



دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر
پروژه درس ساختار و زبان کامپیوتر

عنوان:

پیاده سازی ابزار محاسبه مساحت شکل بسته با استفاده از رزبری پای و دوربین

Area Detection

نگارش

مهدى استاد محمدى، سارينا فرزادنسب، مهدیار مستشار، پارسا ملکيان

استاد

استاد اسدی

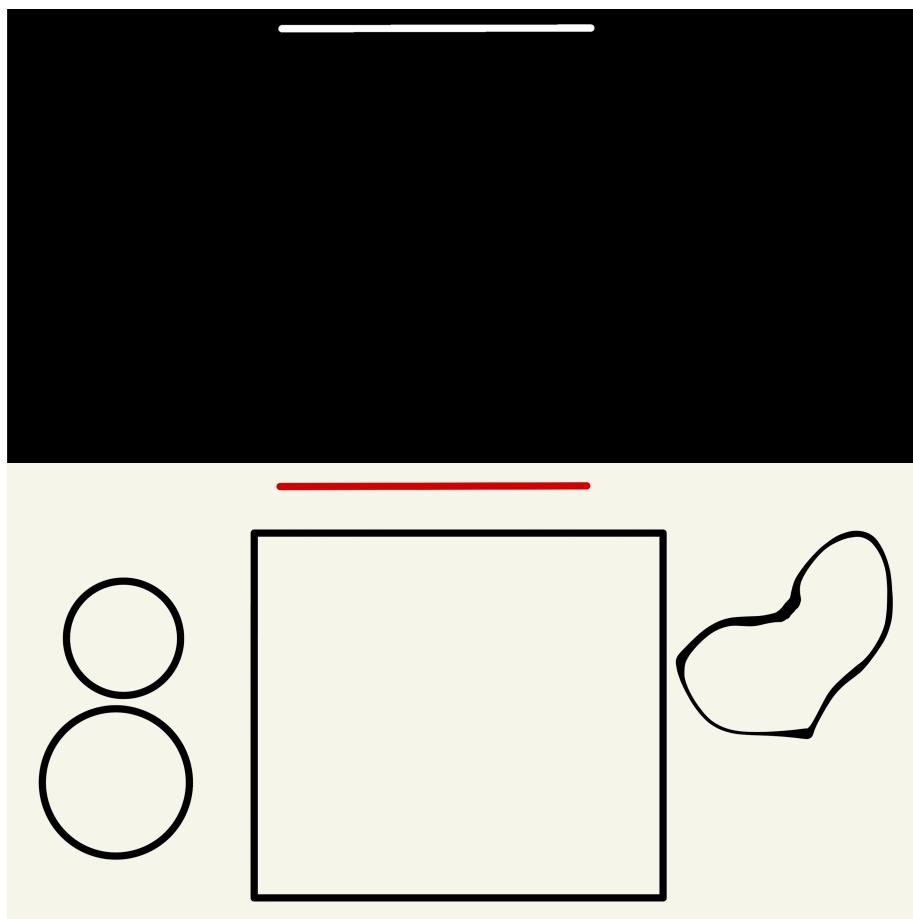
۱۴۰۳ بهمن

چکیده: در این پروژه ما تلاش کردیم تا با کمک زبان پایتون و کتابخانه OpenCV شکل های داخل تصویر را استخراج کرده و مساحت شان را بسته به یک معیار از پیش تعريف شده بدست بیاوریم

۱ مراحل انجام پروژه

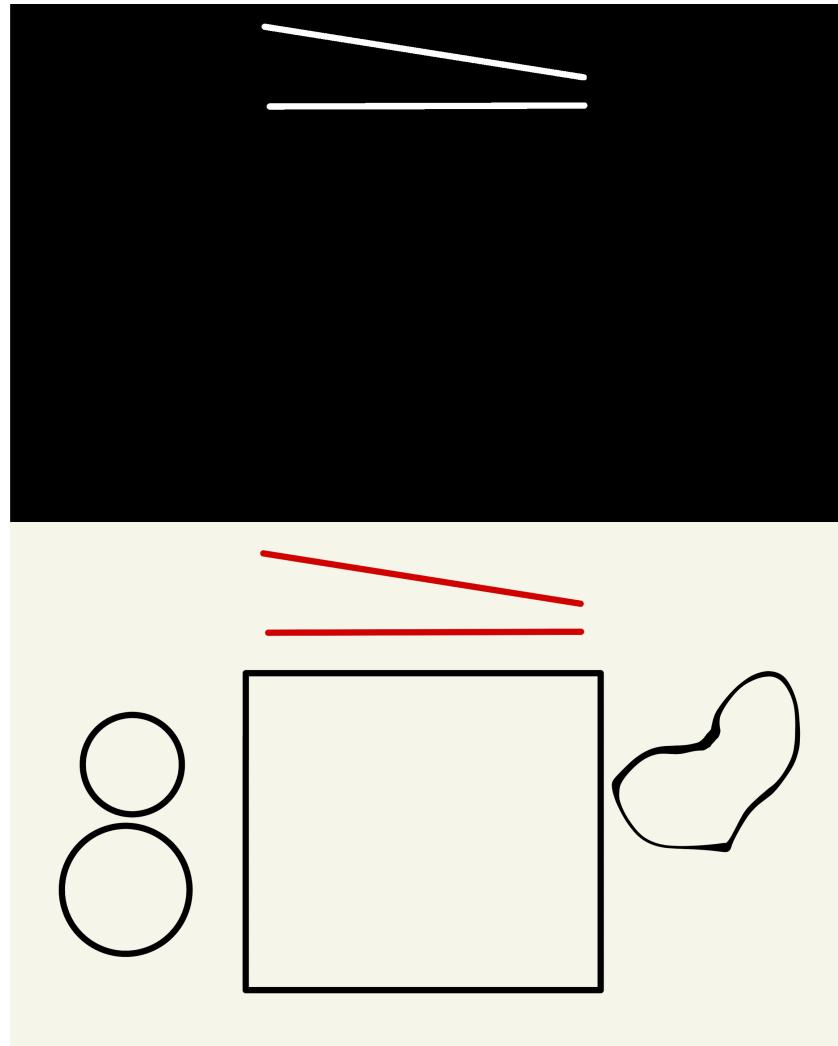
۱-۱ تشخیص متريک

ابتدا تلاش کردیم تا با دادن یک عکس به تصویر کاری کنیم تا متريک که یک خط قرمز رنگ تعريف شده است را تشخیص دهد. برای اينکه چک کنیم به درستی تشخیص داده شده است آن را در یک تب جدا نمایش داده ايم



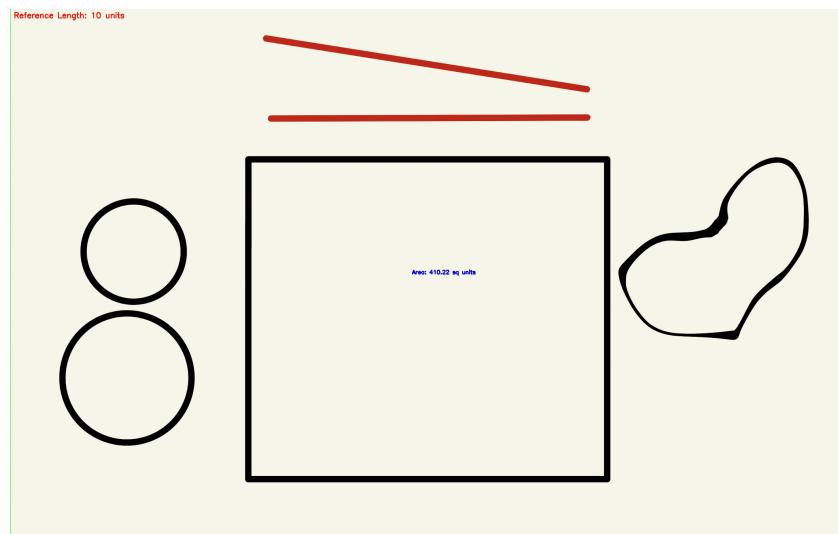
۲-۱ تشخیص چندین متريک به صورت همزمان

در اين بخش مانند بخش قبلی عمل كرديم و سعی كردیم چندین متريک را شناسایي کنیم



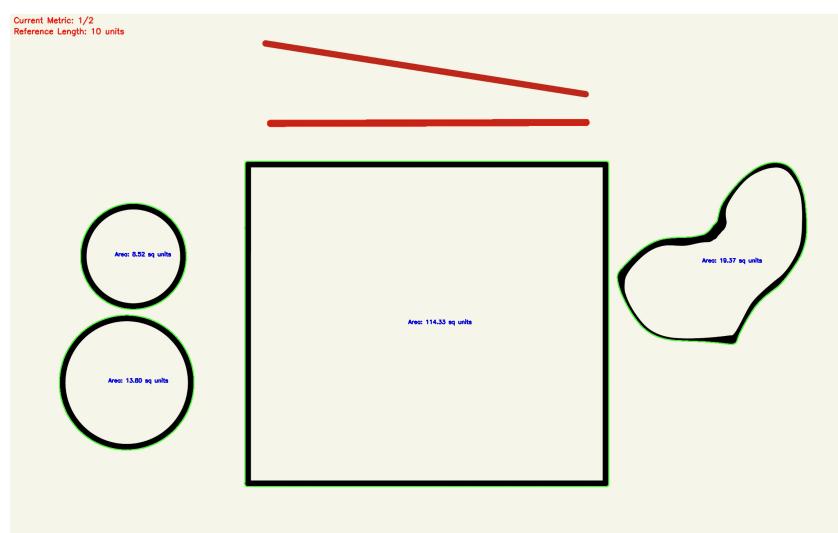
۳-۱ تشخیص یک شکل و اندازه گیری مساحت آن

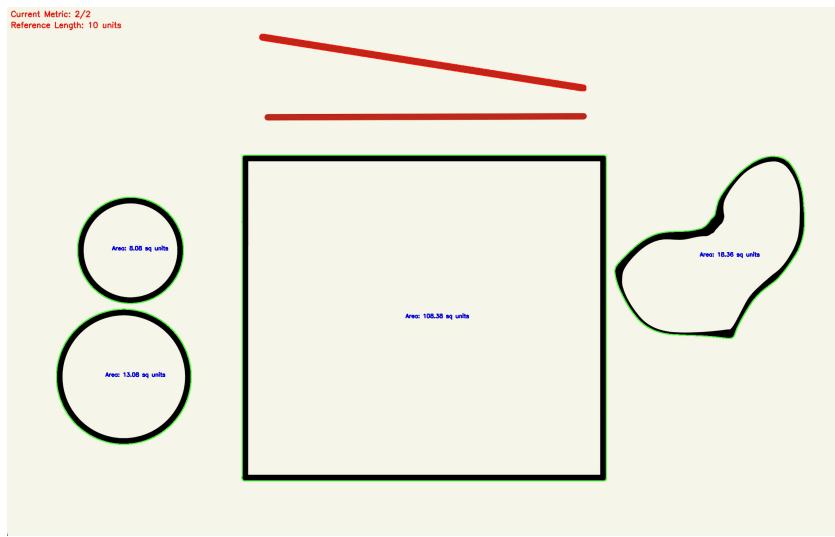
در اين بخش ما به متريک تشخیص داده شده عدد ۱۰ را نسبت داديم و بسته به آن مساحت را بدست آورديم. چالشی که در اين بخش داشتيم اين بود که به جاي شکل هاي داخل صفحه کد ما کل تصویر را به عنوان شکل قلمداد کرد و مساحت آن را نشان داد



۴-۱ تشخیص و جابه‌جایی میان چندین متريک/اندازه‌گیری مساحت شکل‌ها

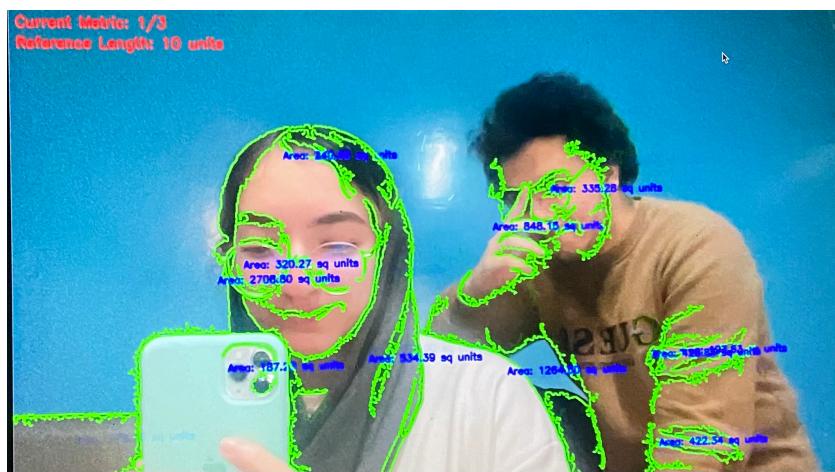
در اين بخش ما مساحت شکل‌ها را با توجه به متريکی که در هر زمان انتخاب شده است بدست آورديم. ميتوانيم با زدن دکمه M ميان متريک‌ها حرکت کنيم. همانطور که ميبيينيد در بالاي صفحه سمت چپ تعداد متريک‌های تشخيص داده شده نشان داده است. و مساحت‌های نشان داده شده برای شکل‌ها با تغيير متريک تغيير پيدا ميکند





۵-۱ اتصال کد به دوربین سیستم

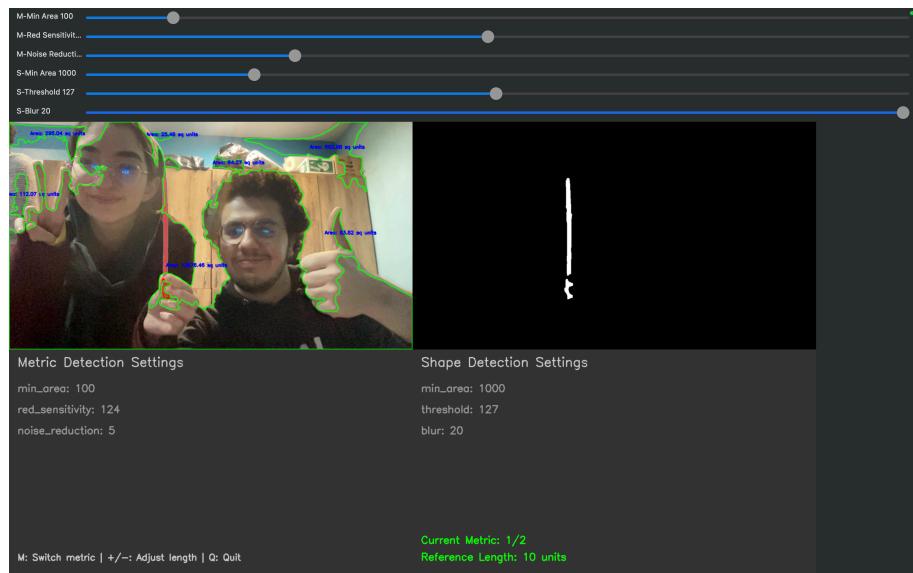
در این مرحله ما به جای ورودی گرفتن عکس کد را به دوربین سیستم متصل کردیم و این کد با توجه به فریم هایی که خود دوربین سیستم میگیرد روی هر فریم فعالیت های پیشین را انجام میدهد. چالشی که در این مرحله داشتیم نداشتن محدودیت روی مساحت اشکال تشخیص داده شده و رنگ قرمزی که به عنوان متریک در نظر داشتیم بود. و به همین دلیل طیف گسترده ای از رنگ های قرمز به عنوان متریک شناسایی شد و تعداد اجسام نیز بسیار زیاد تشخیص داده شد



۶-۱ ساختن منوی تنظیمات

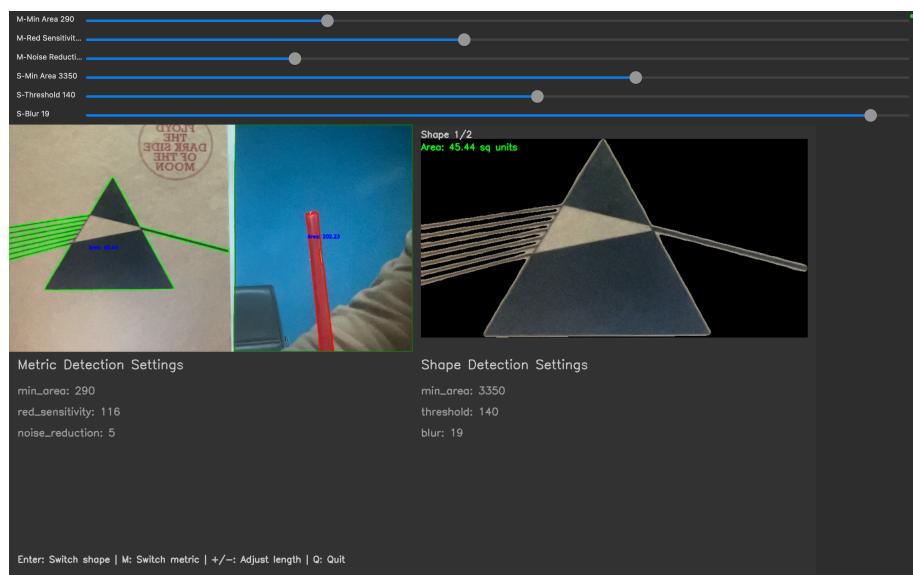
برای حل کردن مشکلات ذکر شده ما یک منوی تنظیمات اضافه کردیم که به کمک آن یک حداقل مساحت برای اجسام در نظر میگیریم و همچنین طیف حساسیت به رنگ قرمز و اندازه متریک را با کمک آن محدود تر میکنیم. برای تشخیص بهتر اجسام یک سیستم blur نیز داریم که اجسام

محسوس را راحت تر تشخیص دهد.



۷-۱ توانایی جابهجایی میان اجسام تشخیص داده شده

در مرحله پایانی با زدن دکمه Enter کاری میکنیم که تصویر زوم شده بتواند بین تمامی اجسام نشان شده به صورت جدا نمایش داده شود. همچنین با آوردن متريک قرمز رنگ به داخل صفحه مساحت جسم نشان داده میشود.



۲ توضیحات کد ها

در این بخش توضیح میدهیم هر بخش از کد در حال انجام چه کاری است

۱-۲ کد زیر مشخصات سنت شده برای متریک و اجسام را نشان میدهد

```
# Settings parameters
self.metric_settings = {
    'min_area': 100,
    'red_sensitivity': 100,
    'noise_reduction': 5
}

self.shape_settings = {
    'min_area': 1000,
    'threshold': 127,
    'blur': 5
}
```

۱-۱-۲ در این بخش بازه رنگ قرمز قابل قبول برای متریک را مشخص میکنیم که پارامتر `sensitivity` از تنظیمات وارد میشود

```
sensitivity = self.metric_settings['red_sensitivity']
lower_red1 = np.array([0, sensitivity, sensitivity])
upper_red1 = np.array([10, 255, 255])
lower_red2 = np.array([160, sensitivity, sensitivity])
upper_red2 = np.array([180, 255, 255])
```

۲-۱-۲ و در نهایت با این بخش از کد متریک های قرمز رنگ که ویژگی های مورد نظر داده شده در تنظیمات را دارند انتخاب میشوند

```
contours, _ = cv2.findContours(red_mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
self.metric_lines = [cnt for cnt in contours
                     if cv2.contourArea(cnt) > self.metric_settings['min_area']]

if self.metric_lines:
    self.current_metric_index = min(self.current_metric_index, len(self.metric_lines) - 1)
else:
    self.current_metric_index = 0
```

```

blur_size = max(1, self.shape_settings['blur'])
if blur_size % 2 == 0:
    blur_size += 1
blurred = cv2.GaussianBlur(gray, ksize: (blur_size, blur_size), sigmaX: 0)

```

۳-۱-۲ برای تشخیص شکل ها ابتدا روی تصویر blur را اعمال میکنیم

۴-۱-۲ بعد از نهایی شدن تصویر خطوط دور اشکال را با کمک کد زیر بدست می آوریم. در این بخش threshold ها را نیز لحاظ میکنیم

```

_, thresh = cv2.threshold(blurred, self.shape_settings['threshold'],
                         maxval: 255, cv2.THRESH_BINARY_INV)

contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

```

۵-۱-۲ در نهایت اشکال بدست آمده در این بخش بر اساس مساحت‌شان فیلتر می‌شوند و اجسام کمتر از مساحت ذکر شده ذخیره نمی‌شوند.

```

self.shape_contours = []
for contour in contours:
    area = cv2.contourArea(contour)
    if area > self.shape_settings['min_area']:
        peri = cv2.arcLength(contour, closed: True)
        approx = cv2.approxPolyDP(contour, 0.02 * peri, closed: True)
        if len(approx) >= 3:
            self.shape_contours.append(contour)

```