自动驾驶-车道线检测

车道线检测算法

该项目的整体目标是将自动驾驶汽车中采集到的视频中的车道线检测分割出来。

Color Selection

对于自动驾驶汽车采集到的场景图像来看，车道线的颜色特征最为明显（在图像中是白色）。一般来说，图像是用三通道的形式表达出来（R、G、B）。下面就用代码来实现这种颜色选择

首先，我们导入三个常用的库：numpy、matplotlib中的image和pyplot

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
```

然后读取图片，并将图像中的信息拷贝出来，这里要注意的是：

python中的=是浅拷贝，所以要使用copy method

```
image = mpimg.imread("test.jpg")
print('This image is: ', type(image), 'with dimensions:', image.shape)

# Grab the x and y size and make a copy of the image.
ysize = image.shape[0]
xsize = image.shape[1]

color_select = np.copy(image)
```

现在我们来定义一个颜色阈值（三通道，经过调试后，在200的阈值下，可以较好的将场景中的白色线段筛选出来）

```
red_threshold = 200
green_threshold = 200
blue_threshold = 200
rgb_threashold = [red_threshold, green_threshold, blue_threshold]
```

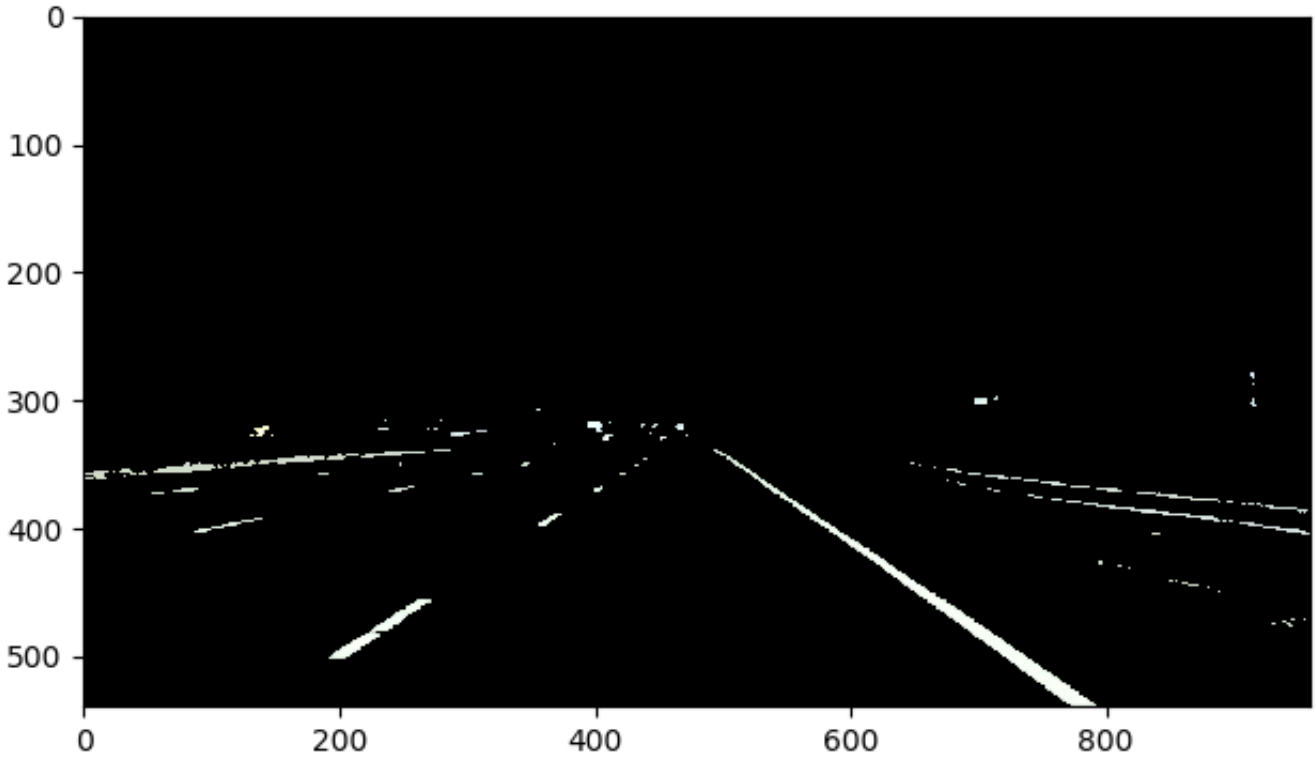
对图像中的像素值进行筛选，将小于阈值的部分都置为0

```
thresholds = (image[:, :, 0] < rgb_threshold[0]) \
             | (image[:, :, 1] < rgb_threshold[1]) \
             | (image[:, :, 2] < rgb_threshold[2])

color_select[thresholds] = [0, 0, 0]

plt.imshow(color_select)
plt.show()
```

这里，就可以得到一个检测车道线的图了~



Region Mask

在车辆的实际运行过程中，我们仅仅需要对当前地面上的车道线进行检测。这里，我们可以设置一个Region Mask，来固定住我们的待检测区域。
通过图像，我们可以发现，这里我们需要检测的区域可以用一个三角形表示出来，左端点，右端点和消影点。将三角形外的部分都滤除（像素变为[0, 0, 0]黑色），像素内的部分保留下来。代码如下：

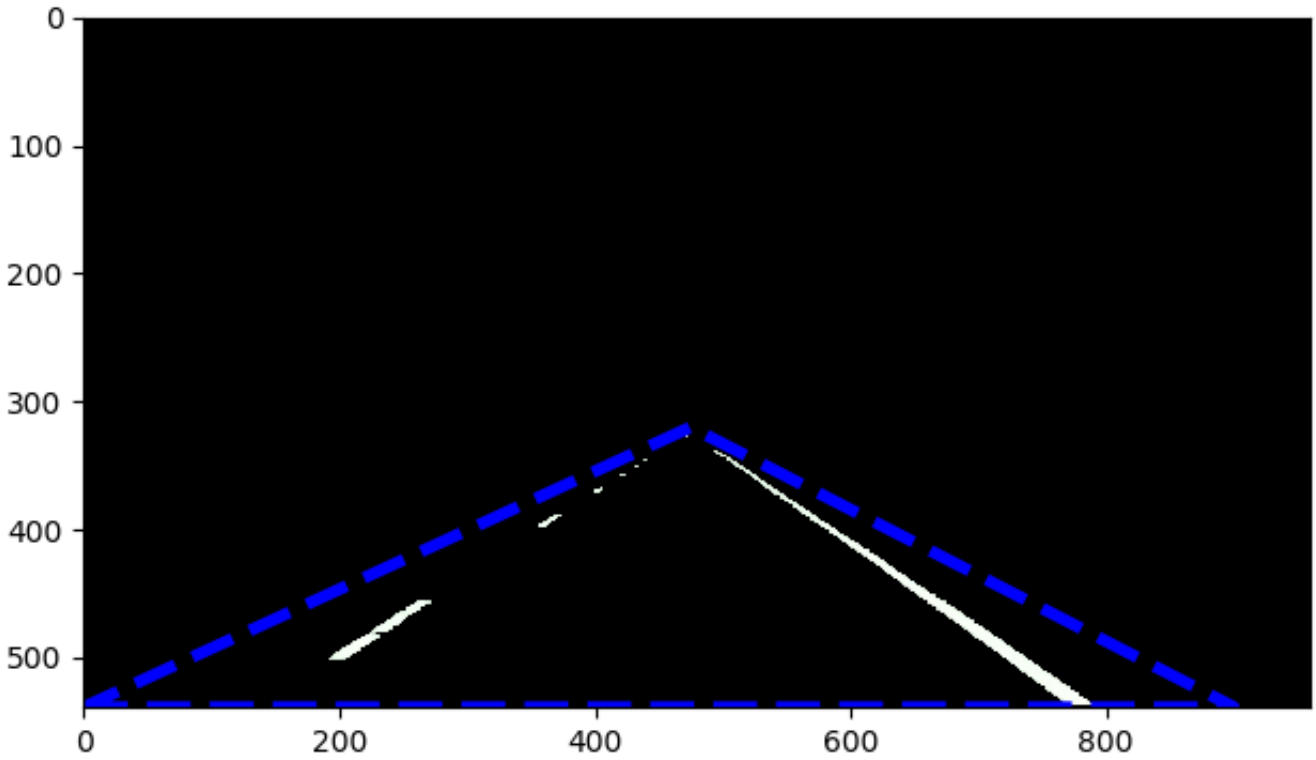
```
# 注意这里的像素坐标原点是图片的左上角。
left_bottom = [0, 539]
right_bottom = [900, 539]
apex = [475, 320]

fit_left = np.polyfit((left_bottom[0], apex[0]), (left_bottom[1], apex[1]),
1)
fit_right = np.polyfit((right_bottom[0], apex[0]), (right_bottom[1], apex[1]
), 1)
fit_bottom = np.polyfit((left_bottom[0], right_bottom[0]), (left_bottom[1],
right_bottom[1]), 1)
```

```
XX, YY = np.meshgrid(np.arange(0, xsize), np.arange(0, ysize))
region_thresholds = (YY > (XX * fit_left[0] + fit_left[1])) & (YY > (XX * fi
t_right[0]+  fit_right[1])) & (YY < (XX * fit_bottom[0] + fit_bottom[1]))

color_select[color_thresholds | ~region_thresholds] = [0, 0, 0]
x = [left_bottom[0], right_bottom[0], apex[0], left_bottom[0]]
y = [left_bottom[1], right_bottom[1], apex[1], left_bottom[1]]
plt.plot(x, y, 'b--', lw=4)
plt.imshow(color_select)
```

得到结果如图所示：



将整体结合，代码如下：

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np

# Read in the image and print out some stats
# Note: in the previous example we were reading a .jpg
# Here we read a .png and convert to 0,255 bytescale
```

```
image = mpimg.imread('/Users/robotics_qi/self-driving/image/lane.jpg')

# Grab the x and y size and make a copy of the image
ysize = image.shape[0]
xsize = image.shape[1]
color_select = np.copy(image)
line_image = np.copy(image)

# Define color selection criteria
# MODIFY THESE VARIABLES TO MAKE YOUR COLOR SELECTION
red_threshold = 200
green_threshold = 200
blue_threshold = 200

rgb_threshold = [red_threshold, green_threshold, blue_threshold]

# Define the vertices of a triangular mask.
# Keep in mind the origin (x=0, y=0) is in the upper left
# MODIFY THESE VALUES TO ISOLATE THE REGION
# WHERE THE LANE LINES ARE IN THE IMAGE
left_bottom = [0, 539]
right_bottom = [900, 539]
apex = [475, 320]

# Perform a linear fit (y=Ax+B) to each of the three sides of the triangle
# np.polyfit returns the coefficients [A, B] of the fit
fit_left = np.polyfit((left_bottom[0], apex[0]), (left_bottom[1], apex[1]),
1)
fit_right = np.polyfit((right_bottom[0], apex[0]), (right_bottom[1], apex[1]
), 1)
fit_bottom = np.polyfit((left_bottom[0], right_bottom[0]), (left_bottom[1],
right_bottom[1]), 1)

# Mask pixels below the threshold
color_thresholds = (image[:, :, 0] < rgb_threshold[0]) | \
                   (image[:, :, 1] < rgb_threshold[1]) | \
                   (image[:, :, 2] < rgb_threshold[2])

# Find the region inside the lines
XX, YY = np.meshgrid(np.arange(0, xsize), np.arange(0, ysize))
region_thresholds = (YY > (XX * fit_left[0] + fit_left[1])) & \
                    (YY > (XX * fit_right[0] + fit_right[1])) & \
                    (YY < (XX * fit_bottom[0] + fit_bottom[1]))

# Mask color and region selection
color_select[color_thresholds | ~region_thresholds] = [0, 0, 0]
# Color pixels red where both color and region selections met
line_image[~color_thresholds & region_thresholds] = [255, 0, 0]
```

```
# Display the image and show region and color selections
plt.imshow(image)
# x = [0, 900, 475, 0]
# y = [539, 539, 320, 539]
x = [left_bottom[0], right_bottom[0], apex[0], left_bottom[0]]
y = [left_bottom[1], right_bottom[1], apex[1], left_bottom[1]]
plt.plot(x, y, 'b--', lw=4)
plt.imshow(color_select)
plt.imshow(line_image)
plt.show()
```

最后结果如图所示：