تمرین سری دوم – بخش عملی

سوال دوم – گزارش کار

مهرشاد فلاح اسطلخزیر – 401521462

منطق کد:

RANSAC:

تابع fit_line:

```python
def fit_line(x1, y1, x2, y2):
    #TODO: Define a function to compute the slope and intercept of a line given two points
    if x1 == x2:
        return float('inf'), x1
    m = (y2 - y1) / (x2 - x1)
    b = y2 - (m * x2)
    return m, b  # Replace with your implementation; return (slope, intercept)
```

✓ 0.0s

در این تابع ابتدا x هر دو نقطه را مقایسه می‌کنیم و در صورت برابر بودن مقدار بی‌نهایت را به عنوان شیب خط برمی‌گردانیم و برای عرض از مبدا هم یکی از مقادیر x1 یا x2 را برمی‌گردانیم. در غیر اینصورت ابتدا با فرمول شیب خط مقدار m را بدست می‌آوریم و y2 – y1 را بر x2 – x1 تقسیم می‌کنیم. این شیب بدست آمده را در فرمول y = mx + b قرار می‌دهیم و مقدار m را هم بدست می‌آوریم و به عنوان خروجی تابع برمی‌گردانیم.

تابع distance_to_line:

```python
import math
def distance_to_line(x0, y0, x1, y1, x2, y2):
    #TODO: Define a function to calculate the perpendicular distance from a point to a line
    # Hint: The line is defined by two points (x1, y1) and (x2, y2), and you need to find the distance to point (x0, y0)
    # Use the formula for the shortest distance from a point to a line; handle the case where the denominator might be zero
    m, b =fit_line(x1, y1, x2, y2)
    if m == float('inf'):
        return abs(x0 - b)
    num = abs((m * x0)  - y0 + b)
    denom = math.sqrt(m ** 2 + 1)
    return num / denom if denom != 0 else float('inf')
```

✓ 0.0s

برای بدست آوردن فاصله نقطه از تابع از فرمول $d = \frac{|Ax_0+By_0+c|}{\sqrt{A^2+B^2}}$ برای نقطه (x_0,y_0) و فرمول خطی $Ax+By+c = 0$ استفاده می‌کنیم. همانطور که مشخص است اگر شیب بی‌نهایت باشد m برابر با قدرمطلق $-x_0$ b است.

تابع ransac_line_fit:

ابتدا شیب و عرض مبدا را از تابع fit_line که در بالا زدیم محاسبه می‌کنیم. در مرحله بعد یک حلقه برای تمام نقاط می‌زنیم و فاصله آن‌ها را از خط تشکیل شده بدست می‌آوریم و اگر از آستانه‌گذاری کمتر باشند به لیست نقاط درونی اضافه می‌شوند و در غیر اینصورت به لیست نقاط بیرونی اضافه می‌شوند. اگر تعداد نقاط درونی هر مدل از بهترین مدلی که تا الان داشته‌ایم بهتر باشد بهترین مدل را عوض می‌کنیم.

Homography:

سلول بعدی برای کاربرد ransac در homography و تطبیق اشیا است. کلیت کد به این صورت است که با استفاده از SIFT ابتدا نقاط کلیدی و توصیفگرها استخراج می‌شوند. سپس با استفاده از knn و تست lowe نقاط مشابه دو تصویر را پیدا می‌کنیم. Knn برای یافتن نزدیکترین همسایه‌ها و تست lowe برای حذف تطبیق‌های ضعیف کاربرد دارد. در نهایت تطبیق‌های خوب بین دو تصویر را رسم می‌کنیم و در img_matches ذخیره می‌کنیم. در مرحله بعد که در پایین عکسش است هوموگرافی تصویر را پیدا می‌کنیم.

```
if len(good_matches) >= 4:  # Minimum 4 points needed for homography
    # Extract source (object) and destination (scene) points from good matches
    # Hint: Use kp_obj[m.queryIdx].pt and kp_scene[m.trainIdx].pt, reshape to (-1, 1, 2)
    src_pts = np.float32([kp_obj[m.queryIdx].pt for m in good_matches]).reshape(-1, 1, 2)
    dst_pts = np.float32([kp_scene[m.trainIdx].pt for m in good_matches]).reshape(-1, 1, 2)

    H, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
            # Hint: Use findHomography with RANSAC and a reasonable threshold (e.g., 5.0)

    matches_mask = mask.ravel().tolist()
```

اولین تغییر ما در اینجا است که ابتدا نقاط تصویر مبدا و صحنه را بدست می‌آوریم و در نهایت ماتریس تبدیل H و نقاط تاثیرگذار در محاسبه هوموگرافی mask را بدست می‌آوریم.

تبدیل پرسپکتیو را پس از یافتن ماتریس تبدیل بر روی گوشه‌های تصویر اعمال می‌کنیم. و در نهایت با استفاده از مقایسه هیستوگرام صحت تطبیق را بدست می‌آوریم.

```
# print(object_img.shape)
obj_hsv = cv2.cvtColor(object_img, cv2.COLOR_GRAY2BGR)
obj_hsv = cv2.cvtColor(obj_hsv, cv2.COLOR_BGR2HSV)
warped_hsv = cv2.cvtColor(warped_scene, cv2.COLOR_BGR2HSV)

hist_obj = cv2.calcHist([obj_hsv], [0], None, [256], [0,256])
hist_warped = cv2.calcHist([warped_hsv], [0], None, [256], [0, 256])
```

در اینجا هم ابتدا object_img را که gray است به دیفالت رنگی opencv که BGR است تبدیل می‌کنیم و بعد BGR را به HSV تبدیل می‌کنیم و warped_scene که همان عکس صحنه است که پرسکپتیو شده را هم تبدیل به HSV می‌کنیم و در مرحله بعد هیستوگرام تصویر را محاسبه می‌کنیم.

Sobel_prewitt_canny:

Sobel:

برای پیاده‌سازی لبه‌یاب Sobel ابتدا فیلترهای آن را تشکیل می‌دهیم. در مرحله بعد در یک حلقه تودرتو برای هر پیکسل کانولوشن بین کرنل Sobel و پنجره 3 در 3 را محاسبه می‌کنیم و همچنین ماتریس magnitude را محاسبه کرده و ذخیره می‌کنیم و در نهایت به عنوان خروجی قرار می‌دهیم.

Prewitt:

به مانند Sobel عمل می‌کنیم صرفا اعداد کرنل متفاوت است.

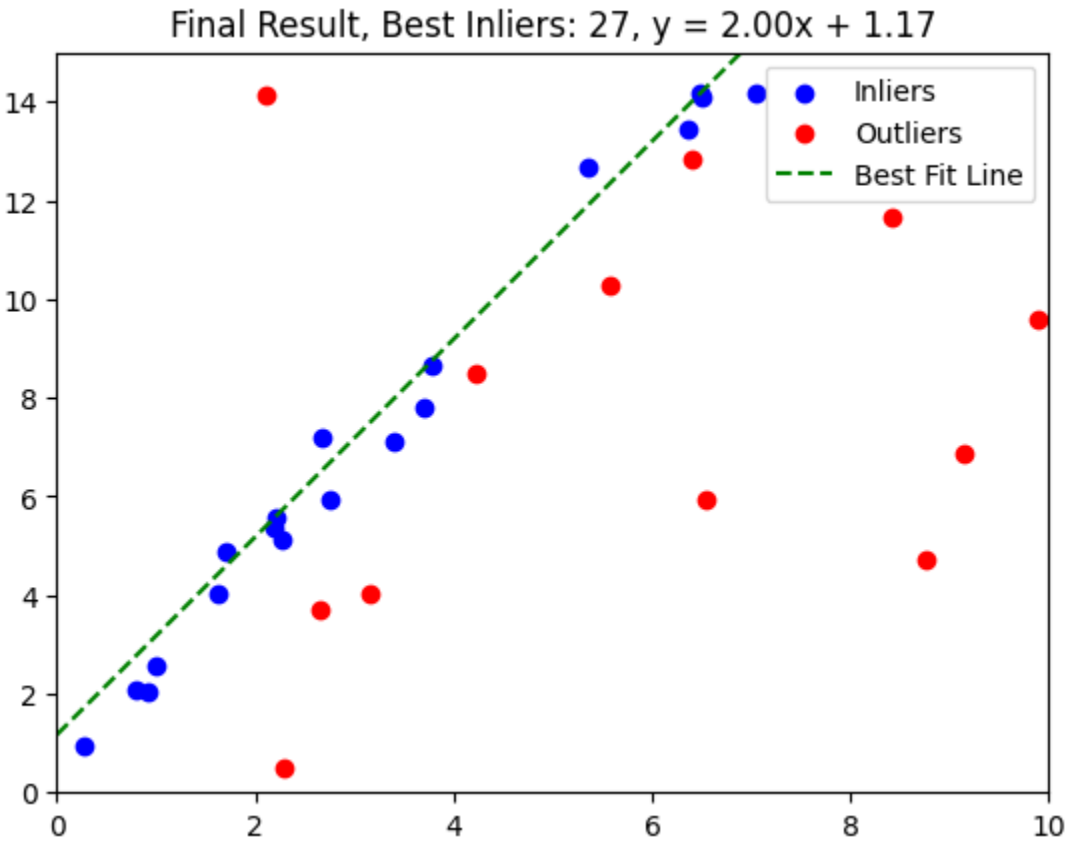
Canny:

برای محاسبه کنی ابتدا GaussianBlur استفاده می‌کنیم و عکس را smooth می‌کنیم و در مرحله بعد sobel را محاسبه می‌کنیم. بعد از این از تابع nms استفاده می‌کنیم. برای پیاده‌سازی nms مقادیر ماتریس direction و magnitude را خارج از تابع محاسبه می‌کنیم. مقادیر تابع direction به رادیان بدست آمده‌اند به همین دلیل آن‌ها را به درجه تبدیل می‌کنیم و برای اینکه بین 0 و 180 باشند ابتدا +180 می‌کنیم و بعد باقیمانده تقسیم آن‌ها بر 180 را بدست می‌آوریم. بعد از بدست آوردن زاویه ناحیه‌ها را مشخص می‌کنیم. برای این منظور صرفا مقادیر خانه‌هایی که باید برای nms مقایسه شوند را بدست می‌آوریم. در مرحله آخر مقایسه را انجام می‌دهیم و مقادیر را در هر سلول قرار می‌دهیم. برای آستانه‌گذاری دو مرحله‌ای دو ماتریس برای لبه‌های قوی و ضعیف ایجاد می‌کنیم و درون تابع ابتدا جهات مختلف را مشخص می‌کنیم. این جهات برای اینکه

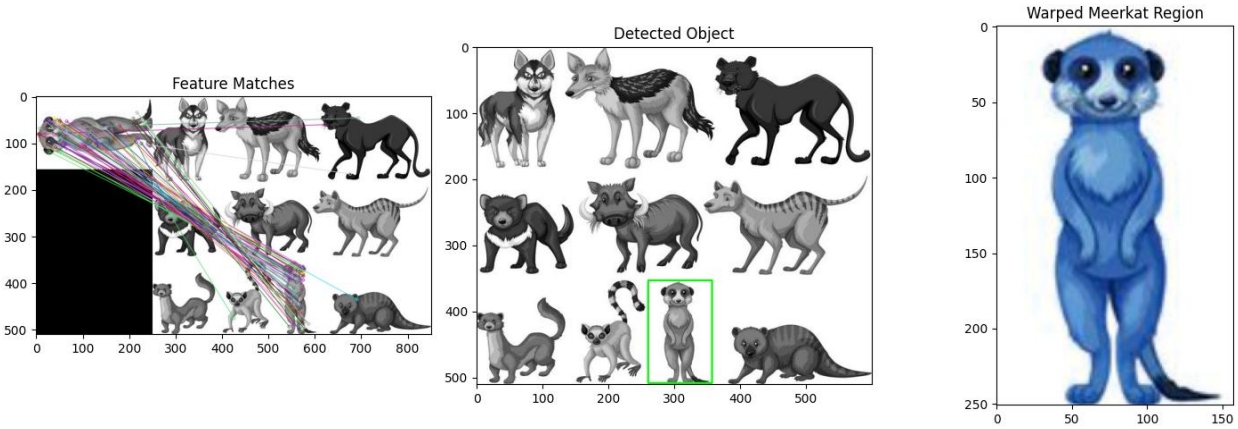
لبه‌های ضعیف چسبیده به لبه‌های قوی را پیدا کنیم کاربرد دارند. بعد از BFS استفاده می‌کنیم. لبه‌های قوی را در frontier وارد می‌کنیم. هر لبه ضعیفی که به یک لبه قوی وصل باشد را وارد frontier می‌کنیم و آن را مساوی لبه قوی قرار می‌دهیم و به این صورت لبه‌های قوی را پیدا می‌کنیم. در سلول نهایی هم از Canny که در خود opencv تعریف شده استفاده می‌کنیم.

تحلیل نتایج:

بهترین نتیجه RANSAC:



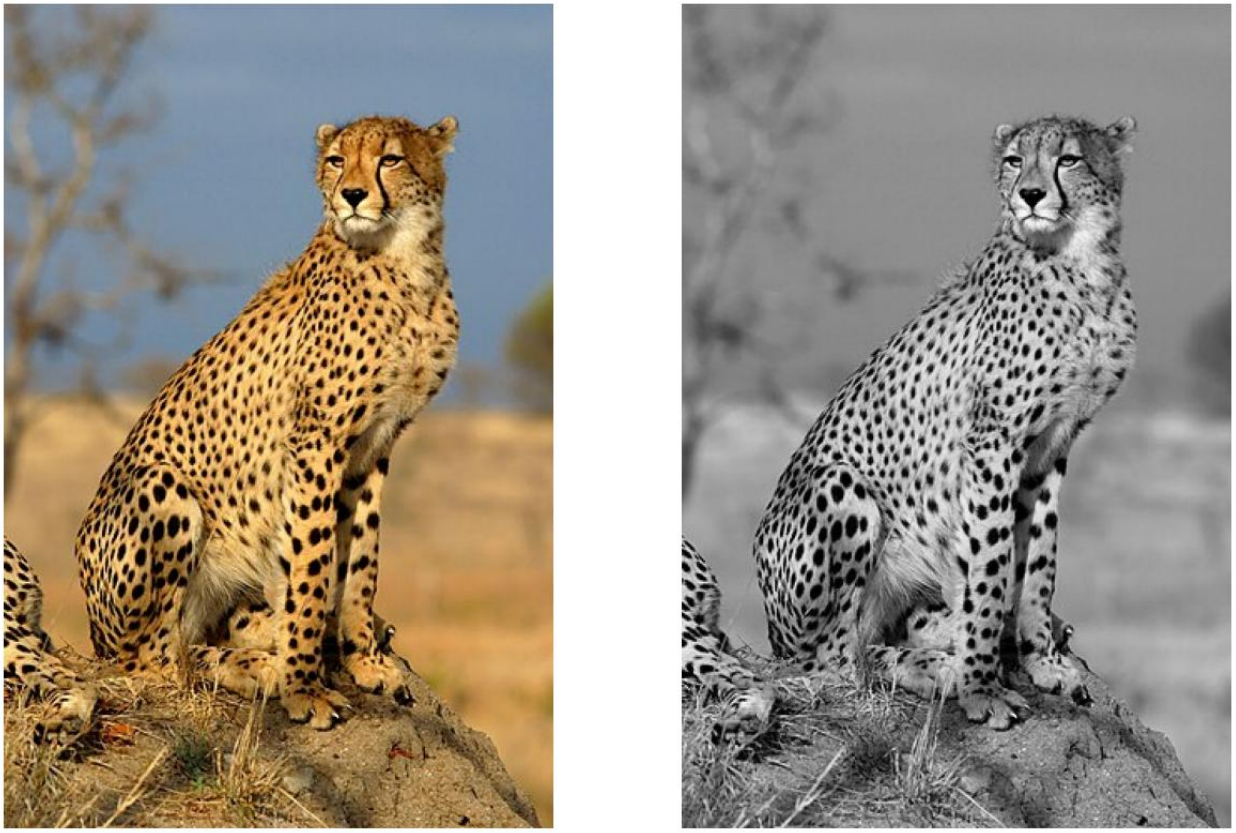
این بهترین نتیجه بدست آمده از اجرای بیست باره RANSAC بود که نزدیک به خواسته در داک سوالات بود.



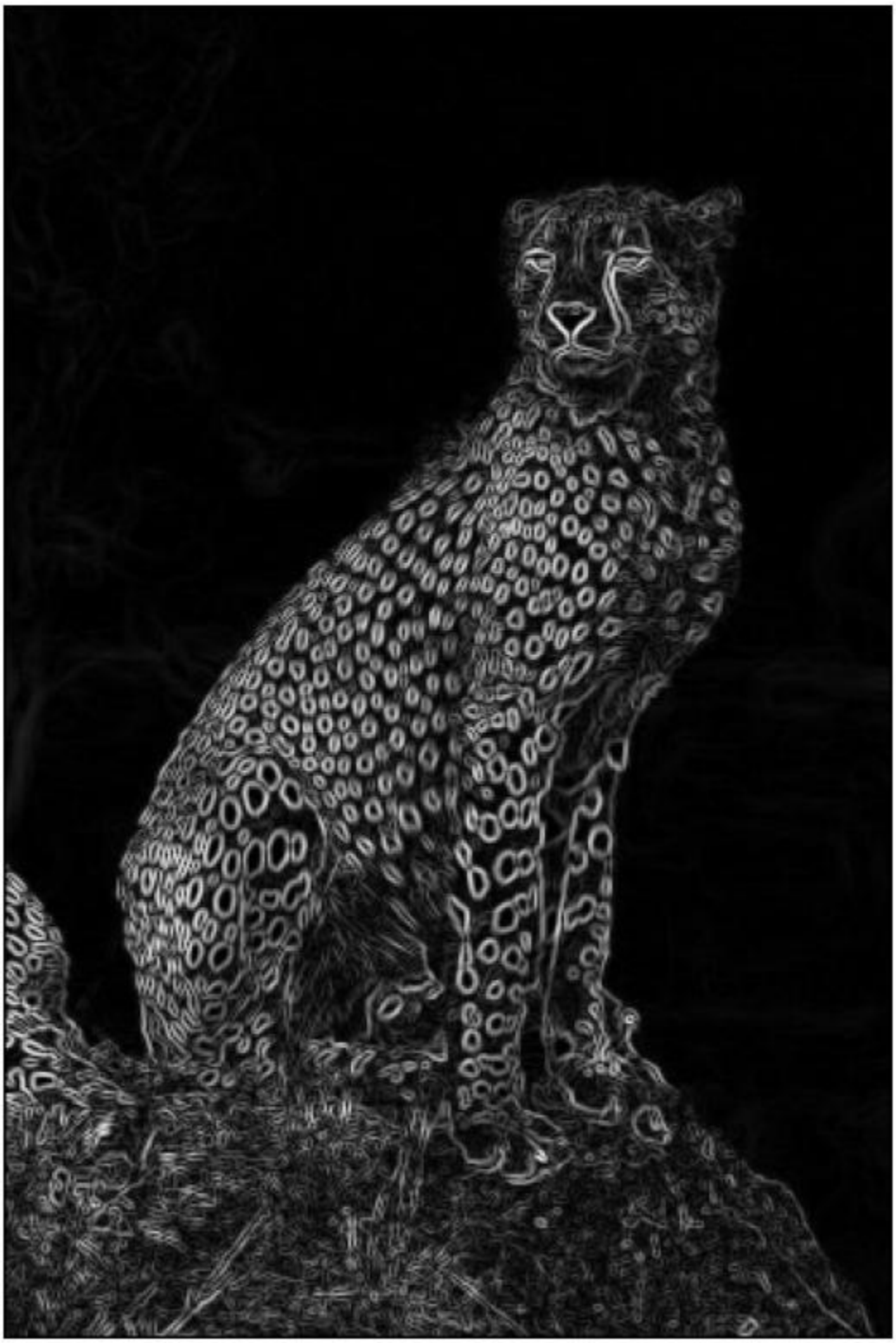
این هم نتیجه اجرای homography است که تصویر meerkat را نود درجه چرخانده‌ایم. همچنین آستانه مقایسه هیستوگرام را به 0.65 تغییر دادیم تا مطابقت بیابد.

Canny:

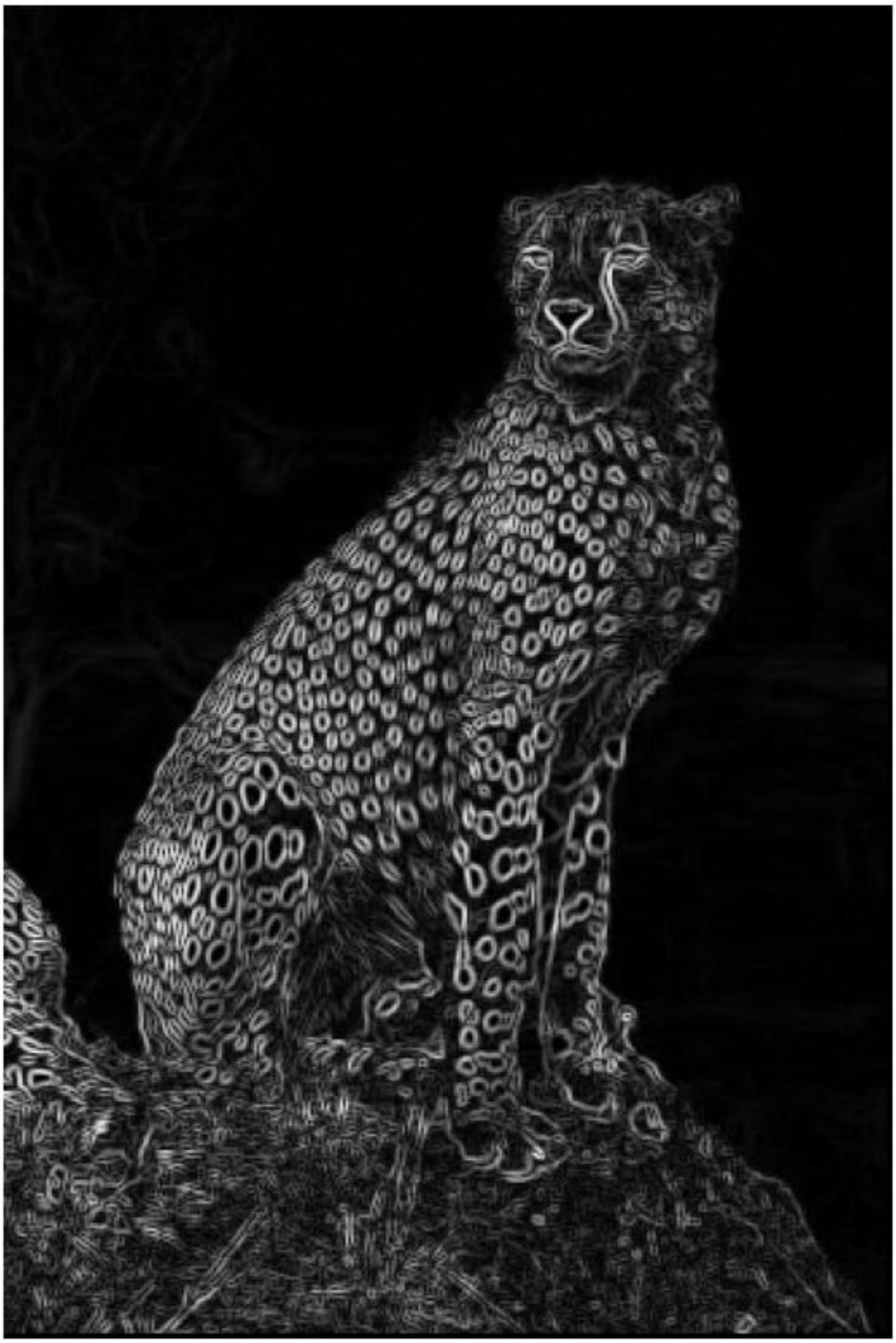
عکس اصلی:



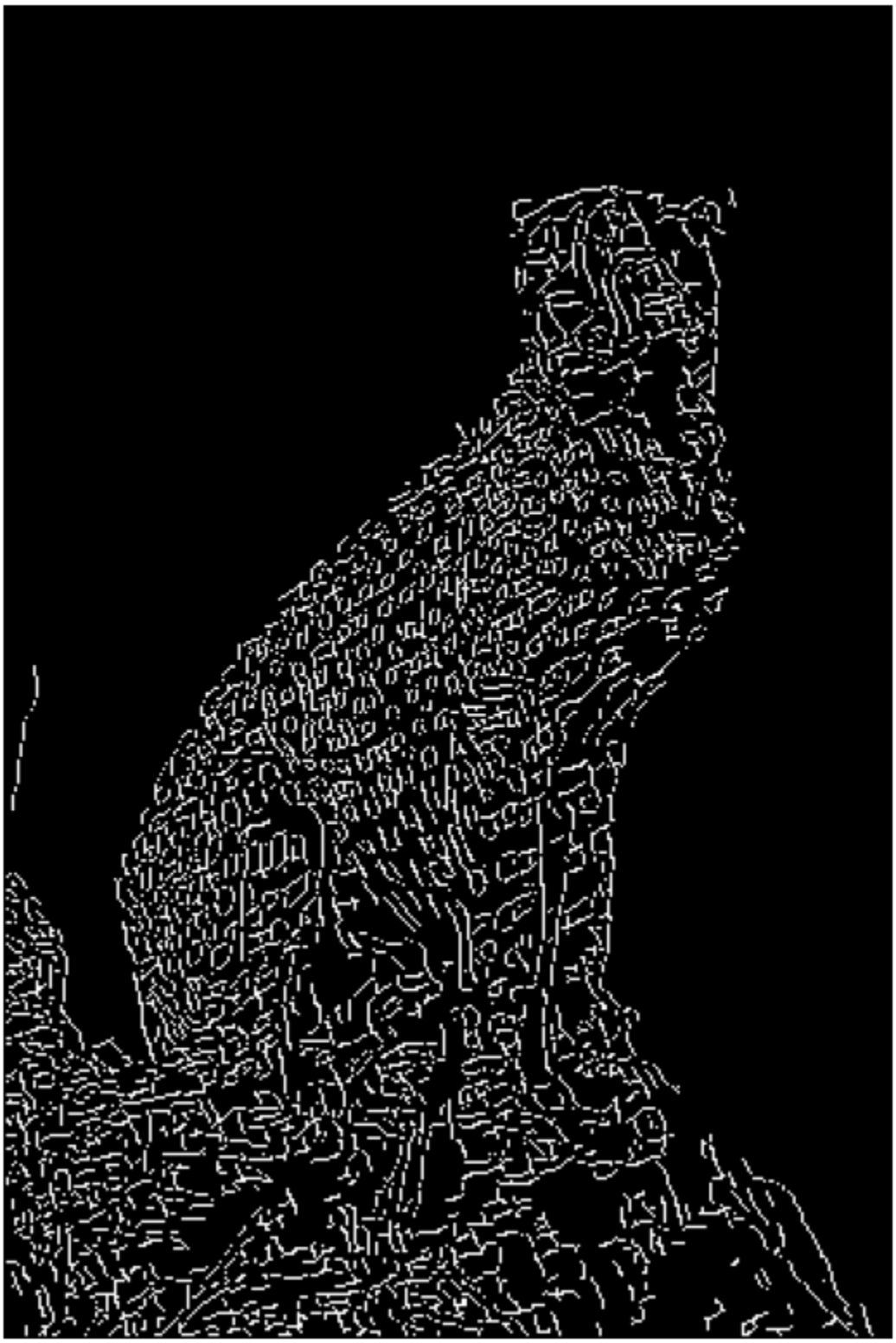
فیلتر Sobel:



فیلتر Prewitt:



فیلتر Canny دستی: این فیلتر با آستانه گذاری پایین 35 و بالای 70 انجام شده.



فیلتر Canny: این فیلتر با آستانه‌گذاری پایین 100 و بالای 200 انجام شده.