به نام خدا

درس مبانی بینایی کامپیوتر

تمرین سری نهم

مدرس درس:
جناب آقای دکتر محمدی

تهیه شده توسط:
الناز رضایی ۹۸۴۱۱۳۸۷

تاریخ ارسال: ۱۴۰۱/۰۹/۱۹

سوال ۱ :

عملگر سایش و افزایش را با توجه به عنصر ساختاری زیر بر روی تصویر مورد نظر اعمال کنید (در صورت نیاز از reflect padding استفاده نمایید). (۱۵ نمره)

input

10	10	10	20	10	10	20	10
10	20	20	20	20	20	20	10
10	10	10	20	10	20	20	10
10	10	20	30	10	20	30	10
10	10	30	10	10	30	10	20
20	10	30	10	30	20	20	10
20	20	20	20	20	10	20	10
20	20	30	20	10	30	10	30

kernel

1	1	1
1	0	0
1	0	0

پاسخ ۱:

قبل از انجام هر یک از عملیات dilation یا erosion، مطابق صورت سوال، باید reflect padding انجام دهیم. سپس هر یک از عملگرهای dilation و erosion را روی ورودی جدید خود اعمال می‌کنیم.

برای عملگر dilation، باید کرنل داده شده را روی ورودی بلغرانیم و در هر ۹ خانه‌ای که کرنل روی آن قرار می‌گیرد، بین خانه‌هایی که کرنل در آن‌ها یک است، در تصویر ورودی خود ماکزیمم می‌گیریم. یعنی در واقع از رابطه زیر استفاده می‌شود:

$$(f \oplus D)(x) = max\{f(z) : z \in D_x\}$$

برای عملگر erosion نیز مشابه dilation عمل می‌کنیم، تنها تفاوت این است که در erosion در تصویر، مینیم جاهایی که کرنل در آن ۱ است را در نظر می‌گیریم. طبق رابطه زیر داریم:

$$(f \ominus D)(x) = min\{f(z) : z \in D_x\}$$

حال با توجه به توضیحات بالا، تصویر را پس از dilation و erosion به دست می‌آوریم.

- padding: reflect

input

10	10	10	10	20	10	10	20	10	10
10	10	10	10	20	10	10	20	10	10
10	10	20	20	20	20	20	20	10	10
10	10	10	10	20	10	20	20	10	10
10	10	10	20	30	10	20	30	10	10
10	10	10	30	10	10	30	10	20	20
20	20	10	30	10	30	20	20	10	10
20	20	20	20	20	20	20	10	20	10
20	20	20	30	20	10	30	10	30	30
20	20	20	30	20	10	30	10	30	30

- dilation: برای اعمال عملگر گسترش، باید ابتدا kernel را ۱۸۰ درجه چرخانده و سپس عملیات گفته شده در بالا را انجام دهیم.

kernel

0	0	1
0	0	1
1	1	1

output

20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	30	30	30	30	30	30
10	30	30	30	30	30	30	20
20	30	30	30	30	30	20	20
20	30	20	30	30	20	20	20
20	30	30	30	30	30	30	30
20	30	30	30	30	30	30	30

- erosion:

output

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	20	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
20	20	20	20	10	10	10	10

kernel

1	1	1
1	0	0
1	0	0

من برای اطمینان از درستی جواب‌ها، کد مربوط به این سوال را نیز پیاده‌سازی کردم و به فایل پاسخ، ضمیمه شده است. همچنین در تصویر زیر جواب‌های به دست آمده از کد را ملاحظه می‌کنید.

Erosion

```
erosion = cv2.erode(input, kernel, borderType=cv2.BORDER_REFLECT)
print(erosion)
```

✓ 0.7s

```
[[10 10 10 10 10 10 10 10]
 [10 10 10 10 10 10 10 10]
 [10 10 10 10 20 10 10 10]
 [10 10 10 10 10 10 10 10]
 [10 10 10 10 10 10 10 10]
 [10 10 10 10 10 10 10 10]
 [10 10 10 10 10 10 10 10]
 [20 20 20 20 10 10 10 10]]
```

Dilation

```
dilation = cv2.dilate(input, kernel_prime, borderType=cv2.BORDER_REFLECT)
print(dilation)
```

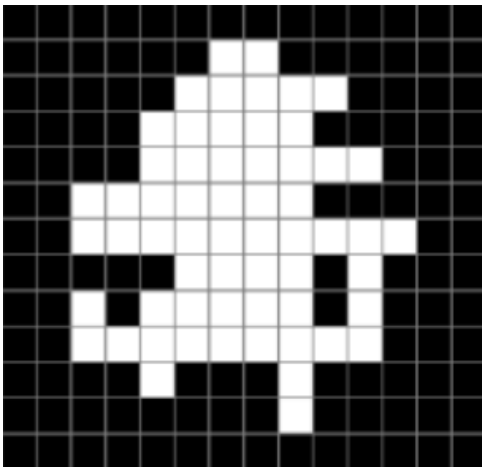
✓ 0.4s

```
[[20 20 20 20 20 20 20 20]
 [20 20 20 20 20 20 20 20]
 [20 20 30 30 30 30 30 30]
 [10 30 30 30 30 30 30 20]
 [20 30 30 30 30 30 20 20]
 [20 30 20 30 30 20 20 20]
 [20 30 30 30 30 30 30 30]
 [20 30 30 30 30 30 30 30]]
```

همانطور که در تصاویر بالا مشاهده می‌کنید، پاسخمان با پاسخ ارائه شده در بالا یکی می‌باشد.

سوال ۲:

برای استخراج مرز تصویر زیر ابتدا عنصرهای ساختاری مناسب را معرفی نمایید و سپس با استفاده از عملگر hit-or-miss خروجی مورد نظر را بدست آورید. (۲۰ نمره)



پاسخ ۲:

برای استخراج مرز، باید از عنصرهای ساختاری زیر استفاده کنیم.

B_1

0	0	0
0	-1	1
0	0	0

B_2

0	0	0
1	-1	0
0	0	0

B_3

0	0	0
0	-1	0
0	1	0

B_4

0	1	0
0	-1	0
0	0	0

حال به توضیح مختصری در مورد عنصرهای معرفی شده می‌پردازیم. عنصر ساختاری اول، الگوهایی را معرفی می‌کند که در سمت راست یک پیکسل سیاه مرکزی، یک پیکسل سفید قرار دارد؛ به عبارت دیگر این عنصر، مرزهای سمت چپی تصویر را تشخیص می‌دهد.

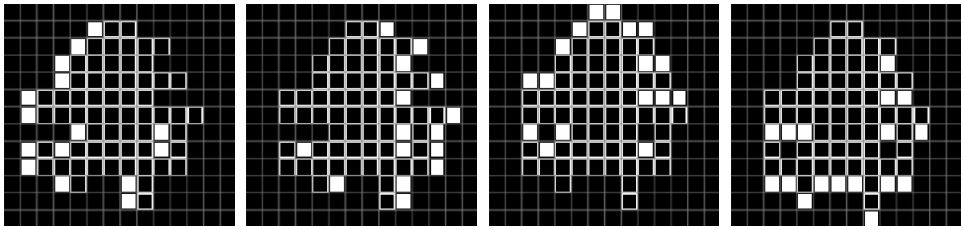
عنصر B_2، الگوهایی که در آن سمت چپ یک پیکسل سیاه مرکزی، یک پیکسل سفید قرار دارد را معرفی می‌کند؛ یعنی این عنصر نیز مرزهای سمت راستی تصویر را به دست می‌آورد.

عنصر ساختاری B_3 نیز الگوهای حاوی یک پیکسل سیاه مرکزی که در بالای آن یک پیکسل سفید قرار دارد را مشخص می‌کند. به بیان دیگر این عنصر مرز بالایی تصویر را محاسبه می‌کند.

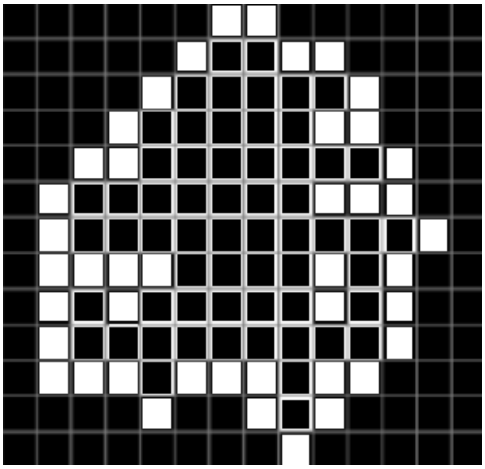
در انتها عنصر ساختاری B_4 را داریم که پیکسل سفید در پایین پیکسل مرکزی سیاه قرار دارد و مرزهای پایینی را تشخیص می‌دهد.

در تصویر زیر، شکل به دست آمده از هر عنصر ساختاری را مشاهده می‌کنید.

B_1 B_2 B_3 B_4



نتیجه حاصل از اعمال تمامی عنصرهای ساختاری، شکل زیر می‌شود که مرزهای خارجی را مشخص می‌کند. اگر می‌خواستیم مرزهای داخلی را نمایان کنیم، کافی بود تا جای ۱ و ۱ـ ها را عوض کنیم.



سوال ۳:

الف) بدون استفاده از توابع آماده کتابخانه‌ای و با استفاده از دانش مورفولوژی اسکلت تصاویر img[1-4].jpg را استخراج کنید. (۱۵ نمره)

ب) با ذخیره مراحل استخراج اسکلت، تصاویر اولیه را از روی اسکلت بازسازی کنید. (۲۰ نمره)

پاسخ ۳:

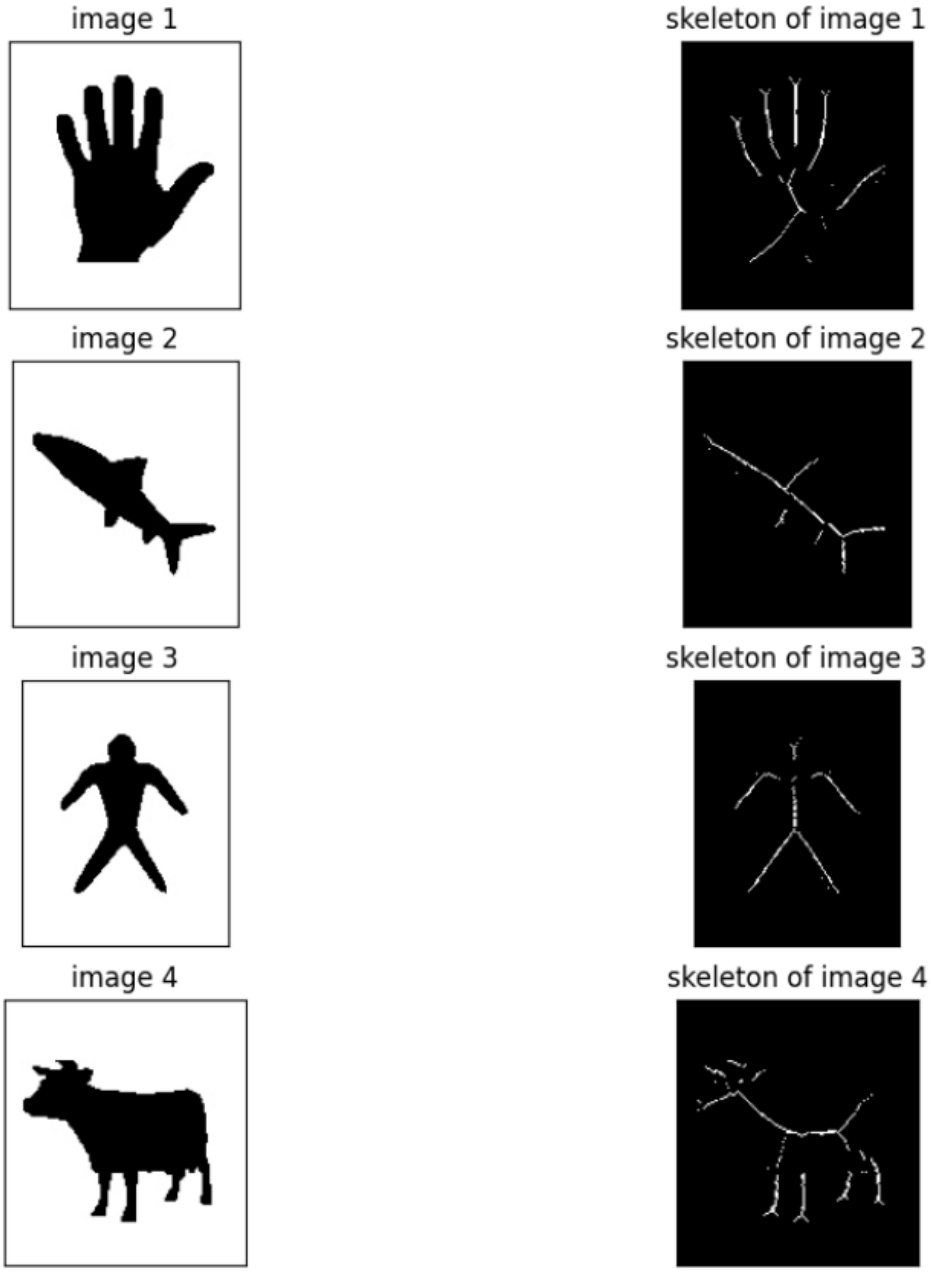
الف) می‌دانیم که برای استخراج اسکلت یک تصویر، باید مراحل زیر را پیاده‌سازی کنیم.

1. $S(A) = \bigcup_{k=0}^{k} S_k(A)$
2. $S_k(A) = (A \ominus kB) - (A \ominus kB)\ o\ B$
3. $A \ominus kB = (((A \ominus B) \ominus B) \ominus ...)$
4. $k = max\{k|(A \ominus kB) \neq \phi\}$

بنابراین با استفاده از قطعه کد زیر، ابتدا تصویر را با یک آستانه گذاری به باینری تبدیل کرده و سپس با استفاده از یک erosion و dilation عملگر open را روی تصویر اعمال می‌کنیم و سپس با تفریق تصویر اصلی از تصویر حاصل پس از عملیات open، S_k را به دست می‌آوریم. S_k و k را که نشان دهنده تعداد دفعات تکرار این عملیات است را در params می‌ریزیم تا از آن برای بازسازی تصویر استفاده کنیم.

```python
def get_skeleton(image):
    """
    Finds the skeleton of the input image.
    Parameters:
        image (numpy.ndarray): The input image.
    Returns:
        numpy.ndarray: The skeleton image.
        numpy.ndarray: The parameters required for reconstructing image
    """
    res = image.copy()
    params = []
    #Write your code here
    k = 0
    _, image = cv2.threshold(image, 100, 255, cv2.THRESH_BINARY_INV)
    res = np.zeros(image.shape, np.uint8)
    element = cv2.getStructuringElement(cv2.MORPH_CROSS, (3,3))
    while cv2.countNonZero(image)!=0:
        erosion = cv2.erode(image, element)
        open = cv2.dilate(erosion, element)
        subtract = cv2.subtract(image, open)
        res = cv2.bitwise_or(res, subtract)
        image = erosion
        params.append((k, subtract))
        k += 1
    return res, params
```

نتیجه این بخش که اسکلت تصاویر داده شده را نمایش می‌دهد، به شرح زیر است:



ب) در این بخش، از رابطه زیر استفاده می‌شود. نتایج پس از اعمال این رابطه، در ادامه آمده است.

$$A = \bigcup_{k=0}^{k} S_k(A) \oplus kB$$

skeleton of image 1

image 1

skeleton of image 2

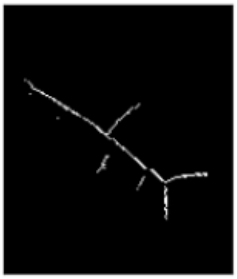


image 2

skeleton of image 3

image 3

skeleton of image 4

image 4

^

در عکس زیر نیز، کدی که برای این بخش زده شده، آورده شده است. در اینجا ابتدا عنصر ساختاری را مانند بخش قبل معرفی کرده و سپس به ازای هر k و S_k که از مرحله قبل به دست آوردیم، عملیات dilation را انجام می‌دهیم و در انتها پیکسل‌های سیاه و سفید را جابه‌جا می‌کنیم.

```python
def recons_skeleton(image, params):
    """
    Finds the original image from the skeleton.

    Parameters:
        image (numpy.ndarray): The skeleton of image.
        params (numpy.ndarray): The parameters required for reconstructing image

    Returns:
        numpy.ndarray: The original image.
    """

    res = image.copy()

    #Write your code here
    element = cv2.getStructuringElement(cv2.MORPH_CROSS,(3,3))
    for k, param in params:
        dilate = cv2.dilate(param ,element, iterations=k)
        res = cv2.bitwise_or(res, dilate)

    res = cv2.bitwise_not(res)
    return res
```

✓ 0.2s

سوال ۴:

با استفاده از دانش مورفولوژی و به کمک لینک۱ و لینک۲ مراحل زیر را انجام دهید. (در این تمرین شما مجاز به استفاده از توابع کتابخانه‌ای می‌باشید)

الف) برنامه‌ای بنویسید که تعداد اتومبیل‌های موجود در تصویر img5.jpg را بدست آورد. (۱۵ نمره)

نکته: این تصویر را با روش‌های موجود در درس به تصویر باینری تبدیل کنید و سپس با استفاده از عملگرهای مورفولوژی خطوط اضافی را حذف کنید و سپس با توابع کمکی اتومبیل‌ها را یافته و تعداد آن‌ها را گزارش کنید.

ب) برنامه‌ای بنویسید که تعداد گل‌های آفتابگردان موجود در تصویر img6.jpg را محاسبه کند. (۱۵ نمره)

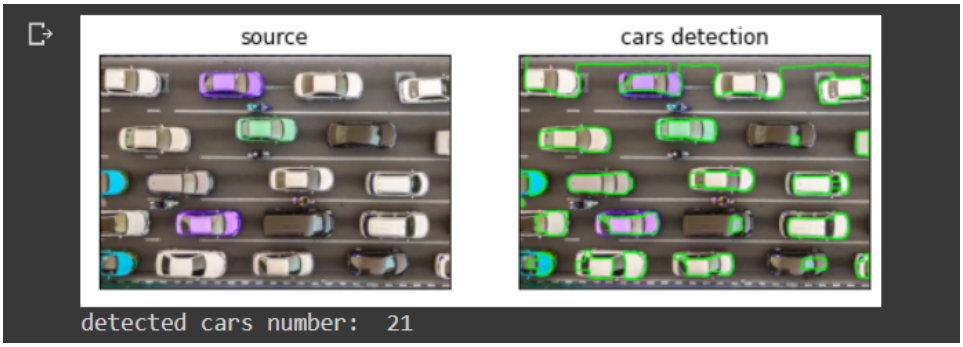
نکته: در این بخش باید با حالت رنگی تصویر کار کنید و تصویر را باینری نکنید. برای بدست آوردن تعداد گل‌ها می‌توانید بر روی بخش دایروی گل کار کنید و آن‌ها را بیابید تا تعداد گل‌ها را بدست آورید.

پاسخ ۴:

الف) با استفاده از قطعه کد زیر، ابتدا تصویر خود را باینری کرده و سپس یک بار عملگر open را می‌زنیم تا جزئیات تصویر، حذف شوند. سپس دوباره یک‌بار عملگر open و سپس close را به صورت متوالی اعمال می‌کنیم تا تصاویر متقطع حاصل از مرحله قبل، یکنواخت شوند.

```python
def detect_car_num(image):
    '''
    Detects shapes and number of cars in the input image.

    Parameters:
        image (numpy.ndarray): The input image.

    Returns:
        numpy.ndarray: The result image.
        integer: number of cars
    '''
    result = image.copy()
    cars_num = 0

    #Write your code here
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (5, 5), 0)
    tresh = cv2.threshold(blur, 120, 255, cv2.THRESH_BINARY)[1]

    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (10,10))
    open = cv2.morphologyEx(tresh, cv2.MORPH_OPEN, kernel)

    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5,5))
    open = cv2.morphologyEx(open, cv2.MORPH_OPEN, kernel)
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (15,15))
    close = cv2.morphologyEx(open, cv2.MORPH_CLOSE, kernel)

    cnts = cv2.findContours(close, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)

    for c in cnts:
      cars_num += 1
      cv2.drawContours(result, [c], -1, (0, 255, 0), 2)
    return result, cars_num
```

سپس با استفاده از findContours، ماشین‌ها را پیدا می‌کنیم. در نهایت، در حلقه for تعداد ماشین‌ها را شمرده و دور آن یک خط می‌کشیم. تصویر زیر خروجی حاصل از این مرحله نشان داده شده است.



ب) برای این بخش نیز ابتدا تصویر خود را به حالت grayscale تبدیل کرده تا بتوانیم از تبدیل hough برای پیدا کردن دایره‌ها که مرکز گل‌ها هستند، استفاده کنیم. همچنین با استفاده از تابع blur، نویزهای تصویر را حذف می‌کنیم. سپس با استفاده از دانشی که از مورفولوژی داشتیم، تصویر را erode کرده تا پیدا کردن دایره‌ها آسان‌تر شود. در ادامه دایره‌ها را با تبدیل hough پیدا کرده و در حلقه for، در انتها این تابع تعداد گل‌ها و تصویری که در آن گل‌ها مشخص هستند را return می‌کند.هم تعداد آن‌ها را شمرده و هم دایره‌ای به دور گل‌ها می‌کشیم تا مشخص شوند. کدی که برای این بخش زده شد، به شکل زیر می‌باشد.

```python
def detect_flower_num(image):
    '''
    Detects shapes and number of flowers in the input image.

    Parameters:
        image (numpy.ndarray): The input image.

    Returns:
        numpy.ndarray: The result image.
        integer: number of flowers
    '''

    result = image.copy()
    flowers_num = 0

    #Write your code here
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    blur = cv2.blur(gray, (3, 3))
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (10,10))
    erosion = cv2.erode(blur, kernel)
    circles = cv2.HoughCircles(erosion, cv2.HOUGH_GRADIENT, 1, 20, param1 = 40, param2 = 30, minRadius = 1, maxRadius = 45)
    circles = np.uint16(np.around(circles))

    for circle in circles[0, :]:
      flowers_num += 1
      x, y, r = circle[0], circle[1], circle[2]
      cv2.circle(result, (x, y), r, (0, 255, 0), 3)

    return result, flowers_num
```

نتیجه نهایی این قسمت، به شرح زیر است.

نکته: در حل این سوال، از لینک زیر کمک گرفته شد.

https://www.geeksforgeeks.org/circle-detection-using-opencv-python/