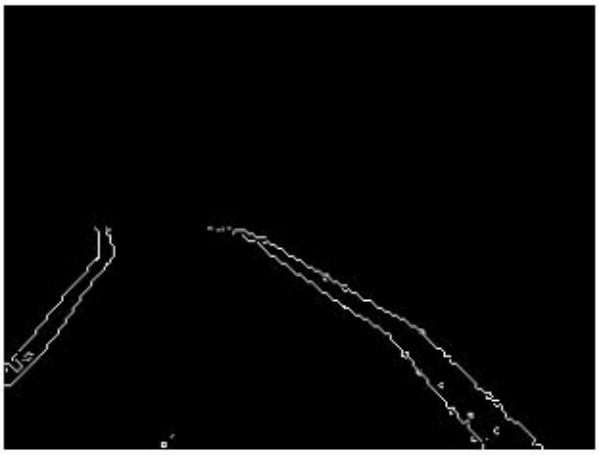
1. Import dependencies (cv2, and others you like)

```
In 1  1 import cv2
      2 from showimages import showrgb
      3 import numpy as np
```

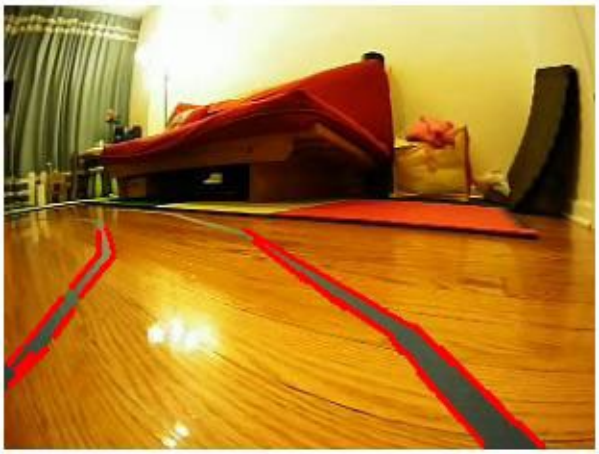
2. Convert the image into canny



3. Detect lines

Hint: cv2.HoughLinesP()

4. Draw the detected lines



5. Make the segmented lines into 2 separate ones (giving the solution here, but please read through it, there are comments to guide you)

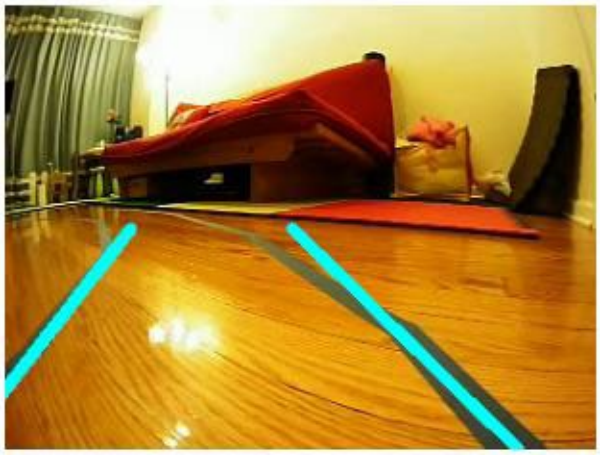
```
def make_points(img, line):
    h, w, _ = img.shape
    slope, intercept = line
    y1 = h # bottom of image
    y2 = y1 // 2
    x1 = max(-w, min(2 * w, int((y1 - intercept) / slope))) # (y1 -
intercept) / slope is solving y = mx + c. Max and min
                                                            # are used
to make sure that x1 is within bounds of the frame.
                                                            # while -
width is technically outside of the frame, it acts as a
                                                            # "safety
buffer" to ensure that any part of the line segment
                                                            # that
falls outside of the frame is clipped and not drawn.
    x2 = max(-w, min(2 * w, int((y2 - intercept) / slope)))
    return [[x1, y1, x2, y2]]

def average_slope_intercept(img, line_segments):
    lines = []
    if line_segments is None:
        print("No line segments detected.")
        return lines

    h, w, _ = img.shape
    left_fit = []
    right_fit = []

    boundary = 1 / 3 # left line and right line split from right
    left_region_boundary = w * (1 - boundary)
    right_region_boundary = w * boundary

    for line_segment in line_segments:
        for x1, y1, x2, y2 in line_segment:
            if x1 == x2:
                continue # avoid division by zero
            fit = np.polyfit((x1, x2), (y1, y2), 1) # np.polyfit
returns the slope and y-intercept of a line given points
                                                    # 1 is the degree
of the polynomial
            slope = fit[0]
            intercept = fit[1]
            if slope < 0:
                if x1 < left_region_boundary and x2 <
left_region_boundary: # both points on left side
                    left_fit.append((slope, intercept))
            else:
                if x1 > right_region_boundary and x2 >
right_region_boundary: # both points on right side
                    right_fit.append((slope, intercept))
    left_fit_average = np.average(left_fit, axis=0) # average of slope
and intercept
    if len(left_fit) > 0:
        lines.append(make_points(img, left_fit_average))
```

```
    right_fit_average = np.average(right_fit, axis=0) # average of
slope and intercept
    if len(right_fit) > 0:
        lines.append(make_points(img, right_fit_average))
    return lines
```

6. Putting it all together

```
def detect_lanes(img):
    img_canny = get_canny(img)
    img_lines = get_lines(img_canny)
    img_lines = average_slope_intercept(img, img_lines)
    return img_lines
```

7. Draw the two lines obtained



8. Get the middle point of the lines