Computer vision

# Homework 9

Parsa Eissazadeh 97412364

---

**سوال 1**

در مرحله اول باید عکس را باینری کنیم . بنابراین هیتسوگرام آن را رسم میکنیم :

10 : 29 تا

20 : ۲۶ تا

30 : ۹ تا

حداکثر سطح روشنایی را ۳۰ در نظر میگیریم . به احتمال زیاد حد آستانه بین ۱۰ و ۲۰ است .

{محاسبات }

در این حالت $W_1$ می شود 29 و $W_2$ می شود 35 . واریانس گروه اول که همه 10 هستند صفر است و برای محاسبه ی  واریانس گروه دوم ابتدا نیاز داریم میانگین را حساب کنیم .

$$( 20 * 26 + 9 * 30) / 35 = 22.57$$

واریانس :

$$26 * (20 - 22.57)^2 + 9 * (30 - 22.57)^2 = 668.57$$

حالت بعدی این است که حد آستانه بین 20 و 30 قرار بگیرد که در آن صورت بخش دوم واریانس صفر دارد و w اش اهمیتی نخواهد داشت . و گروه اول ( 10 ها و 20 ها ) w اشان می شود 55 .

میانگین :

$$( 29*10 + 26*20 ) / 55 = 14.72$$

واریانس :

$$26 * (20 - 14.72)^2 + 29 * (10 - 14.72)^2 = 1370$$

در نتیجه حالت اول حالت بهتری بود برای حد آستانه و خانه های 10 را 0 و باقی خانه ها را 1 در نظر میگیریم .

---

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

**سایش**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

**افزایش**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

## سوال 2

وقتی از عملگر hit-or-miss استفاده می کنیم ، تنها در صورت وجود شکل عنصر ساختاری در عکس آنجا را سفید میکنیم و نه لزوما در زمان زیرمجموعه بودن .

گوشه ها بخشی از مرز ها هستند در نتیجه از فیلتر هایی که برای گوشه ها استفاده میکنیم اینجا هم استفاده میکنیم .

| | | |
|---|---|---|
| 0 | -1 | -1 |
| 1 | 1 | -1 |
| 0 | 1 | 0 |

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | -1 |
| 0 | -1 | -1 |

| | | |
|---|---|---|
| 0 | 1 | 0 |
| -1 | 1 | 1 |
| -1 | -1 | 0 |

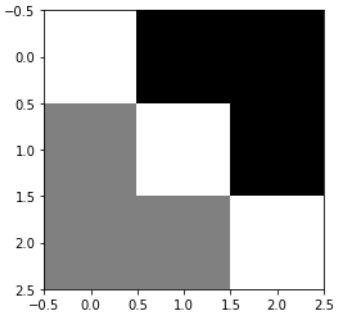
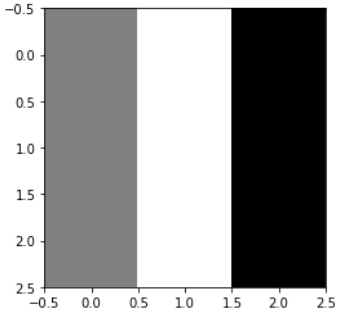
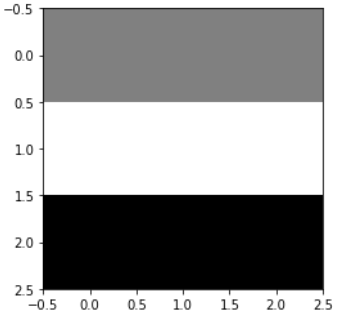
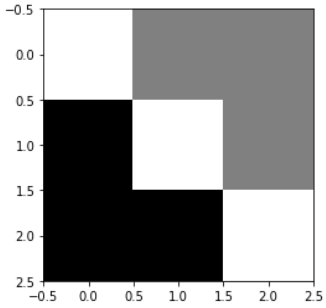
| | | |
|---|---|---|
| -1 | -1 | 0 |
| -1 | 1 | 1 |
| 0 | 1 | 0 |

همچنین بقیه جاهای مرز به شکل خطوطی صاف را اریب هستند . تعداد فیلتر های ما زیاد هستند ، دسته بندی :

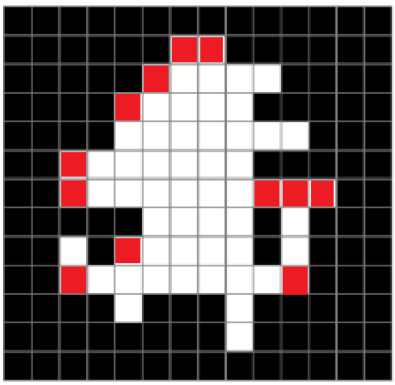
- اریب
  - به سمت چپ
    - پایین خط -1
    - بالای خط -1

- به سمت راست
  - پایین خط -1
  - بالای خط -1
- صاف
  - عمودی
    - راست خط -1
    - چپ خط -1
  - افقی
    - پایین خط -1
    - بالا خط -1

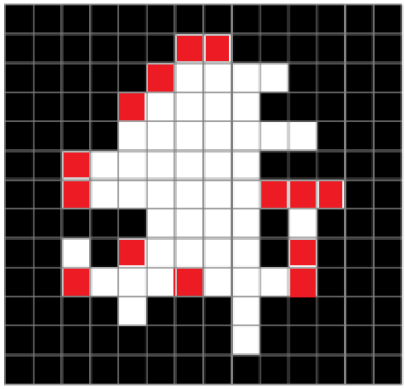
در مجموع 8 فیلتر داریم ، بعضی از آنها :

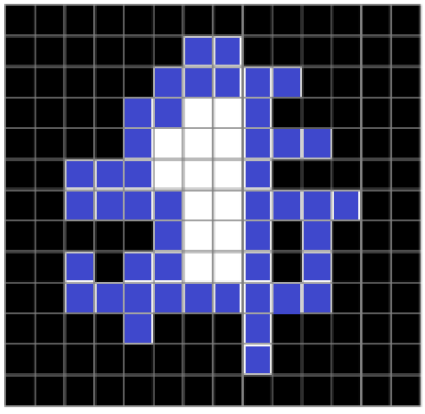
نتیجه اعمال فیلتر های گوشه یاب :



بعد از اعمال فیتلر های لبه یاب به شکل رو به رو میرسیم که مشخص است مرز ها را به خوبی پیدا نکرده است.



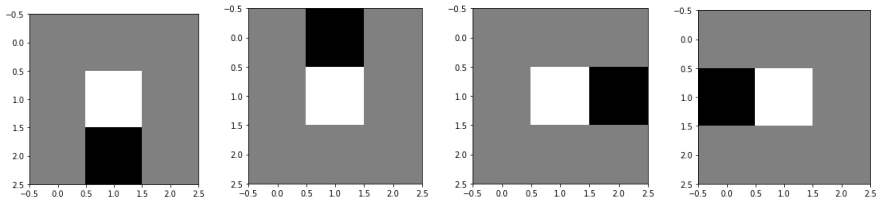
مرزی که مد نظر ما است بدین شکل است :



پس باید فیلتر هایی طراحی کنیم که این ها را جدا کند :

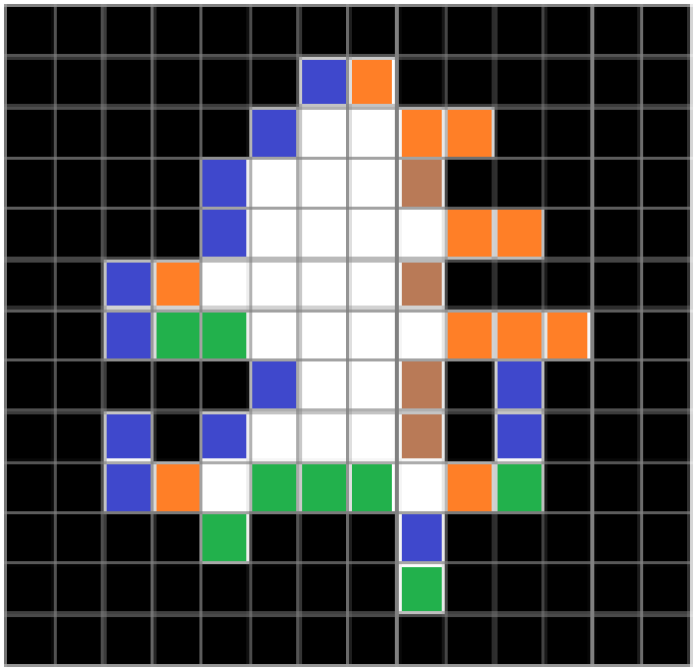
فیلتر های گوشه یاب کماکان درست کار میکنند .

فیلتر های دیگر به شکل زیر است :



در این چهار جهت و چهار جهت قطری .

نتیجه اعمال این فیلتر با hit-or-miss :



**آبی :** وسط سفید ، چپ سیاه don't care

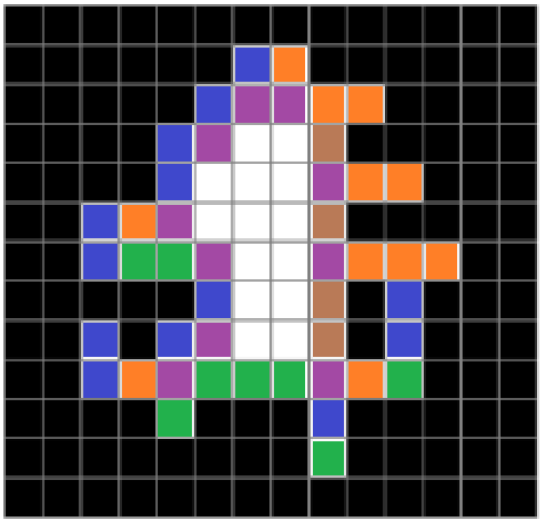
**نارنجی :** وسط سفید ، بالا سیاه don't care

**قهوه ای :** وسط سفید ، راست سیاه don't care

**سبز :** وسط سفید ، پایین سیاه ، بقیه don't care

در نتیجه همین 4 فیلتر کفایت میکنند و نیازی به فیلتر های گوشه یاب نیست .

بعد از اعمال فیلتر هایی با جهات قطبی (رنگ بنفش ):



البته بدون اعمال فیلتر های قطبی هم به مرز خوبی رسیده بودیم .

### سوال 3

قبل از پیاده سازی متد استخراج اسکلت به چند متد احتیاج داریم :

- سایش
- گسترش
- عملگر باز

طبق فرمول سایش ، زمانی یک پیکسل انتخاب می شود که عنصر ساختاری زیرمجموعه خانه های همسایه اش باشد . این یعنی اگر convolve بگیریم و در عنصر ساختاریمان مثلا 5 تا 1 داشته باشیم ، مقدار جدید یک پیکسل باید بشود 5 . کد :

```
[44] def erosion(img , str_elem):
       c = signal.convolve2d(str_elem , img)

       height = img.shape[0]
       width = img.shape[1]

       return np.where(c>=5 , 1 , 0)[1:height + 1 , 1:width + 1]
```

همچنین برای گسترش ، اگر خروجی کانولوشن از 0 بیشتر شود یعنی حداقل یک عضو مشترک داشتیم و اشتراک تهی نیست و انتخاب می شود .

```
[49] def dilation(img , str_elem):
       c = signal.convolve2d(str_elem , img)

       height = img.shape[0]
       width = img.shape[1]

       return np.where(c>0 , 1 , 0)[1:height + 1 , 1:width + 1]
```

عملگر باز هم متشکل است از یک سایش و یک افزایش پشت سر هم :

```
[60] def opening(img , str_elem):
       erode = erosion(img , str_elem)
       return dilation(erode , str_elem)
```

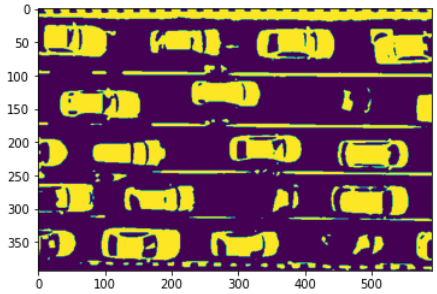
- خروجی همه متد ها در notebook تست شده است .

## سوال 4

**الف )**

```
image = cv2.imread("/content/img5.jpg")
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (7, 7), 0)
thresh = cv2.threshold(blurred, 130, 255, cv2.THRESH_BINARY)[1]

plt.imshow(thresh)
```

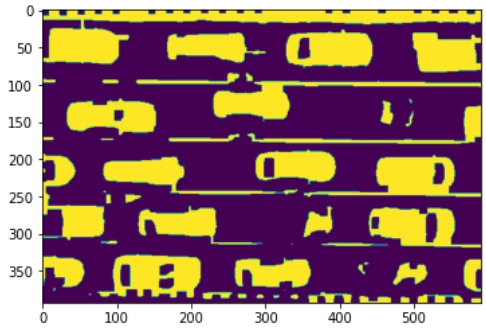
```
<matplotlib.image.AxesImage at 0x7f899f11a0a0>
```



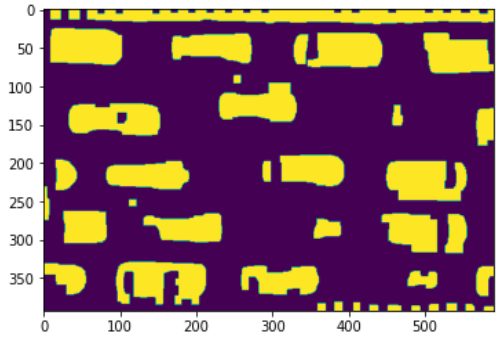

در ابتدا عکس را بلور میکنیم و اتسو را اعمال میکنیم .

در مرحله اول می خواستم حفره ها را کوچکتر کنم برای همین عملگر Close را زدم سپس عملگر باز را تا خطوط اضافی را حذف کنم . چون پنجره های ماشین ها آن ها را به چند شکل تقسیم میکرد .

حاصل عملگر Close :




حاصل عملگر باز :




برای استفاده از عملگر های باز و بسته از لینک زیر استفاده کردم :

https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html

همانطور که مشاهده می شود خط های جدا کننده خیابان محو شده اند .

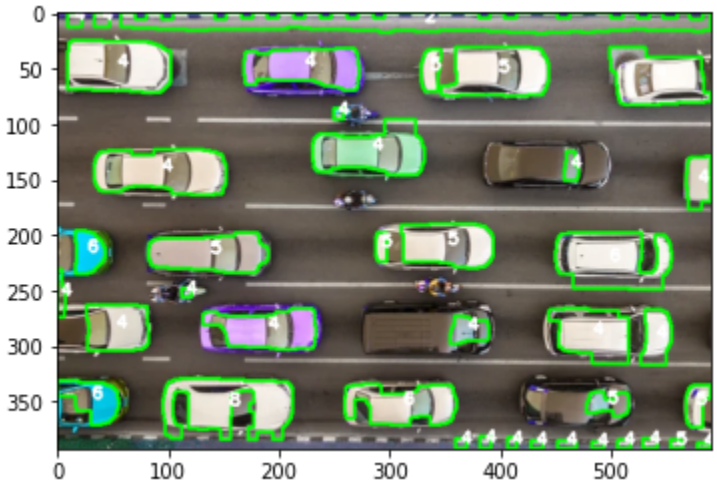
سپس به کمک لینک اولی که فرستاده شده بود ، یک shape یابی به روی عکس اعمال کردم . کد :

```
[28] import imutils

     cnts = cv2.findContours(opened.copy(), cv2.RETR_EXTERNAL,
       cv2.CHAIN_APPROX_SIMPLE)
     cnts = imutils.grab_contours(cnts)
```

```python
def detect(c):
  # initialize the shape name and approximate the contour
  shape = "unidentified"
  peri = cv2.arcLength(c, True)
  approx = cv2.approxPolyDP(c, 0.04 * peri, True)
  return str(len(approx))
```

```python
for c in cnts :
  M = cv2.moments(c)
  cX = int(M["m10"] / M["m00"])
  cY = int(M["m01"] / M["m00"])

  shape = detect(c)

  cv2.drawContours(image, [c], -1, (0, 255, 0), 2)
  cv2.putText(image, shape, (cX, cY), cv2.FONT_HERSHEY_SIMPLEX,
    0.5, (255, 255, 255), 2)

  plt.imshow(image)
```

نتیجه :

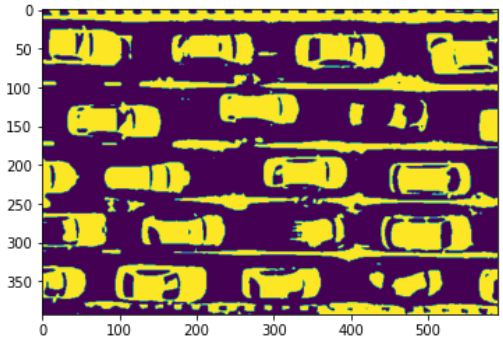



در این عکس ، اشکال معلوم شده اند و تعداد اضلاعشان نیز نمایش داده شده است . ولی بسیاری از موانع کنار خیابان هم به شکل چهارضلعی نمایش داده شده اند که در نتیجه نهایی ما مشکل ایجاد خواهند کرد .

ماشین های سیاه یکی از مشکلات بودند چون به خاطر سطح روشنایی پایینی که دارند اختلافشان با آسفالت قابل تشخیص نبود . برای همین یک clahe اعمال کردم روی عکس .

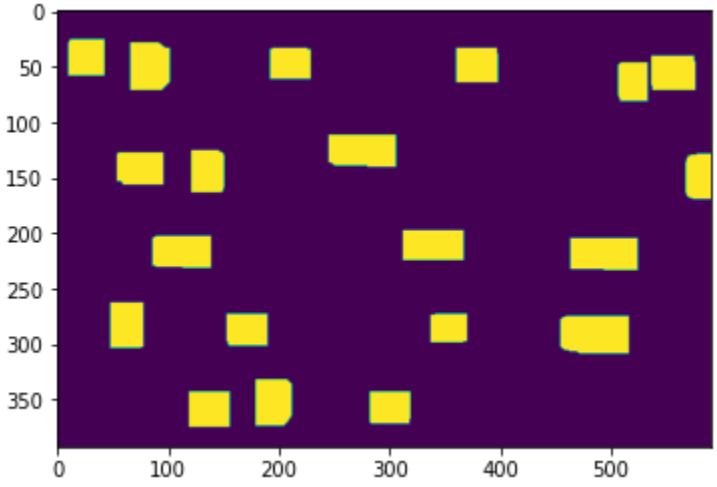
```
[46] clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
     gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
     clh = clahe.apply(gray)
     plt.imshow(clh)
```

نتیجه باینری کردن عکس بهتر شد ولی همچنان مشکل حل نشد .

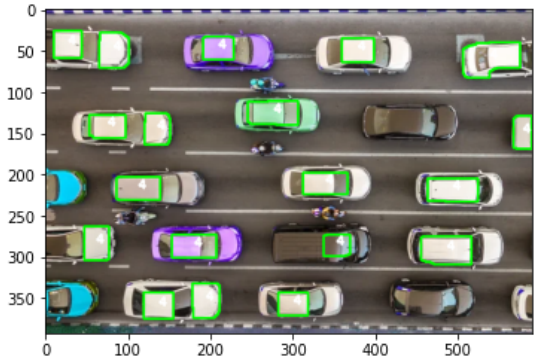



از آنجایی که ماشین ها بزرگترین اشکال موجود در عکس هستند ما می توانیم از فیلترای بزرگ برای گسترش استفاده کنیم ، به قدری بزرگ که همه چیز جز ماشین ها حذف شوند . برای همین اندازه فیلتر ها را به 25

افزایش دادم . نتیجه بدین صورت شد :




موانع و موتور ها و خطوط خیابان همگی پاک شده اند و این نزدیک ترین نتیجه است :




در این مثال از متدی که به کمک آن اشکال را تشخیص میدادیم استفاده ای نشد .

**ب )**

ابتدا در سه کانال میخوانیم عکس را :


ما به دنبال این هستیم که دایره های وسط گل ها باقی بمانند و بقیه تقریبا حذف شوند .

وسط عکس بسیار تیره است در نتیجه clahe را بر روی عکس اعمال میکنیم :

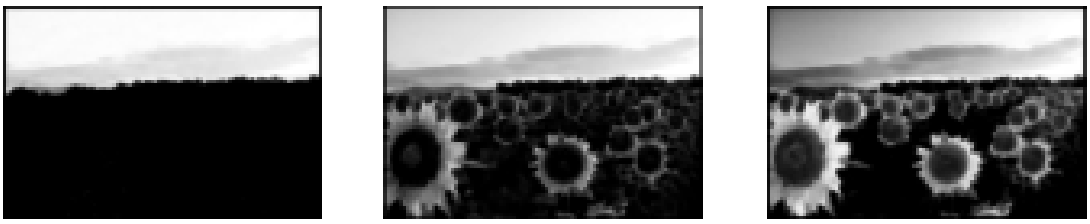
نتیجه خیلی بهتر شد به گونه ای که در کانال سبز در وسط پیکسل های روشن هم داریم .

در ابتدا فرمول کاهش را پیاده سازی میکنیم . فرمول کاهش همانند max pooling است فقط با این تفاوت که minimum را پیدا میکنیم .
کانال آبی به ما اطلاعات خاصی نمی دهد در نتیجه میتوان از آن صرف نظر کرد .

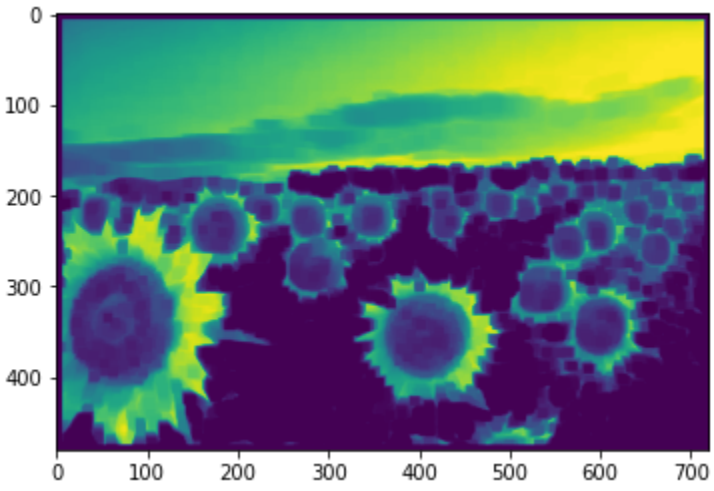
برای پیاده سازی از فیلتر minimum_filter از کتابخانه ndimage استفاده کردیم . کد :

```
kernel_size = 10

channels_dilated = []

for ch in channels :

  ascent = misc.ascent()
  res = ndimage.minimum_filter(ch , size=kernel_size , mode='constant')

  channels_dilated.append(res)

show_channels(channels_dilated)
```

و نتیجه به این صورت شد :



در کانال قرمز به شکل زیر رسیدیم :




که تقریبا بهترین خروجی است و بر اساس همین میتوان تعداد گل ها را شمرد .
حال باید متد findcontours که در بخش قبل استفاده شد را اعمال کنیم .

```
for c in cnts :
  M = cv2.moments(c)
  cX = int(M["m10"] / M["m00"])
  cY = int(M["m01"] / M["m00"])

  shape = detect(c)

  cv2.drawContours(im, [c], -1, (0, 255, 0), 2)
  cv2.putText(im, shape, (cX, cY), cv2.FONT_HERSHEY_SIMPLEX,
    0.5, (255, 255, 255), 2)

  plt.imshow(im)
```

اما برای این کانال قرمز که بیشترین اطلاعات را دارد به ارور زیر برخوردیم :

```
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-61-a3fb78ebb454> in <module>
      3 for c in cnts :
      4   M = cv2.moments(c)
----> 5   cX = int(M["m10"] / M["m00"])
      6   cY = int(M["m01"] / M["m00"])
      7

ZeroDivisionError: float division by zero
```