بسم الله الرحمن الرحیم

نام دانشجو: سیدرضا مسلمی

شماره دانشجویی: 220700046

نام استاد: استاد فرزانه عبدلی

دانشکدگان فارابی، دانشگاه تهران

مقدمه:

در این گزارش، جزئیات پروژه نهایی درس بینایی کامپیوتر را ارائه می‌دهم. هدف اصلی این پروژه، تشخیص خطوط جاده در تصاویر و ویدیوهای رانندگی بوده است. برای دستیابی به این هدف، از الگوریتم‌ها و روش‌های بینایی کامپیوتر مانند استخراج ویژگی‌ها، تشخیص لبه‌ها، تبدیل رنگ و هاف تبدیل استفاده شده است. پس از پیاده‌سازی، عملکرد آن بر روی تصاویر و ویدیوها ارزیابی شده است.

بخش ۱:

تابع cv2.HoughLinesP در کتابخانه OpenCV از الگوریتم تبدیل هاف برای تشخیص خطوط در تصاویر استفاده می‌کند. این تابع پارامترهای مختلفی دارد که تاثیر متفاوتی بر روی خروجی دارند. در ادامه، پارامترهای اصلی این تابع و تاثیر هر کدام توضیح داده می‌شوند:

1. image: تصویر ورودی که بر روی آن خطوط قرار است تشخیص داده شوند. این تصویر باید به صورت تصویر سیاه و سفید (binary) باشد.

2. rho: نرخ رزولوشن محور rho در پارامتریک تبدیل هاف. این پارامتر تعیین می‌کند که خطوط با چه فاصله از مبدا تبدیل هاف مشخص شوند.

3. theta: نرخ رزولوشن زاویه در پارامتریک تبدیل هاف. این پارامتر تعیین می‌کند که خطوط با چه زاویه‌هایی تشخیص داده شوند.

4. threshold: آستانه‌ای که برای تشخیص خطوط استفاده می‌شود. هر خطی که توسط تبدیل هاف شناسایی شود و تعداد رأی‌هایی که در این خط به دست می‌آید بیشتر از این آستانه باشد، به عنوان خروجی قرار می‌گیرد.

5. minLineLength: حداقل طولی که یک خط باید داشته باشد تا به عنوان خروجی معتبر در نظر گرفته شود.

6. maxLineGap: حداکثر فاصله مجاز بین قطعه‌های خطی مختلف تشخیص داده شده در یک خط. اگر فاصله بین دو قطعه خطی بیشتر از این مقدار باشد، آنها به عنوان خطوط مجزا در نظر گرفته می‌شوند.

تفاوت اصلی بین تابع cv2.HoughLinesP و الگوریتم تبدیل هاف پایه در پیاده‌سازی و روش تشخیص خطوط استفاده شده است. در الگوریتم تبدیل هاف پایه، تمام نقاط تصویر در فضای پارامتریک تبدیل هاف بررسی می‌شوند و خطوط توسط تعداد رأی‌هایی که در آنها به دست می‌آیند شناسایی می‌شوند. اما در تابع cv2.HoughLinesP، از روش خوشه‌بندی خطوط استفاده می‌شود. این روش با استفاده از آستانه‌ی تعداد رأی‌ها و تحلیل فواصل بین نقاط مجاور، خطوط را در تصویر تشخیص می‌دهد. به عبارت دیگر، تابع cv2.HoughLinesP نسخه بهبود یافته‌ای از الگوریتم تبدیل هاف است که دقت و قابلیت استفاده‌پذیری بیشتری دارد.

این کد شامل چندین بخش است که هر بخش وظایف مختلفی را انجام می‌دهد. در زیر توضیحاتی درباره هر بخش آورده شده است:

بخش ۲:

- تصویر ورودی را به اندازه نصف عرض و ارتفاع اصلی آن تغییر اندازه می‌دهد.

- تصویر تغییر اندازه‌ی حاصل را به سطح خاکستری تبدیل می‌کند.

- cv2.GaussianBlur: این عملیات برای اعمال فیلتر گوسی بر روی تصویر استفاده می‌شود. فیلتر گوسی باعث انتشار نرمال تصویر می‌شود و تأثیر نویز را کاهش می‌دهد. پارامترهایی که در این عملیات استفاده شده‌اند عبارتند از: image_ که تصویر خاکستری حاصل از تبدیل BGR به خاکستری است و (3, 3) که اندازه فیلتر گوسی است.

- cv2.Canny: این عملیات برای تشخیص لبه‌ها در تصویر استفاده می‌شود. از الگوریتم Canny استفاده می‌کند که با استفاده از یافتن تغییرات شدت نور در تصویر، لبه‌ها را شناسایی می‌کند. دو پارامتر اصلی که در این عملیات

استفاده شده‌اند عبارتند از: threshold1=50 که حداقل مقدار شدت نور برای تشخیص لبه‌ها است و threshold2=300 که حداکثر مقدار شدت نور برای تشخیص لبه‌ها است.

با استفاده از این دو عملیات در بخش ۲، تصویر ورودی به صورت تغییر اندازه داده شده و سپس با اعمال فیلتر گوسی و تشخیص لبه‌ها، آماده‌ی پردازش در بخش بعدی می‌شود.

```
#---------------- Section 2 ----------------
    h, w = image.shape[:2]
    w //= 2
    h //= 2
    image = cv2.resize(image, (w, h))
    image_ = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    g_image = cv2.GaussianBlur(image_, (3, 3), 0)

    edges = cv2.Canny(g_image, threshold1=50, threshold2=300)
```

بخش ۳:

- یک حاشیه را به عنوان یک آرایه‌ی numpy که شامل مختصات یک پنج ضلعی است، تعریف می‌کند.

- با پرکردن حاشیه تعریف شده با پیکسل‌های سفید (عدد ۲۵۵)، یک ماسک ایجاد می‌شود.

- ماسک را بر روی لبه‌های تشخیص داده شده با استفاده از الگوریتم تشخیص لبه Canny اعمال می‌کند.

```
#---------------- Section 3 ----------------
    # Bounding image to five vector
    contour = np.array([[0, h], [w, h], [w//2, 6*h//10], [w//4, 6*h//10], [3*w//4, 6*h//10]])
    mask = np.zeros_like(edges)
    cv2.fillPoly(mask, np.int32([contour]), 255)
    masked_edges = cv2.bitwise_and(edges, mask)
```

بخش ۴:

- با استفاده از تبدیل هاف با احتمال، تشخیص خط انجام می‌شود.

- خط‌های تشخیص داده شده را بدست می‌آورد.

بخش ۵ و بخش امتیازی (بخش ۷):

- تابعی به نام `calculator` را تعریف می‌کند که با استفاده از شیب و اینترسپت خط، مختصات آن را محاسبه می‌کند.

- بررسی می‌کند که آیا خط‌های تشخیص داده شده وجود دارند یا خیر.

- خط‌های تشخیص داده شده را بر اساس شیب آن‌ها به خطوط چپ و راست تقسیم می‌کند.

- اگر خطوط چپ یا راست خالی نباشند:

  - میانگین خط‌های مربوطه را محاسبه می‌کند.

  - با استفاده از تابع `calculator` نقاط شروع و پایان خط را بدست می‌آورد.

  - خط را با استفاده از تابع `cv2.line` در تصویر رسم می‌کند.

- اگر خطوط چپ یا راست خالی نباشند:

  - نقاط به دست آمده در فریم قبلی را در prev_left/right_line ذخیره می‌کند.

  - خط حاصل در بخش قبل را رسم می‌کند.

با انجام این مراحل، پیوستگی خطوط حفظ و نویزهای احتمالی حذف می‌شوند؛ همچنین تا حدودی قدرت تشخیص در شرایط جوی نامناسب و شرایط نامطلوب دیگر بالا می‌رود.

```python
#---------------- Section 5 & Bonus(Section 7) ----------------
def calculator(x1, y1, x2, y2, w, h):

    # Calculate the slope and intercept of the line
    slope = (y2 - y1) / (x2 - x1)
    intercept = y1 - slope * x1

    # Set the starting and ending points for the line
    x1 = w
    y1 = int(slope * x1 + intercept)

    x2 = int((h - intercept)/slope)
    y2 = h

    return x1,y1,x2,y2
```

```
#---------------- Section 4 ----------------
    # Perform Hough line detection using Probabilistic Hough Transform
    lines = cv2.HoughLinesP(
        masked_edges,
        rho=1,
        theta=np.pi / 180,
        threshold=55,
        maxLineGap=50
    )


    #---------------- Section 4 & 5 & Bonus(Section 7) ----------------
    if lines is not None:
        left_lines = []
        right_lines = []

        for line in lines:
            x1, y1, x2, y2 = line[0]
            slope = (y2 - y1) / (x2 - x1)

            if slope < 0:
                right_lines.append(line)
            elif slope > 0:
                left_lines.append(line)

        if len(left_lines) > 0:
            left_line_avg = np.mean(left_lines, axis=0, dtype=np.int32)
            x1, y1, x2, y2 = left_line_avg[0]
            x1, y1, x2, y2 = calculator(x1, y1, x2, y2, w, 6*h//10)
            cv2.line(image, (x1, y1), (x2, y2), (0, 0, 255), 1)
            prev_left_line = (x1, y1, x2, y2)
        elif prev_left_line is not None:
            x1, y1, x2, y2 = prev_left_line
            cv2.line(image, (x1, y1), (x2, y2), (0, 0, 255), 1)

        if len(right_lines) > 0:
            right_line_avg = np.mean(right_lines, axis=0, dtype=np.int32)
            x1, y1, x2, y2 = right_line_avg[0]
            x1, y1, x2, y2 = calculator(x1, y1, x2, y2, 0, 6*h//10)
            cv2.line(image, (x1, y1), (x2, y2), (0, 0, 255), 1)
            prev_right_line = (x1, y1, x2, y2)
        elif prev_right_line is not None:
            x1, y1, x2, y2 = prev_right_line
            cv2.line(image, (x1, y1), (x2, y2), (0, 0, 255), 1)

    return image, prev_left_line, prev_right_line
```

بخش ۶:

- تابعی به نام `play_video` را تعریف می‌کند که فریم‌های ویدیو را پردازش می‌کند.

- فریم‌ها را از ویدیو می‌خواند و هر فریم را به تابع `detector` منتقل می‌کند.

- فریم پردازش شده را در یک پنجره نمایش می‌دهد.

- در صورت فشردن کلید 'q'، پخش ویدیو متوقف می‌شود و پنجره‌ها بسته می‌شوند.

```python
#---------------- Section 6 ----------------
def play_video(video):
    prev_left_line, prev_right_line = None, None
    while video.isOpened():
        ret, frame = video.read()

        if not ret:
            break

        processed_frame, prev_left_line, prev_right_line = detector(frame, prev_left_line, prev_right_line)

        cv2.imshow('Processed Frame', processed_frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cv2.destroyAllWindows()
```

بخش اصلی (main):

- دو شیء `VideoCapture` را برای خواندن ویدیوهای با نام‌های 'vid1.mp4' و 'vid2.mp4' ایجاد می‌کند.

- برای هر ویدیو تابع `play_video` را فراخوانی می‌کند.

- یک تصویر با نام 'img1.jpg' خوانده شده و به تابع `detector` منتقل می‌شود.

- تصویر پردازش شده را نمایش می‌دهد.

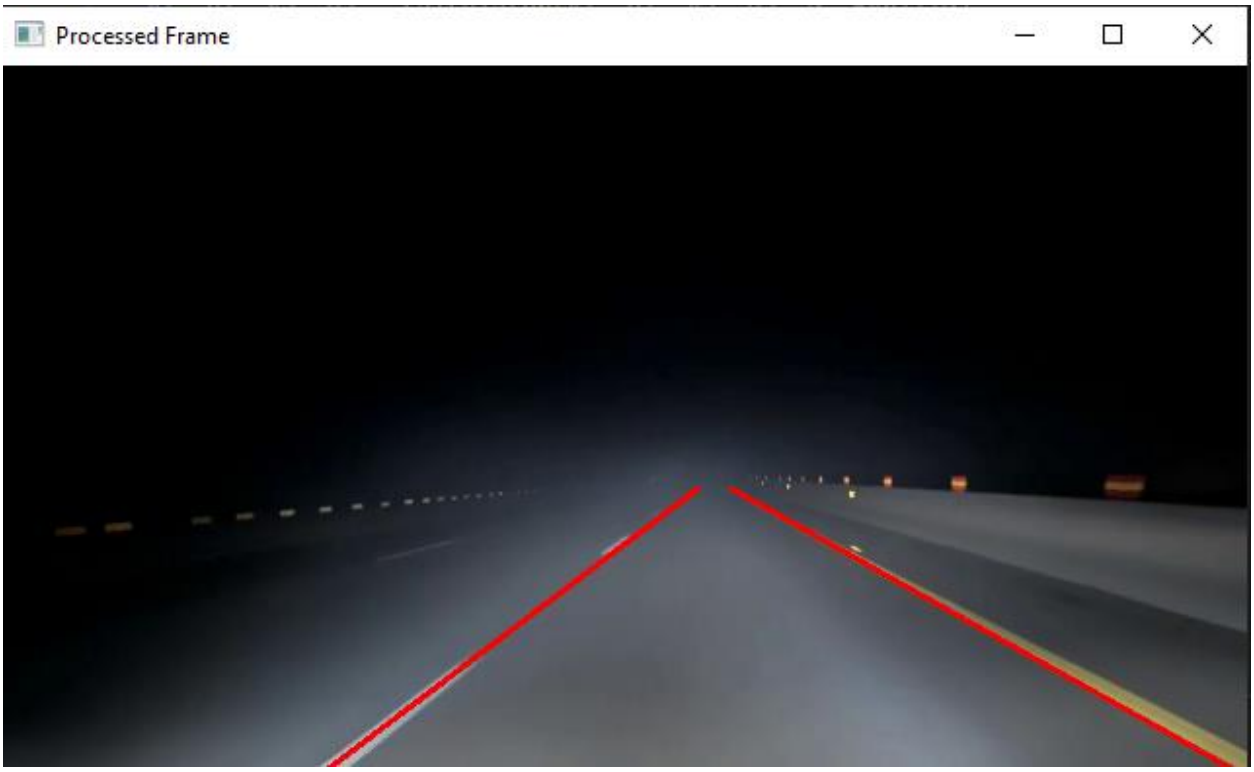
```python
# main
video1 = cv2.VideoCapture('Final_project\\vid1.mp4')
play_video(video1)

video2 = cv2.VideoCapture('Final_project\\vid2.mp4')
play_video(video2)

image = cv2.imread('Final_project\img1.jpg')
image,_,_ = detector(image, None, None)
cv2.imshow('Original Image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Ouputs:


Processed Frame


Processed Frame