی که در بالا گفته شد برای کراپ کردن عکس ها به صورتی که 1100 عکس تولید شود به صورتی مربع های 16*16 را جدا میکنیم که همپوشانی نداشته باشند و دقیقا از کنار هم شروع شوند. شرط مهمی هم که باید چک کنیم این است که تمامی پیکسل های مربع انتخاب شده صفر نباشند.

در آخر برای اینکه خروجی ها را درون یک عکس نمایش داده شود از کتابخانه opencv و متد copyMakeBorder استفاده کرده شد.(توضیحات مربوط به این تابع از روی سایت اصلی opencv مطالعه شد).

خروجی نهایی به صورت زیر میباشد:

```
for i in range(5):
  temp_img = []
  for j in range(4):
    rand_inx = np.random.randint(1100)
    sel_img = cv2.copyMakeBorder(All_cropped_images[i][rand_i
    sel_img = resize(sel_img, (50, 50))
    temp_img.append(sel_img)
  res_images.append(temp_img)

final_img = cv2.vconcat([cv2.hconcat(i) for i in res_images])
cv2_imshow(final_img)
```

سوال سوم)

الف) برای تولید یک ماتریس n*n که اعضای آن را اعداد رندوم تشکیل میدهند از تابع random.randint کتابخانه numpy استفاده میکنیم. به این صورت که در ورودی این تابع به ترتیب شروع تولید عدد رندوم، پایان عدد رندوم، سایز ماتریس تولید شده و در نهایت نوع متغیر های آن آرایه میباشد. پس از ساخته شدن ماتریس رندوم آن را چاپ کرده و بوسیله تابع count_digit تعداد اراقام صفر تا نه موجود در این ماتریس را محاسبه کرده و در خروجی چاپ میکنیم.

- ماتریس 6*6 که توسط اعداد رندوم در بازه 6 تا 106 تولید شده است:

```
Enter the size of the matrix: 6
31 19 60 101 85 84
32 66 84 52 13 84
40 68 67 55 34 36
72 80 91 23 63 19
76 66 23 101 105 55
90 27 84 8 95 102
```

- محاسبه تعداد تکرار ارقام صفر تا نه در ماتریس تولید شده:

```
##############################################

{0: 4, 1: 3, 2: 0, 3: 1, 4: 2, 5: 1, 6: 3, 7: 4, 8: 1, 9: 0}
```

ب) برای پیمایش به صورت گفته شده از تابع بازگشتی استفاده کردم. نحوه پیمایش را به این صورت در نظر گرفتم که ما دو حرکت اصلی داریم: یک حرکت به صورت مورب و رو به پایین که تا جایی که مقدار ایندکس y برابر با صفر نشود و یا مقدار ایندکس x با n-1 برابر نشود ادامه پیدا میکند و مقدار y افزایش و مقدار x را کاهش میدهد. حرکت دیگر هم مورب و رو به بالا میباشد. این روند را آنقدر ادامه میدهیم که تمامی اعضای آرایه پیمایش شده باشند.

سوال چهارم) هدف از این سوال آشنایی بیشتر با توابع محاسباتی موجود در کتابخانه numpy بود.

الف) در این قسمت خواسته شده است که عملیات گفته شده بر روی ماتریس های داده شده را محاسبه کنیم. در ابتدا ماتریس [[[1,2,3],[4,5,6],[7,8,9]]] را با استفاده از تابع transpose موجود در کتابخانه numpy محاسبه کرده و حاصل را در همان ماتریس ابتدایی ضرب کنیم که آن هم بوسیله تابع dot قابل انجام است. به همین ترتیب عملیات گفته شده را انجام میدهیم تا به جواب برسیم. نکته مهم در حل این سوال جمع یک ماتریس یک در سه با ماتریسی سه در سه است که در حالت عادی و روی این کاغذ این اتفاق ممکن نیست، در حالیکه در کتابخانه numpy و پایتون مفهمومی به نام broadcasting وجود دارد که این موضوع را مدیریت میکند و ماتریس با سایز یک در سه را به صورت که ماتریس سه در سه در نظر میگیرد که تمامی سطر های آن با هم یکی هستند.

```
array([[0.796875, 1.796875, 2.796875],
       [4.      , 5.      , 6.      ],
       [7.203125, 8.203125, 9.203125]])
```

ب) در این قسمت خواسته شده بود که با توابع پایه ای پایتون ماتریس سه در سه A را بر روی مانریس پنج در پنج B بلغزانیم. منظور این است که از ارایه پنج در پنج به صورت متوالی خانه های سه در سه جدا کرده و درایه به درایه در ماتریس A ضرب و حاصل آن هارا جمع کرده و درون یک درایه از ماتریس نهایی بریزیم.

نتیجه نهایی به صورت زیر خواهد بود:

```
A = np.array([[1,1,1], [1,-9,1], [1,1,1]], np.int32)
B = np.array([[1,1,1,1,1], [1,2,2,2,1], [1,2,2,2,1], [1,2,2,2,1], [1,1,1,1,1]], np.int32)
print(window_sliding(A, B))
```

```
[[-7 -5 -7]
 [-5 -2 -5]
 [-7 -5 -7]]
```

سوال پنجم) در ابتدا خواسته شده است که اطلاعات پایه ای عکس گفته شده در فایل تمرین را خروجی دهیم که تمامی این اطلاعات بوسیله توابع کتابخانه های opencv, numpy قابل حل است.
اما قبل از اینکه اطلاعات خواسته شده را خروجی دهیم میبایست عکس را خوانده و درون متغیری بریزیم. برای خواندن عکس از کتابخانه opencv استفاده کردم که همانطور در سوال اول توضیح دادم این خروجی به صورت BGR خواهد بود، پس کافی است با توابع موجود در همین کتابخانه opencv عکس BGR را به فرمت RGB تبدیل نماییم.
کد نوشته شده برای این قسمت به صورت زیر میباشد:

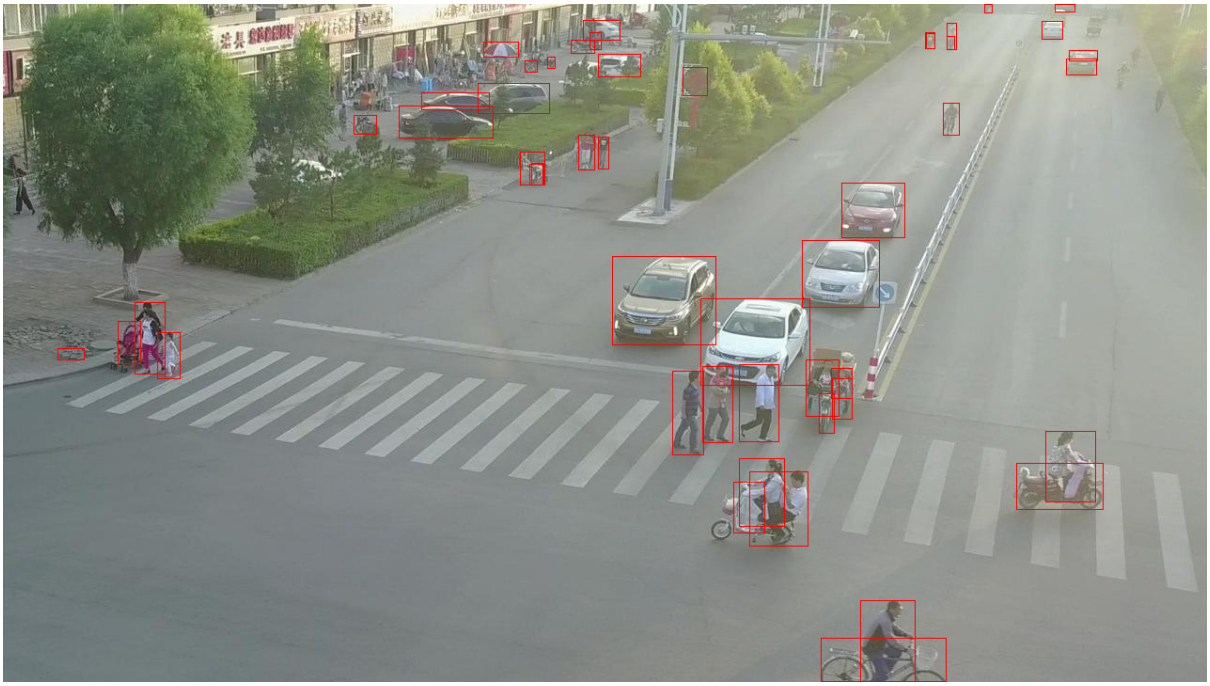
```
path = '/content/drive/MyDrive/ColabNotebooks/FCV/HW0/Q5.png'
# read image with cv2 in 'BGR' mode
image = cv2.imread(path)
# conv the 'BGR' image to 'RGB'
rgb_img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

print(f'heights: {rgb_img.shape[0]} pixels, width: {rgb_img.shape[1]} pixels')
print(f'type of value: {rgb_img.dtype}')
print(f'average the complete image: {np.average(rgb_img)}\naverage the image over 3 channel: {np.average(rgb_img, axis=(0,1))}')
# red channel
print(f'The minimum of red channel: {rgb_img[:,:,0].min()}, The maximum of red channel: {rgb_img[:,:,0].max()}')
# green channel
print(f'The minimum of green channel: {rgb_img[:,:,1].min()}, The maximum of green channel: {rgb_img[:,:,1].max()}')
# blue channel
print(f'The minimum of blue channel: {rgb_img[:,:,2].min()}, The maximum of blue channel: {rgb_img[:,:,2].max()}')

# find min and mux in image
print(f'The smallest pixel: {rgb_img.min(axis=(0,1,2))}, The largest pixel: {rgb_img.max(axis=(0,1,2))}')
```

خروجی هم به صورت زیر خواهد شد:

```
heights: 721 pixels, width: 1281 pixels
type of value: uint8
average the complete image: 139.98719468688319
average the image over 3 channel: [139.17862475 144.50184225 136.28111706]
The minimum of red channel: 10, The maximum of red channel: 255
The minimum of green channel: 10, The maximum of green channel: 255
The minimum of blue channel: 0, The maximum of blue channel: 255
The smallest pixel: 0, The largest pixel: 255
```

در ادامه با استفاده از قطعه کد نوشته شده مدل های موجود در عکس را شناسایی کرده و درون متغیر detections ریخته که در ادامه با استفاده از کتابخانه opencv و تابع rectangle دور مدل های شناسایی شده موجود در عکس مستطیل کشیده و خروجی عکس جدید را با عنوان گفته شده در مسیر جاری ذخیره کنیم.

خروجی نهایی عکس زیر خواهد بود:



برای ذخیره عکس نهایی در مسیر دلخواه از تابع imwrite کتابخانه opencv استفاده میکنیم، ورودی این تابع مسیر ذخیره فایل به همراه عکس گرفته و آن را در مسیر گفته شده نگهداری میکند.