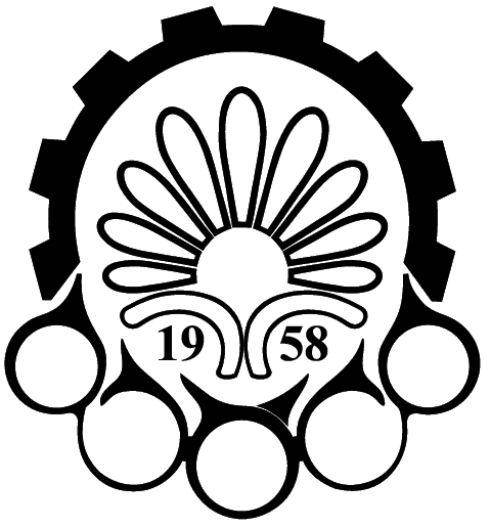


شبکه های پردازش زبان طبیعی

دانشجو:

پویا شریفی

استاد:

دکتر شریفیان

سوال ۱)

این گزارش یک پیاده‌سازی کد پایتون برای تشخیص و استخراج محیط با رنگ سبز از یک تصویر با استفاده از کتابخانه OpenCV را ارائه می‌دهد. این کد تصویر ورودی را پردازش می‌کند، آستانه‌بندی رنگی را در فضای رنگی HSV اعمال می‌کند، عملیات مورفولوژی برای بهبود ماسک‌ها را انجام می‌دهد و در نهایت اشیاء تشخیص داده شده را شناسایی و باکس‌های محدودکننده را اطراف آنها رسم می‌کند.

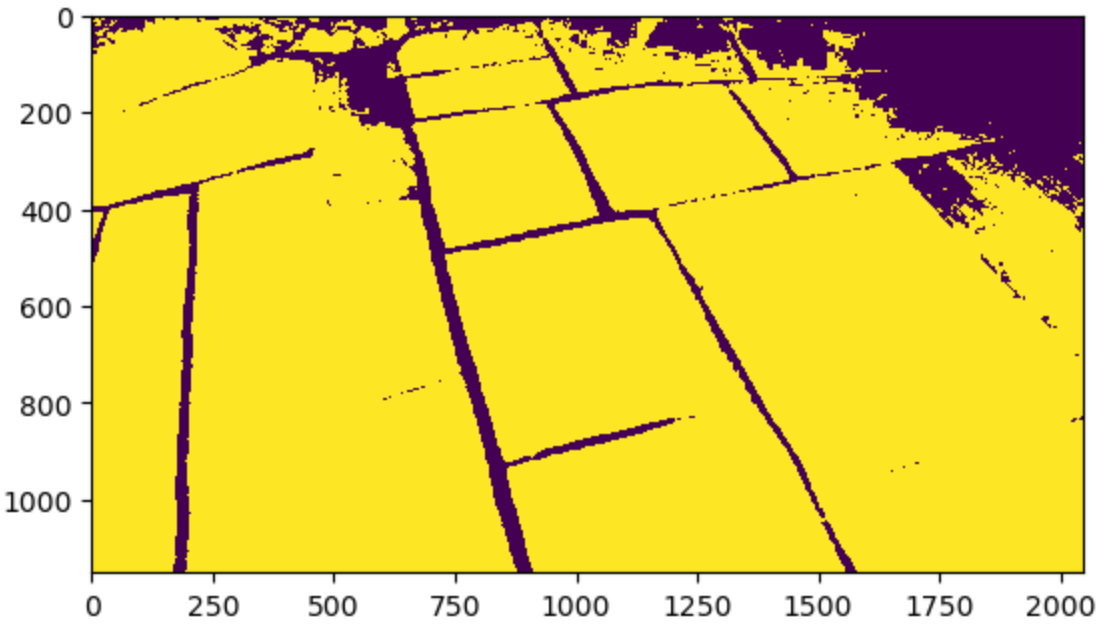
توضیحات کد:

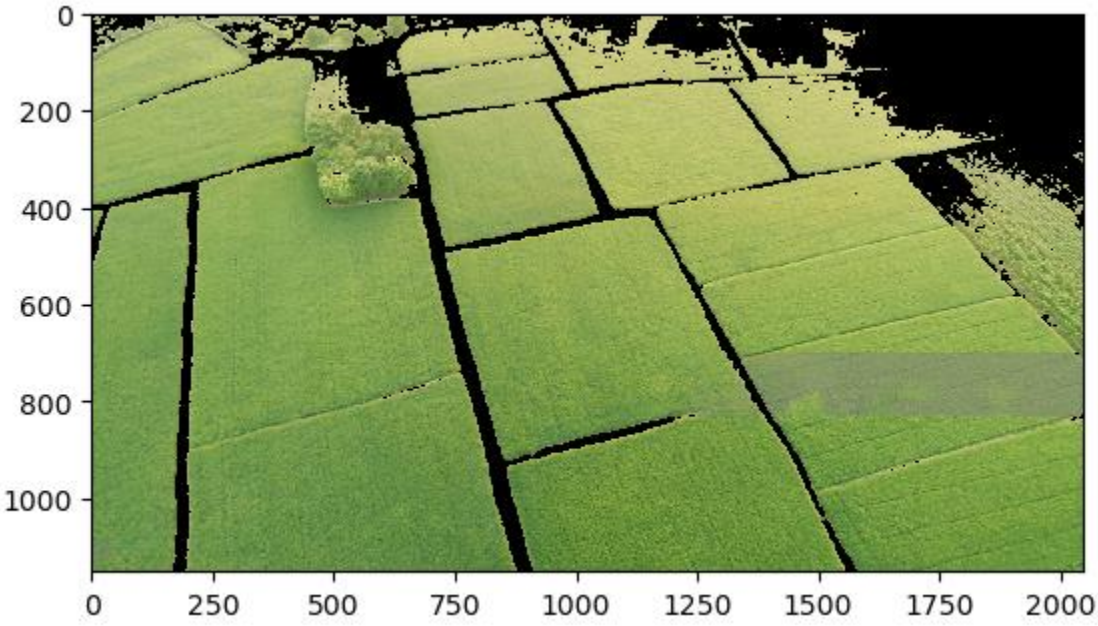
```
sv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

lower_green = np.array([30, 0, 0])
upper_green = np.array([90, 255, 255])
```

از روی hue رنگ های سبز در اینترنت می بینیم که رنگ سبز 90 به بالا است,پس آن را برای upper green قرار میدهیم.

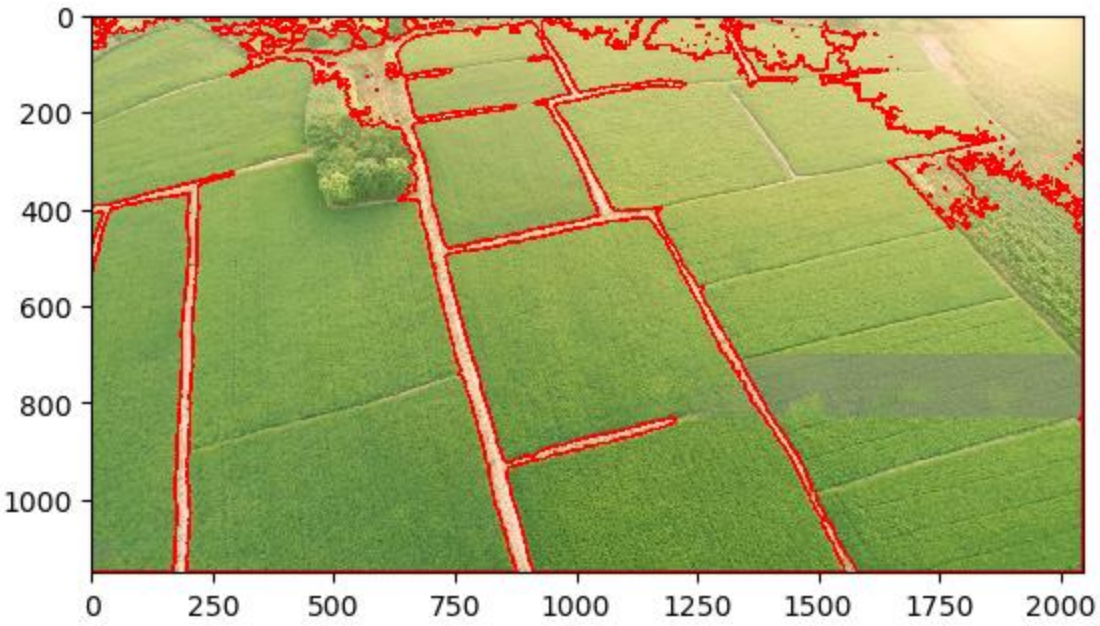
عملیات مورفولوژی: به منظور بهبود ماسک‌ها و حذف نویز، عملیات مورفولوژی از قبیل توسعه و فرسایش با استفاده از توابع cv2.dilate() و cv2.erode() اعمال می‌شود.

سپس پس از درست کردن ماسک آن را در عکس ضرب می کنیم تا فیلترمان را ببینیم.





حالا کانتور ها را می کشیم.



سپس برای پیدا کردن مساحت از کد زیر استفاده می کنیم:

```
areas = []
for c in contours1:
    areas.append(cv2.contourArea(c))

areas.sort(reverse=True)
print(areas)
```

و خروجی زیر را می گیریم:

```
[1249017.5, 641203.5, 138709.0, 890.0, 224.0, 218.0, 198.5,
    189.5, 114.5, 90.5, 84.0, 76.0, 57.5, 57.0, 52.0, 50.0,
 48.5, 47.5, 37.5, 36.5, 36.0, 35.0, 32.0, 31.0, 29.5, 28.5,
 26.5, 26.5, 22.0, 18.5, 14.5, 14.0, 13.5, 13.0, 12.0, 11.0,
10.0, 9.0, 8.5, 8.0, 7.5, 7.5, 7.0, 6.5, 6.5, 6.5, 6.0, 5.5,
 5.0, 4.5, 4.5, 4.5, 4.5, 4.5, 4.0, 4.0, 3.5, 3.5, 3.0, 3.0,
 3.0, 3.0, 3.0, 2.5, 2.5, 2.5, 2.5, 2.0, 2.0, 1.5, 1.5, 1.5,
```

```
1.5, 1.5, 1.5, 1.5, 1.5, 1.0, 1.0, 1.0, 1.0, 1.0, 0.5, 0.5,
0.5, 0.5, 0.5, 0.5, 0.5,[
```

سوال۲)

ابتدا `template` ها را استخراج میکنیم:

مقدمه:

این گزارش یک پیاده‌سازی کد پایتون برای تشخیص اشیاء با استفاده از هم‌نشانگر قالب و کتابخانه OpenCV را ارائه می‌دهد. این کد تصویر اصلی و تصویر قالب را بارگیری می‌کند، آنها را به فضای رنگ خاکستری تبدیل می‌کند، هم‌نشانگر قالب را اعمال کرده، آستانه‌ای را برای حذف مطابقت‌های ضعیف تنظیم می‌کند، سرکوب بیشینه غیرمتقاطع را برای حذف جعبه‌های محدود کننده تداخلی اعمال می‌کند و مستطیل‌هایی را در اطراف اشیاء تشخیص‌داده شده بر روی تصویر اصلی می‌کشد.

توضیحات کد:

بارگیری تصاویر: کد از تابع `cv2.imread`() برای بارگیری تصویر اصلی و تصویر قالب استفاده می‌کند.

تبدیل به خاکستری: هر دو تصویر اصلی و تصویر قالب به خاکستری با استفاده از تابع `cv2.cvtColor`() تبدیل می‌شوند.

هم‌نشانگر قالب: هم‌نشانگر قالب با استفاده از تابع `cv2.matchTemplate`() اجرا می‌شود. این تابع تصویر قالب را با نسخه خاکستری تصویر اصلی مقایسه کرده و مطابقت‌ها را پیدا می‌کند.

تنظیم آستانه: آستانه‌ای تنظیم می‌شود تا مطابقت‌های ضعیف را حذف کند. تابع np.where() برای پیدا کردن مکان‌هایی که مقدار مطابقت بیشتر از آستانه استفاده می‌شود.

یافتن بهترین مطابقت‌ها: تابع cv2.minMaxLoc() برای یافتن حداقل و حداکثر مقادیر مطابقت و مکان‌های متناظر آنها استفاده می‌شود. بیشینه مقدار مطابقت مطابقت بهترین را نشان می‌دهد.

سرکوب بیشینه غیرمتقاطع: برای حذف جعبه‌های محدود کننده تداخلی، تابع سرکوب بیشینه غیرمتقاطع NMS() پیاده‌سازی می‌شود. این تابع جعبه‌های محدود کننده تشخیص داده شده را به عنوان ورودی دریافت کرده و جعبه‌های تکراری را براساس آستانه همپوشانی مشخص شده حذف می‌کند. تابع مقدار همپوشانی هر جعبه را با سایر جعبه‌ها مقایسه کرده و فقط کسانی را که بالاترین امتیاز را دارند نگه می‌دارد.

کشیدن مستطیل‌ها: کد با استفاده از تابع cv2.rectangle() مستطیل‌هایی را در اطراف اشیاء تشخیص‌داده شده بر روی تصویر اصلی می‌کشد. مختصات بالا-چپ و پایین-راست مستطیل‌ها بر اساس شکل قالب و موقعیت بهترین مطابقت تعیین می‌شود.

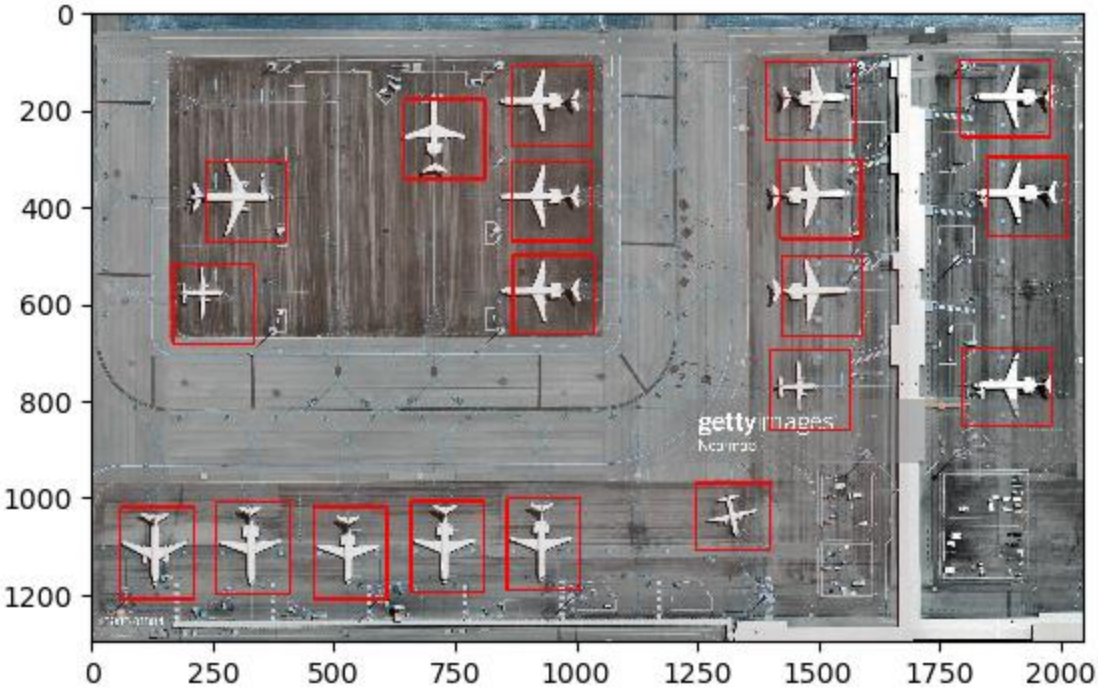
نمایش نتیجه: تصویر حاصل با اشیاء تشخیص‌داده شده با استفاده از تابع cv2.imshow() نمایش داده می‌شود. برنامه برای بستن پنجره و پایان اجرا منتظر فشردن یک کلید می‌ماند.

نتیجه‌گیری:

کد ارائه شده فرایند تشخیص اشیاء با استفاده از هم‌نشانگر قالب در OpenCV را نشان می‌دهد. با توجه به تصاویر و آستانه‌های مشخص شده، مستطیل‌هایی را در اطراف اشیاء تشخیص‌داده شده بر روی تصویر اصلی می‌کشد. همچنین تابع سرکوب بیشینه غیرمتقاطع پیاده‌سازی شده، تداخل جعبه‌های محدود کننده را حذف می‌کند تا نتایج دقیق‌تری به دست آید.

ضمیمه: تابع سرکوب بیشینه غیرمتقاطع

```
def NMS(boxes, overlapThresh = 0.4):
    #return an empty list, if no boxes given
    if len(boxes) == 0:
        return []
    x1 = boxes[:, 0]  # x coordinate of the top-left corner
    y1 = boxes[:, 1]  # y coordinate of the top-left corner
    x2 = boxes[:, 2]  # x coordinate of the bottom-right corner
    y2 = boxes[:, 3]  # y coordinate of the bottom-right corner
    # compute the area of the bounding boxes and sort the bounding
    # boxes by the bottom-right y-coordinate of the bounding box
    areas = (x2 - x1 + 1) * (y2 - y1 + 1) # We have a least a box of one pixel,
therefore the +1
    indices = np.arange(len(x1))
    for i,box in enumerate(boxes):
        temp_indices = indices[indices!=i]
        xx1 = np.maximum(box[0], boxes[temp_indices,0])
        yy1 = np.maximum(box[1], boxes[temp_indices,1])
        xx2 = np.minimum(box[2], boxes[temp_indices,2])
        yy2 = np.minimum(box[3], boxes[temp_indices,3])
        w = np.maximum(0, xx2 - xx1 + 1)
        h = np.maximum(0, yy2 - yy1 + 1)
        # compute the ratio of overlap
        overlap = (w * h) / areas[temp_indices]
        if np.any(overlap) > treshold:
            indices = indices[indices != i]
    return boxes[indices].astype(int)
```

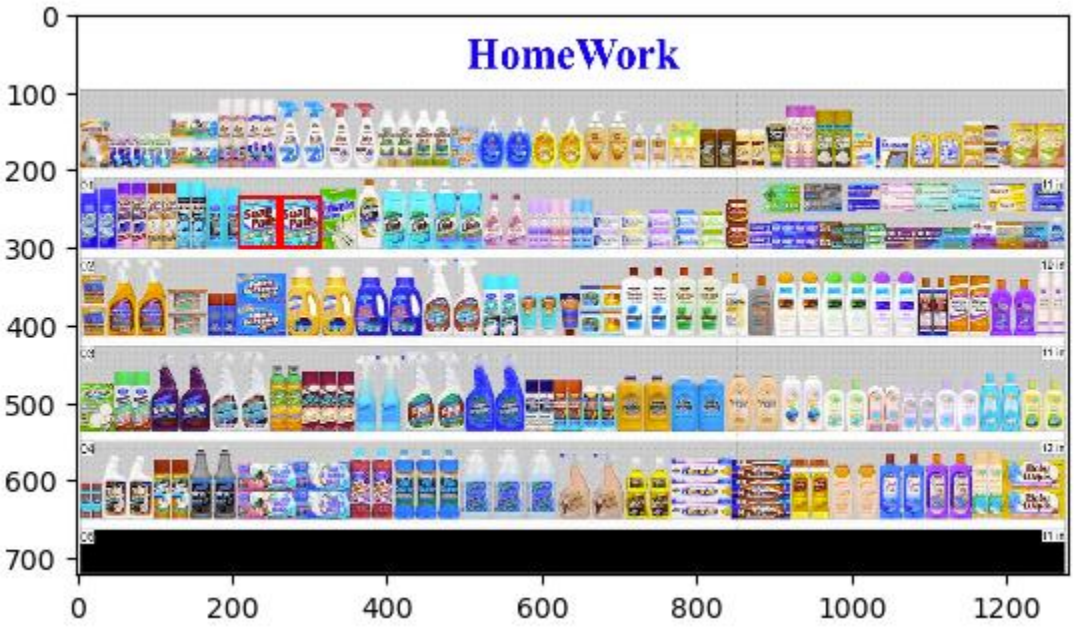


همانطور که می بینید همه ی هواپیما ها تشخیص داده شدند.

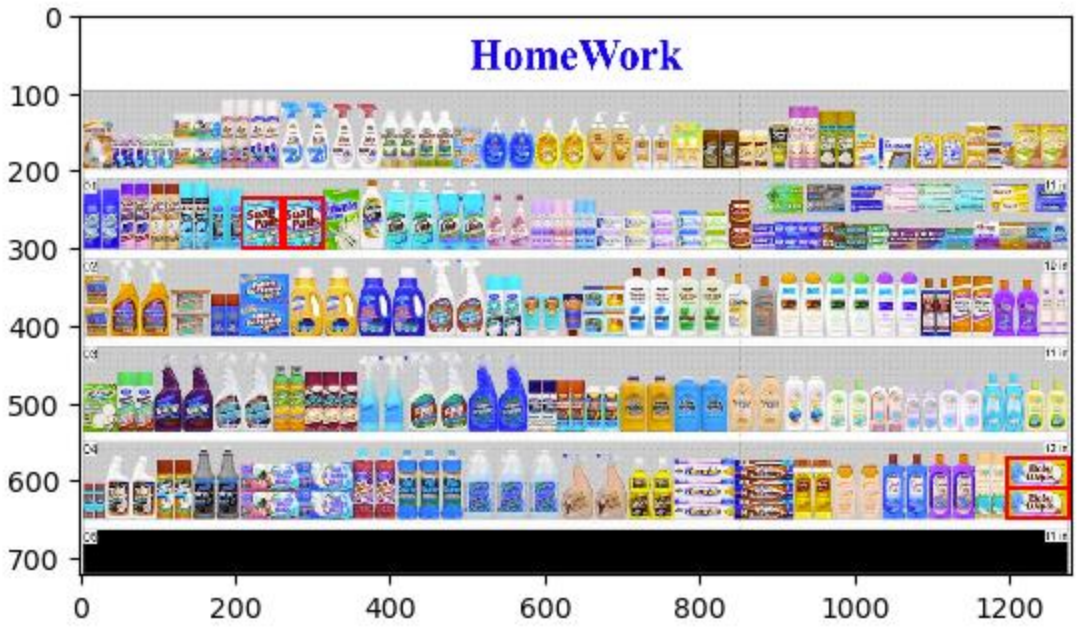
سوال۳)

این سوال نیز مانند سوال قبلی عمل میکنیم,ابتدا تمپلت ها یا قالب ها را جدا میکنیم.

حالا مثل قبل `template matching` را انجام میدهیم.



number of products 2



number of products 2