

: Contour moment and area

به تعداد پیکسل هایی که درون مرز یک شکل بسته قرار دارند مساحت کانتور گفته می شود. همچنین مرکز هندسی یک کانتور نقطه ای است که شکل را از نظر توزیع پیکسل ها متعادل می کند. یعنی اگر شکل را یک صفحه ی نازک فلزی در نظر بگیریم، مرکز جرم جایی است که آن شکل را می توان روی نوک یک سوزن بدون افتادن نگه داشت.

برای یافتن مرکز هندسی از مفهومی به نام **ممان های هندسی (Moments)** استفاده می شود.

در OpenCV با تابع `cv2.moments()` ممان های مختلفی از یک کانتور محاسبه می شوند. این ممان ها اعدادی هستند که ویژگی های آماری توزیع شکل را بیان می کنند.

سه ممان مهم برای پیدا کردن مرکز کانتور:

- `m00` مساحت کانتور
- `m10` مجموع x های وزن دار
- `m01` مجموع y های وزن دار


برای درک اینکه `m10` و همچنین `m01` چیست و منظور از "مجموع x های وزن دار" چیست، باید کمی به مفاهیم ریاضی پشت ممان ها (Moments) نگاه کنیم:

تعریف ساده ی "مجموع x های وزن دار (`m10`)"

وقتی می گوییم:

$$m10 = \text{مجموع مختصات } x \text{ پیکسل ها، با در نظر گرفتن شدت آن ها (یا وزن آن ها)}$$

یعنی در اصل داریم به این نگاه می کنیم که چه پیکسل هایی در کجای تصویر قرار دارند، و چه سهمی در شکل دارند.

مثال خیلی ساده (تصور ذهنی): 

فرض کن یک تصویر باینری داریم (یعنی فقط ۰ و ۲۵۵)، و یک شکل (مثلاً دایره) داریم که شامل چند پیکسل سفید (255) است. فرض کن پیکسل‌های سفید مختصات زیر را دارند:

(1, 2), (2, 2), (3, 2)

در اینجا:

- مختصات x ها: 1, 2, 3
- شدت هر پیکسل: 255 (چون سفید است)

حالا:

✓ یعنی: m10

$$m10 = (1 * 255) + (2 * 255) + (3 * 255) = 255 * (1 + 2 + 3) = 255 * 6 = 1530$$

یعنی مجموع مختصات x، با وزن هر پیکسل (که در تصویر باینری معمولاً برابر با 255 یا 1 است).

✓ چیست؟ m00

m00 = مجموع وزن‌ها = مجموع شدت پیکسل‌های داخل کانتور

در این مثال:

$$m00 = 255 + 255 + 255 = 765$$

✓ مرکز هندسی: (Centroid)

مرکز: x:

$$cx = m10 / m00 = 1530 / 765 = 2.0$$

که دقیقاً وسط نقاط 1، 2، 3 هست.

✳ چرا می‌گوییم "وزن دار"؟

چون مختصات هر پیکسل (مثلاً $x=3$) ممکن است نسبت به شدت آن پیکسل اهمیت بیشتری داشته باشد. مثلاً اگر یک پیکسل روشن‌تر (مثلاً در تصویر Grayscale) باشد، در مرکز ثقل وزن بیشتری دارد.

فرمول محاسبه مرکز:

$$\begin{aligned} cx &= m10 / m00 \\ cy &= m01 / m00 \end{aligned}$$

⚠ نکته مهم:

اگر $m00$ برابر صفر باشد (یعنی کانتور خیلی کوچک یا ناصحیح باشد)، تقسیم بر صفر اتفاق می‌افتد. باید اول بررسی کنیم که $m00 \neq 0$.

همچنین در صورتی که بخواهیم مساحت کانتورها را در یک عکس به دست بیاوریم می‌توانیم از متد `cv2.contourArea()` استفاده کنیم. این مساحت می‌تواند معیار خوبی برای تشخیص بزرگی یا کوچکی اشیاء در تصویر باشد.

بیاید این دو عملیات را بر روی عکس بالون‌ها انجام دهیم و نتیجه هر کدام را بررسی کنیم.

برای انجام هر یک از اعمال فوق لازم است تا ابتدا یکبار عملیات مختلفی که بر روی عکس‌های باینری انجام می‌شود را انجام دهیم.

بعد از انجام عملیات با استفاده از دستور زیر می‌توانیم مرکز هر یک از کانتورها را مشخص کرده و آن را رسم کنیم

```
for i in contours:
    M = cv2.moments(i)
    cx = int(M['m10'] / M['m00'])
    cy = int(M['m01'] / M['m00'])
    cv2.circle(image, (cx, cy), 3, (255, 125, 125), -1)
```

در این دستور چه اتفاقی افتاده است؟

در این کد ابتدا یک حلقه `for` با متغیر `i` در لیست کانتورها برای پیمایش ایجاد می‌کنیم

`M = cv2.moments(i)`

- برای کانتور `i`، ممان‌های هندسی را محاسبه می‌کنیم و در دیکشنری `M` ذخیره می‌کنیم.
- این ممان‌ها شامل مقادیری مثل `m00`, `m10`, `m01` هستند که برای محاسبه مرکز هندسی به کار می‌روند.

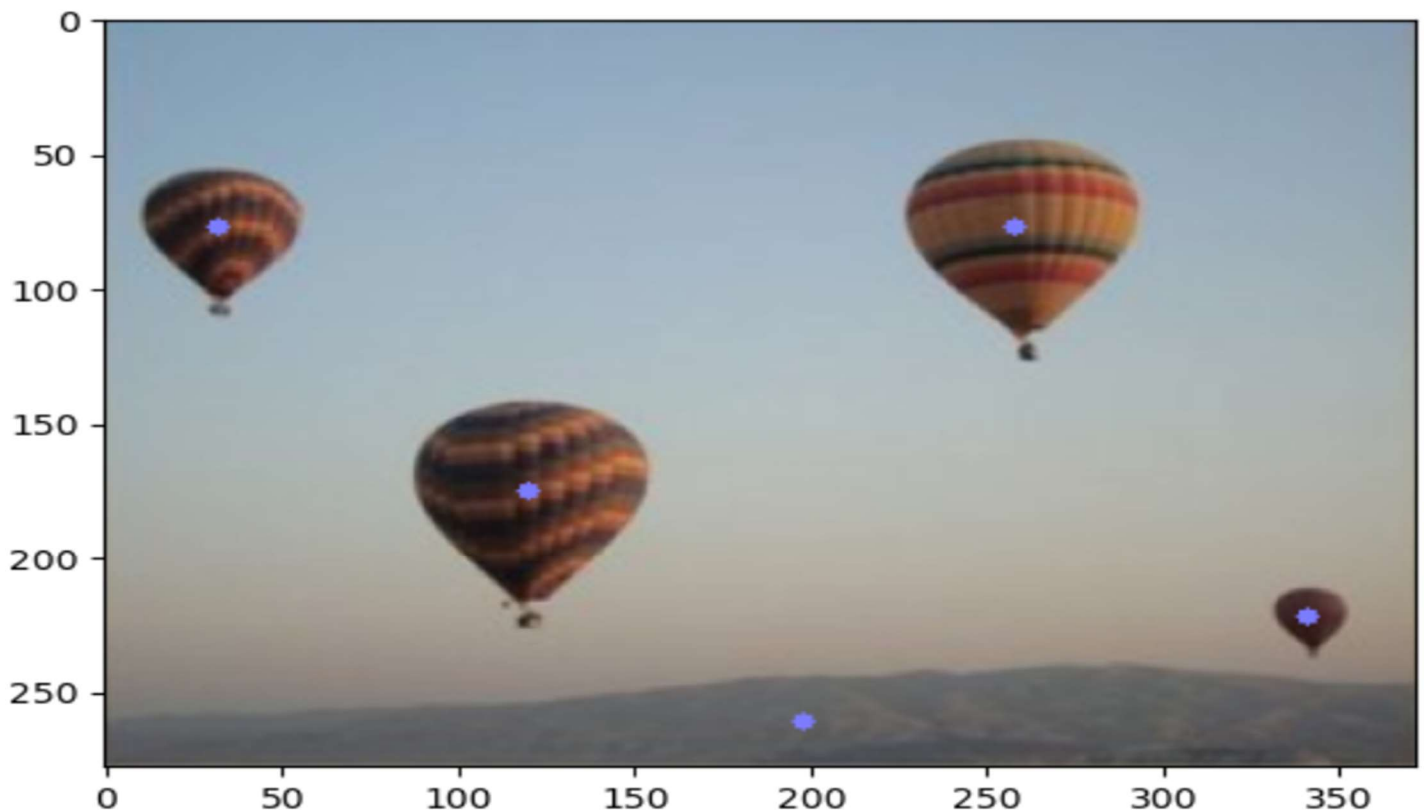
`cx = int(M['m10'] / M['m00'])`

`cy = int(M['m01'] / M['m00'])`

- این دو خط مرکز هندسی کانتور را محاسبه می‌کنند:
 - $m10 / m00 \rightarrow$ مقدار مختصات افقی مرکز (x)
 - $m01 / m00 \rightarrow$ مقدار مختصات عمودی مرکز (y)
- تبدیل به `int` می‌شود چون مختصات پیکسلی باید عدد صحیح باشد.

⚠ اگر `M['m00'] == 0` باشد، تقسیم بر صفر می‌شود (در این کد بررسی نشده، ولی بهتر است قبل از تقسیم، چک شود).

در نهایت با استفاده از فرمول فوق به عکس زیر خواهیم رسید:



همچنین برای محاسبه محیط کانتورها به ترتیب از بزرگترین به کوچکترین می توانیم از قطعه کد زیر استفاده کنیم:

```
for (i,c) in enumerate(sorted_contours):
    M = cv2.moments(c)
    cx = int(M['m10'] / M['m00'])
    cy = int(M['m01'] / M['m00'])
    cv2.circle(copy_img,(cx,cy), 3,(255,125,125),-1)
    cv2.putText(copy_img, str(i+1),(cx,cy), cv2.FONT_HERSHEY_PLAIN,2,(0,255,0),2)
    cv2.drawContours(copy_img,[c], -1,(0,0,255),3)
```

اگر حلقه for موجود در این کد برایتان آشنا نیست به توضیحات زیر دقت کنید:

این یک حلقه for است که همزمان:

- هر کانتور از لیست sorted_contours را دریافت می کند با نام c
- و همچنین شماره‌ی اندیس (index) آن را در لیست به نام i ذخیره می کند.

تابع enumerate() پایتون، یک لیست را گرفته و به ما اجازه می دهد همزمان به:

- مقدار در اینجا: کانتور c
- و شماره‌ی آن مقدار در لیست در اینجا: اندیس i دسترسی داشته باشیم.