AI VIET NAM – COURSE 2023

Lane Detection

(Computer Vision Foundation)

Ngày 5 tháng 10 năm 2023

Giới thiệu về bài tập xác định làn đường (ego-lane) cho xe tự hành:

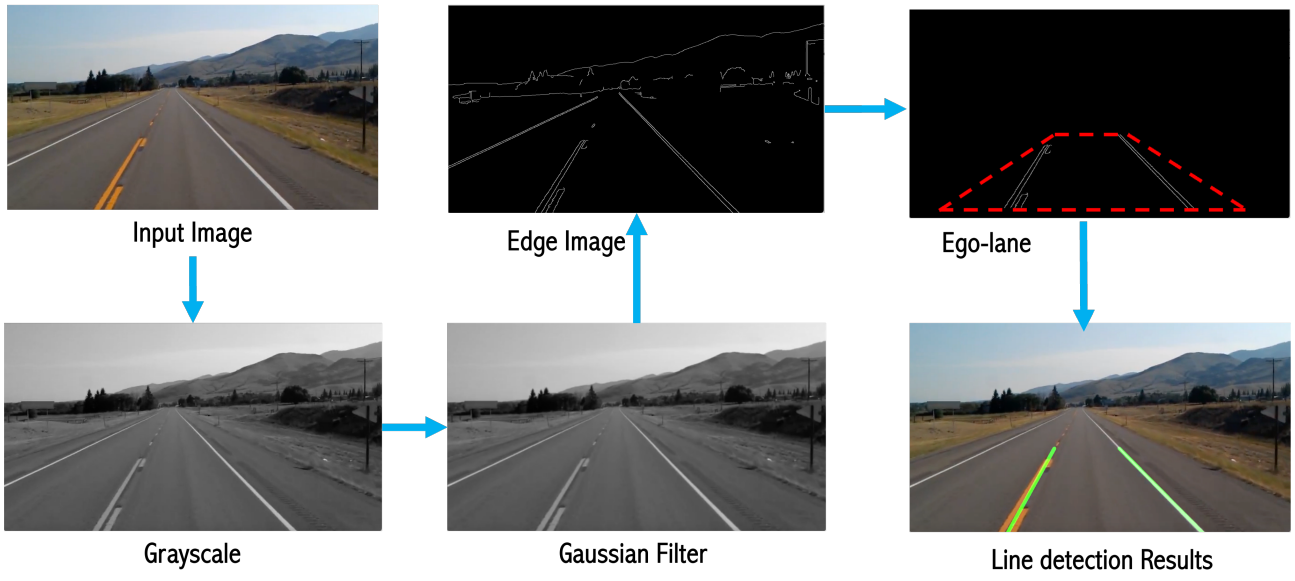
Phát hiện làn đường (lane detection) là một nhiệm vụ thị giác máy tính liên quan đến việc xác định ranh giới của làn đường lái xe trong video hoặc hình ảnh. Mục tiêu là xác định và theo dõi chính xác vạch kẻ làn đường trong thời gian thực, ngay cả trong những điều kiện khó khăn như ánh sáng kém, ánh sáng chói hoặc bố cục đường phức tạp.

Phát hiện làn đường là một thành phần quan trọng của hệ thống hỗ trợ người lái tiên tiến (ADAS) và xe tự hành, vì nó cung cấp thông tin về bố cục đường và vị trí của phương tiện trong làn đường, điều này rất quan trọng cho việc điều hướng và an toàn. Các thuật toán thường sử dụng kết hợp các kỹ thuật thị giác máy tính, chẳng hạn như phát hiện cạnh, lọc màu và biến đổi Hough, để xác định và theo dõi các vạch kẻ làn đường.

Cho trước dữ liệu là đoạn video chứa thông tin hình ảnh làn đường (dataset.mp4), hãy phát triển chương trình xác định vị trí làn đường. Để hoàn thành bài tập này, bạn cần nắm rõ các giải thuật về grayscale conversion, blurring techniques, edge detection, and line detection. Hình 1, thể hiện kết quả sau thi thực hiện giải thuật phát hiện làn đường dành cho ô tô.

Để hoàn thành bài tập này bạn cần từng bước hoàn thiện các bước còn thiếu bên dưới để phát hiện làn đường từ ảnh đầu vào:

```
def detect_lane_single_image(image):

  # Step 1: Convert the RGB image to Gray scale
  grayscale = #******************** Your code here ****************************

  # Step 2: Remove noise in the input image using Gaussian Filter
  kernel_size = 5
  blur =  #******************** Your code here ****************************

  # Step 3: Detect edges using Canny Algorithm
  low_t = 50
  high_t = 150
  edges = #******************** Your code here ****************************

  # Step 4: Limit the search space based on the edge information for finding the lane
    region
  region = limit_research_space(edges)

    # Step 5: Use hough transform to detect left and right lane
  hough = hough_transform(region)

  # Step 6: Draw the left and right lane to the screen
  result = draw_lane_lines(image, lane_lines(image, hough))

  return result
```



Hình 1: Kết quả phát hiện làn đường (lane detection) từ ảnh đầu vào

Bài tập 1: (Step 1: grayscale conversion) Hãy hoàn thiện đoạn code sau để chuyển đổi ảnh đầu vào (RGB color) to grayscale color sử dụng thư viện OpenCV.

```
# Step 1: Convert the RGB image to Gray scale
image = cv2.imread("test_image.jpg")
grayscale = #******* your code here ******************
print("total intensities: ", sum(sum(grayscale)))
```

Question 1: Hãy cho biết kết quả sau khi thực hiện đoạn code ở bài tập 1

```
a) Total intensities is  162333
b) Total intensities is  172333
c) Total intensities is  182333
d) Total intensities is  192333
```

Bài tập 2: (Step 2: Noise removal) Hãy hoàn thiện đoạn code sau để chuyển giảm nhiễu từ ảnh đầu vào sử dụng Gaussian Filter.

```
kernel_size = 5
blur = # ************** your code here ***********************
print("total intensities: ", sum(sum(blur)))
```

Question 2: Hãy cho biết kết quả sau khi thực hiện đoạn code ở bài tập 2

```
a) Total intensities is  162333
b) Total intensities is  159945
c) Total intensities is  182333
d) Total intensities is  192333
```

Bài tập 3: (Step 3: Edge detection) Hãy hoàn thiện đoạn code sau để phát hiện thông tin cạnh từ ảnh đầu vào sử dụng giải thuật Canny.

```
low_t = 50
```

```
high_t = 150
edges = # ******** your code here *****************
print("total intensities: ", sum(sum(edges)))
```

Question 3: Hãy cho biết kết quả sau khi thực hiện đoạn code ở bài tập 3

```
a) Total intensities is  162333
b) Total intensities is  159945
c) Total intensities is  318588
d) Total intensities is  192333
```

Bài tập 4: (Step 4: limit the search space using ego-lane information) Hãy hiện thực lại đoạn code sau để giới hạn vùng cần tìm kiếm cho việc xác định thông tin làn đường.

```
def limit_research_space(image):
  mask = np.zeros_like(image)
  ignore_mask_color = 255
  # creating a polygon to focus only on the road in the picture
  # we have created this polygon in accordance to how the camera was placed
  rows, cols = image.shape[:2]
  bottom_left = [cols * 0.1, rows * 0.95]
  top_left   = [cols * 0.4, rows * 0.6]
  bottom_right = [cols * 0.9, rows * 0.95]
  top_right = [cols * 0.6, rows * 0.6]
  vertices = np.array([[bottom_left, top_left, top_right, bottom_right]], dtype=np.
    int32)
  # filling the polygon with white color and generating the final mask
  cv2.fillPoly(mask, vertices, ignore_mask_color)
  # performing Bitwise AND on the input image and mask to get only the edges on the
    road
  masked_image = cv2.bitwise_and(image, mask)
  return masked_image
region = limit_research_space(edges)
print("total intensities: ", sum(sum(region)))
```

Question 4: Hãy cho biết kết quả sau khi thực hiện đoạn code ở bài tập 4

```
a) Total intensities is  162333
b) Total intensities is  159945
c) Total intensities is  318588
d) Total intensities is  104652
```

Bài tập 5: (Step 5: Hough transformation for detecting lines) Hãy hiện thực lại đoạn code sau để phát hiện làn trái và phải của làn đường cần phát hiện sử dụng kỹ thuật Hough Transform.

```
# Question 5: Use hough transform to detect left and right lane
def hough_transform(image):
  """
  Determine and cut the region of interest in the input image.
  Parameter:
    image: grayscale image which should be an output from the edge detector
  """
  # Distance resolution of the accumulator in pixels.
  rho = 1
  # Angle resolution of the accumulator in radians.
  theta = np.pi/180
  # Only lines that are greater than threshold will be returned.
  threshold = 20
  # Line segments shorter than that are rejected.
  minLineLength = 20
```

```
  # Maximum allowed gap between points on the same line to link them
  maxLineGap = 500
  # function returns an array containing dimensions of straight lines
  # appearing in the input image
  return cv2.HoughLinesP(image, rho = rho, theta = theta, threshold = threshold,
              minLineLength = minLineLength, maxLineGap = maxLineGap)

hough = hough_transform(region)
print("Dimensions of straight lines", hough.shape)
```

Question 5: Hãy cho biết kết quả sau khi thực hiện đoạn code ở bài tập 5

```
a) Dimensions of straight lines (17, 1, 4)
b) Dimensions of straight lines (18, 1, 4)
c) Dimensions of straight lines (19, 1, 4)
d) Dimensions of straight lines (20, 1, 4)
```

Bài tập 6: (Step 6: Draw left and right lanes to the screen) Hãy hiện thực lại đoạn code sau để hiện thị kết quả lane detection ra màn hình.

```

# Question 6: Draw the left and right lane to the screen

# Question 6: Draw the left and right lane to the screen

def average_slope_intercept(lines):
  """
  Find the slope and intercept of the left and right lanes of each image.
  Parameters:
    lines: output from Hough Transform
  """
  left_lines = [] #(slope, intercept)
  left_weights = [] #(length,)
  right_lines = [] #(slope, intercept)
  right_weights = [] #(length,)

  for line in lines:
    for x1, y1, x2, y2 in line:
      if x1 == x2:
        continue
      # calculating slope of a line
      slope = (y2 - y1) / (x2 - x1)
      # calculating intercept of a line
      intercept = y1 - (slope * x1)
      # calculating length of a line
      length = np.sqrt(((y2 - y1) ** 2) + ((x2 - x1) ** 2))
      # slope of left lane is negative and for right lane slope is positive
      if slope < 0:
        left_lines.append((slope, intercept))
        left_weights.append((length))
      else:
        right_lines.append((slope, intercept))
        right_weights.append((length))
  #
  left_lane = np.dot(left_weights, left_lines) / np.sum(left_weights) if len(
    left_weights) > 0 else None
  right_lane = np.dot(right_weights, right_lines) / np.sum(right_weights) if len(
    right_weights) > 0 else None
  return left_lane, right_lane


```

```
def pixel_points(y1, y2, line):
  """
  Converts the slope and intercept of each line into pixel points.
    Parameters:
      y1: y-value of the line's starting point.
      y2: y-value of the line's end point.
      line: The slope and intercept of the line.
  """
  if line is None:
    return None
  slope, intercept = line
  x1 = int((y1 - intercept)/slope)
  x2 = int((y2 - intercept)/slope)
  y1 = int(y1)
  y2 = int(y2)
  return ((x1, y1), (x2, y2))

def lane_lines(image, lines):
  """
  Create full lenght lines from pixel points.
    Parameters:
      image: The input test image.
      lines: The output lines from Hough Transform.
  """
  left_lane, right_lane = average_slope_intercept(lines)
  y1 = image.shape[0]
  y2 = y1 * 0.6
  left_line = pixel_points(y1, y2, left_lane)
  right_line = pixel_points(y1, y2, right_lane)
  return left_line, right_line


def draw_lane_lines(image, lines, color=[0, 255, 0], thickness=12):
  """
  Draw lines onto the input image.
    Parameters:
      image: The input test image (video frame in our case).
      lines: The output lines from Hough Transform.
      color (Default = red): Line color.
      thickness (Default = 12): Line thickness.
  """
  line_image = np.zeros_like(image)
  for line in lines:
    if line is not None:
      cv2.line(line_image, *line, color, thickness)
  return cv2.addWeighted(image, 1.0, line_image, 1.0, 0.0)

result = draw_lane_lines(image, lane_lines(image, hough))
cv2.imshow("result",result)
cv2.waitKey(0)
close_window_os()
print("Image result's resolution", result.shape)
```

Question 6: Hãy cho biết kết quả sau khi thực hiện đoạn code ở bài tập 6

```
a) Image result's resolutio  (704, 1279, 3)
b) Image result's resolutio  (804, 1279, 3)
c) Image result's resolutio  (904, 1279, 3)
d) Image result's resolutio  (604, 1279, 3)
```

Bài tập 7: (Lane detection for video) Hãy hiện thực giải thuật lane detection trên đoạn video cho sẵn (dataset.mp4).

```
#Question 7
#Lane detection for video image
import cv2
cap = cv2.VideoCapture("/dataset.mp4")

while True:
    # Capture frame-by-frame
    ret, frame = cap.read()
    # if frame is read correctly ret is True
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break

    # ********* your code here **********************
    if cv2.waitKey(1) == ord('q'):
        break

close_window_os()
```