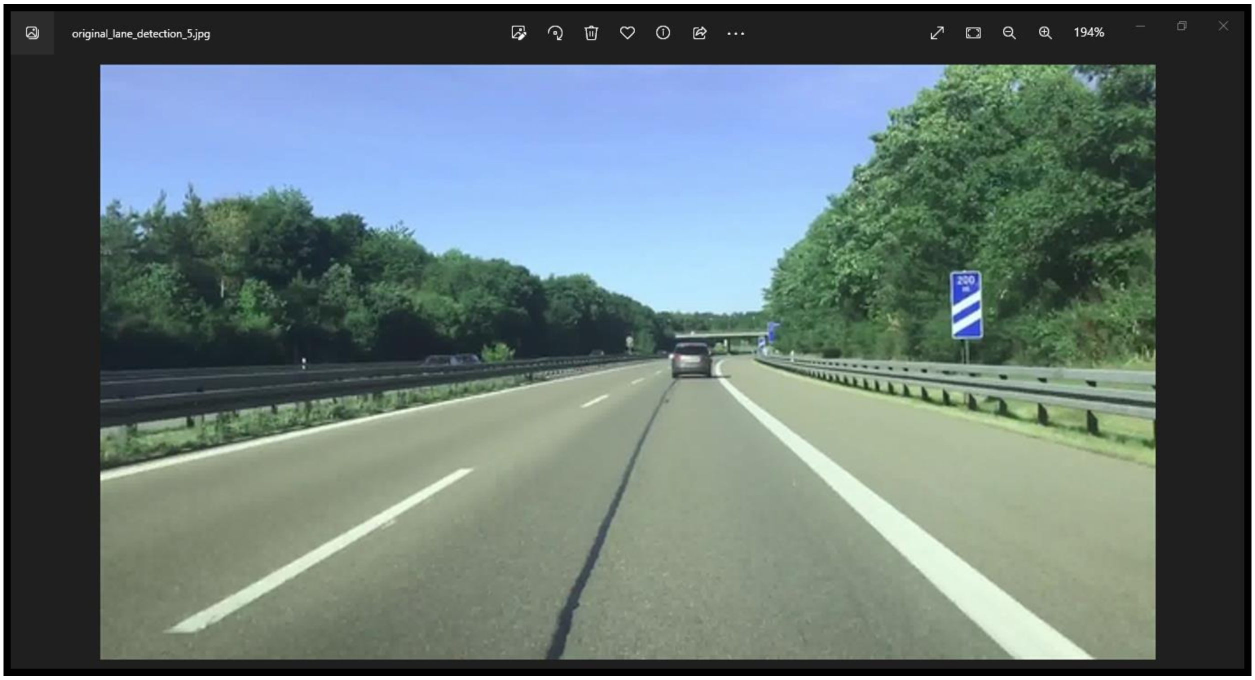
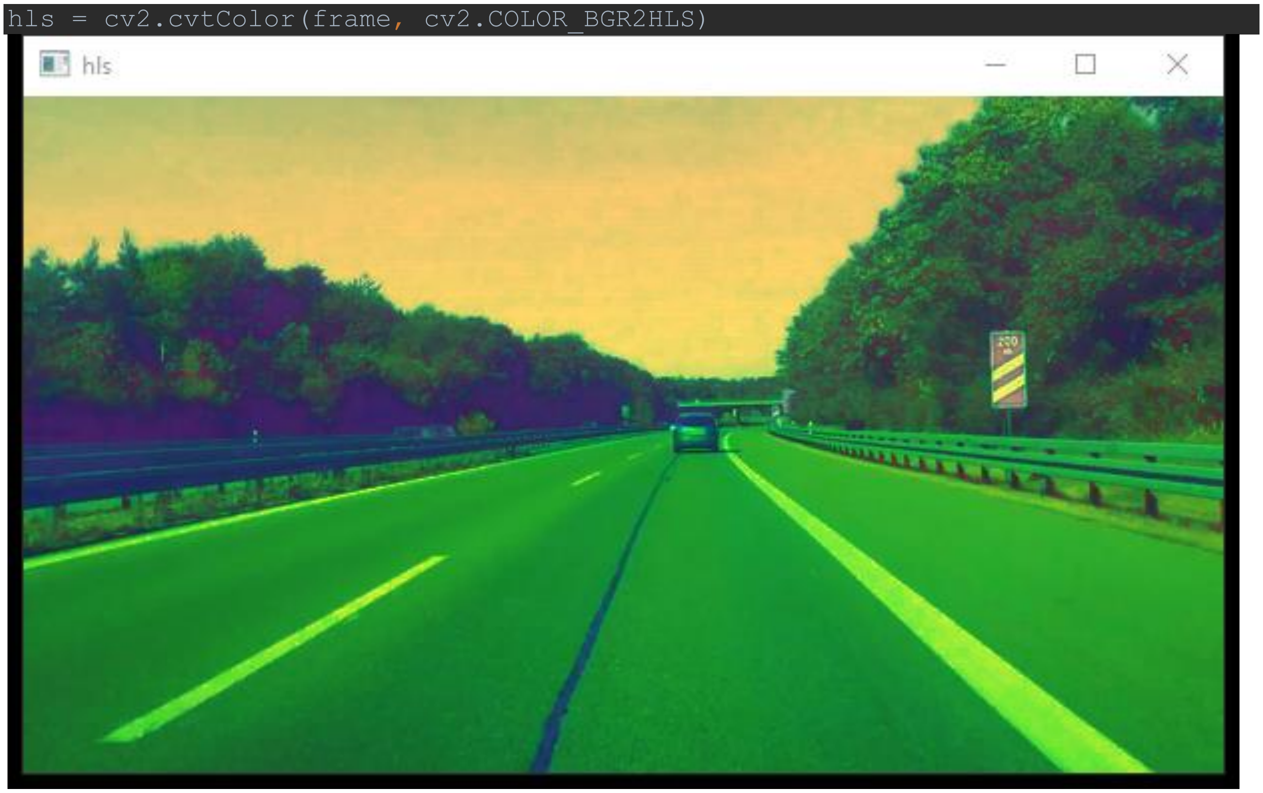
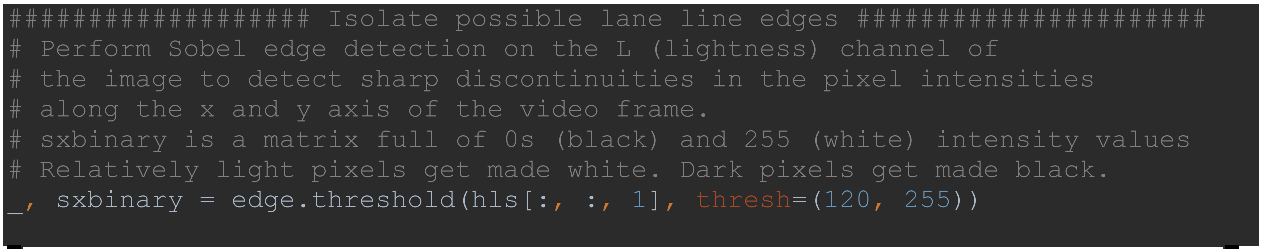
Input Image

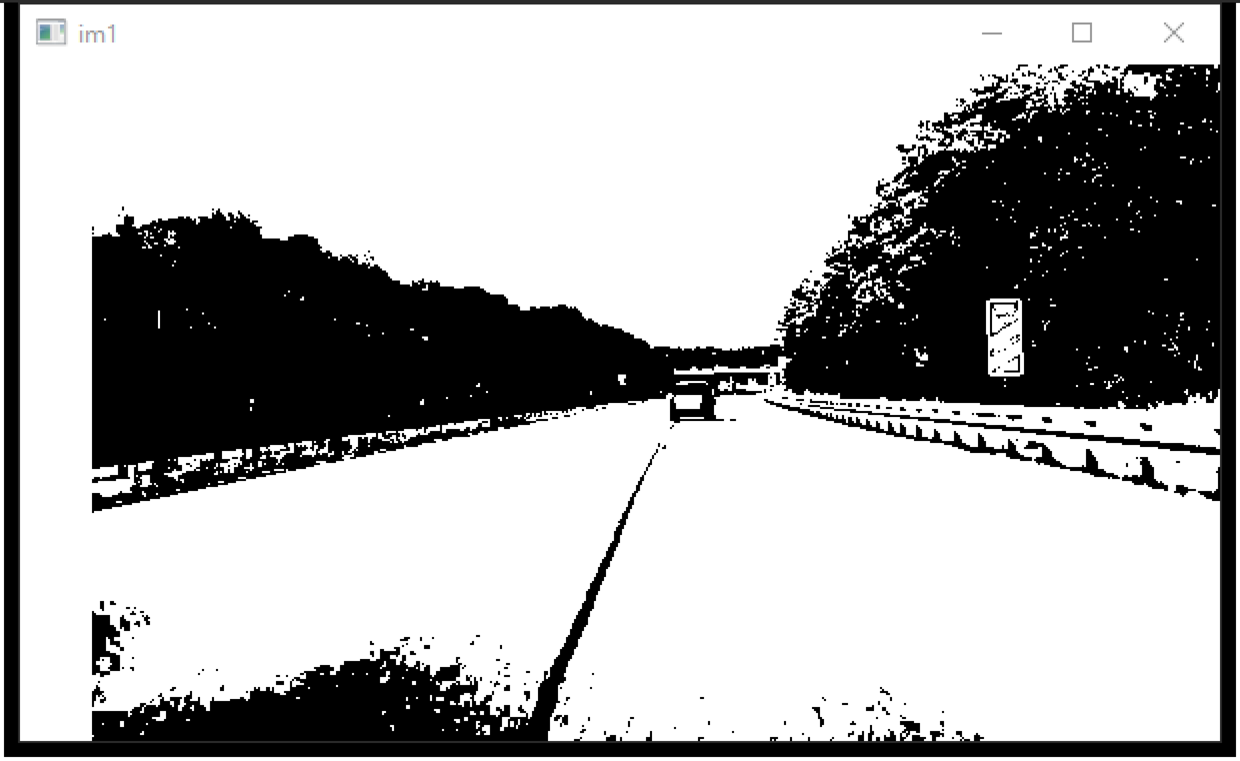


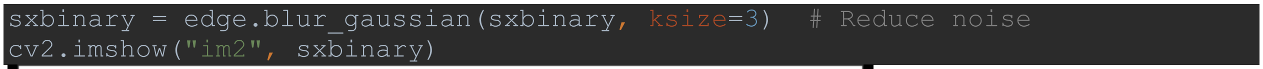
```
hls = cv2.cvtColor(frame, cv2.COLOR_BGR2HLS)
```



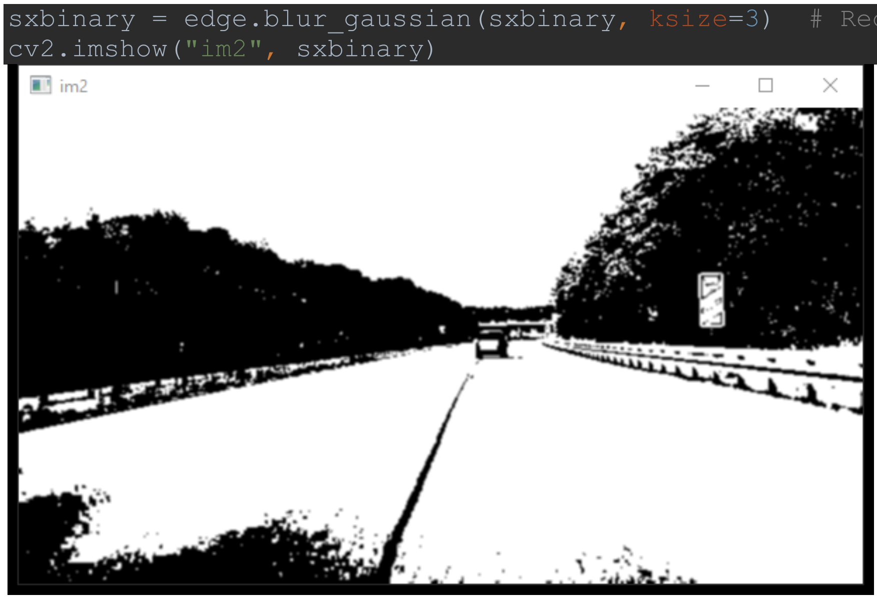


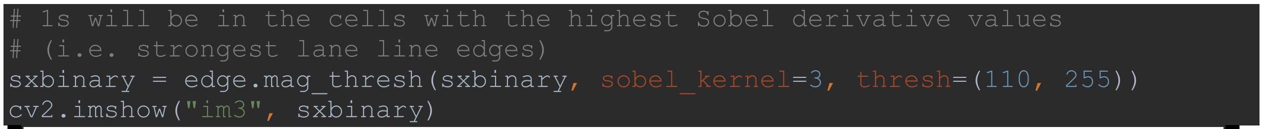
```
################### Isolate possible lane line edges ######################
# Perform Sobel edge detection on the L (lightness) channel of
# the image to detect sharp discontinuities in the pixel intensities
# along the x and y axis of the video frame.
# sxbinary is a matrix full of 0s (black) and 255 (white) intensity values
# Relatively light pixels get made white. Dark pixels get made black.
_, sxbinary = edge.threshold(hls[:, :, 1], thresh=(120, 255))
```



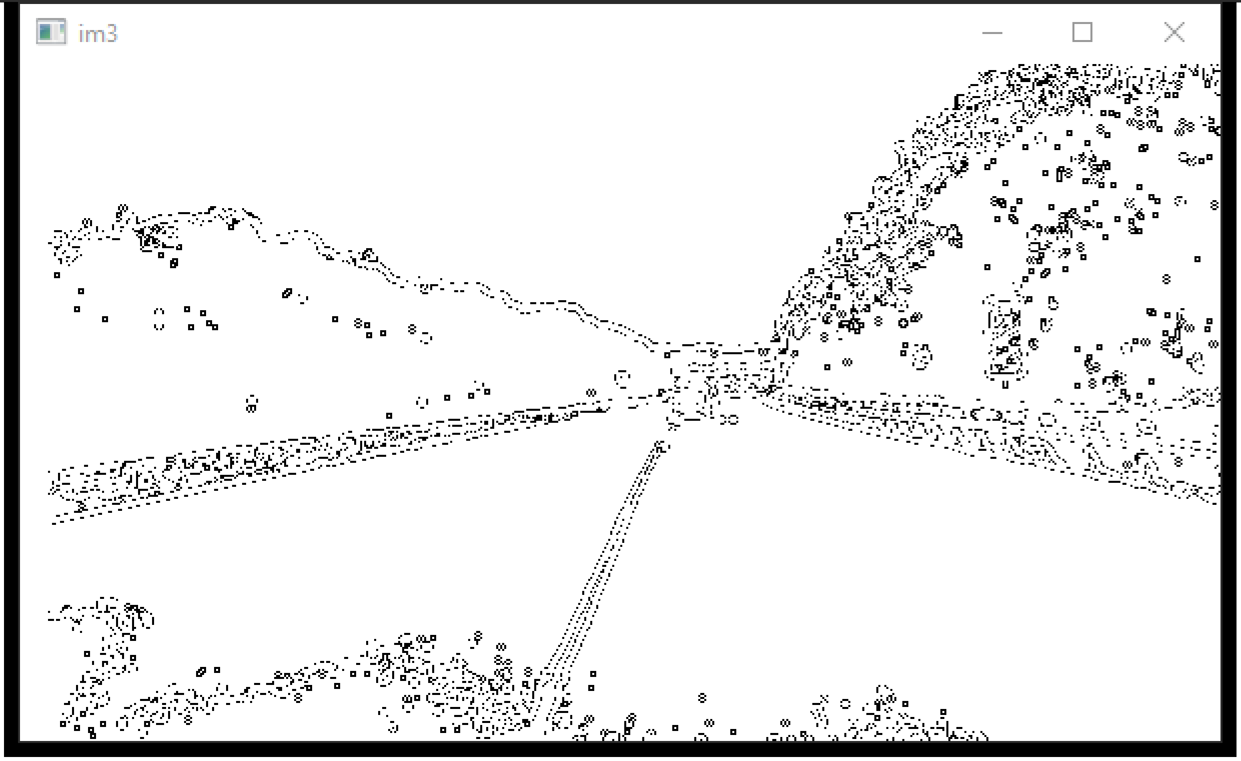


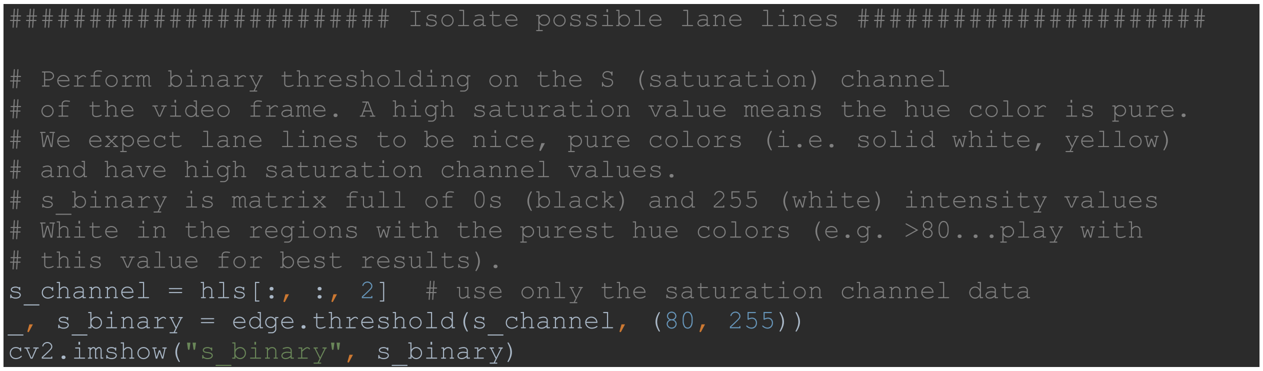
```
sxbinary = edge.blur_gaussian(sxbinary, ksize=3)  # Reduce noise
cv2.imshow("im2", sxbinary)
```



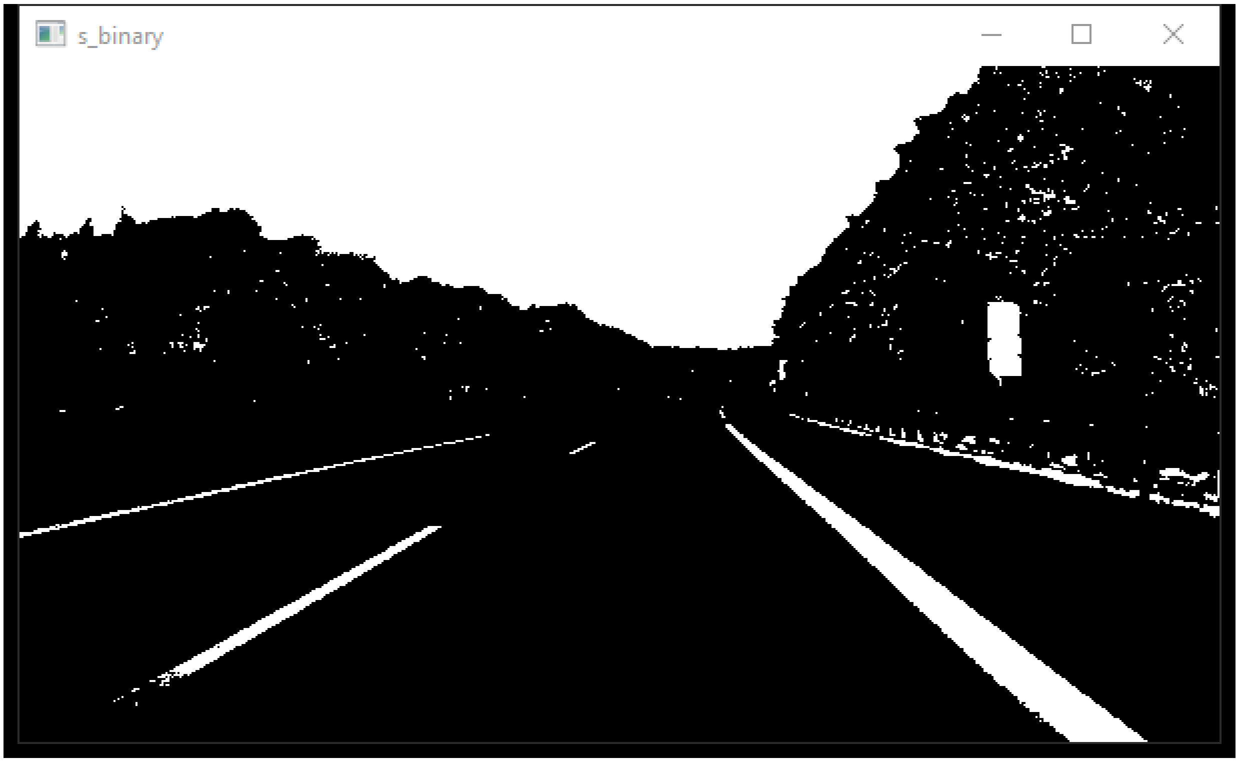


```
# 1s will be in the cells with the highest Sobel derivative values
# (i.e. strongest lane line edges)
sxbinary = edge.mag_thresh(sxbinary, sobel_kernel=3, thresh=(110, 255))
cv2.imshow("im3", sxbinary)
```

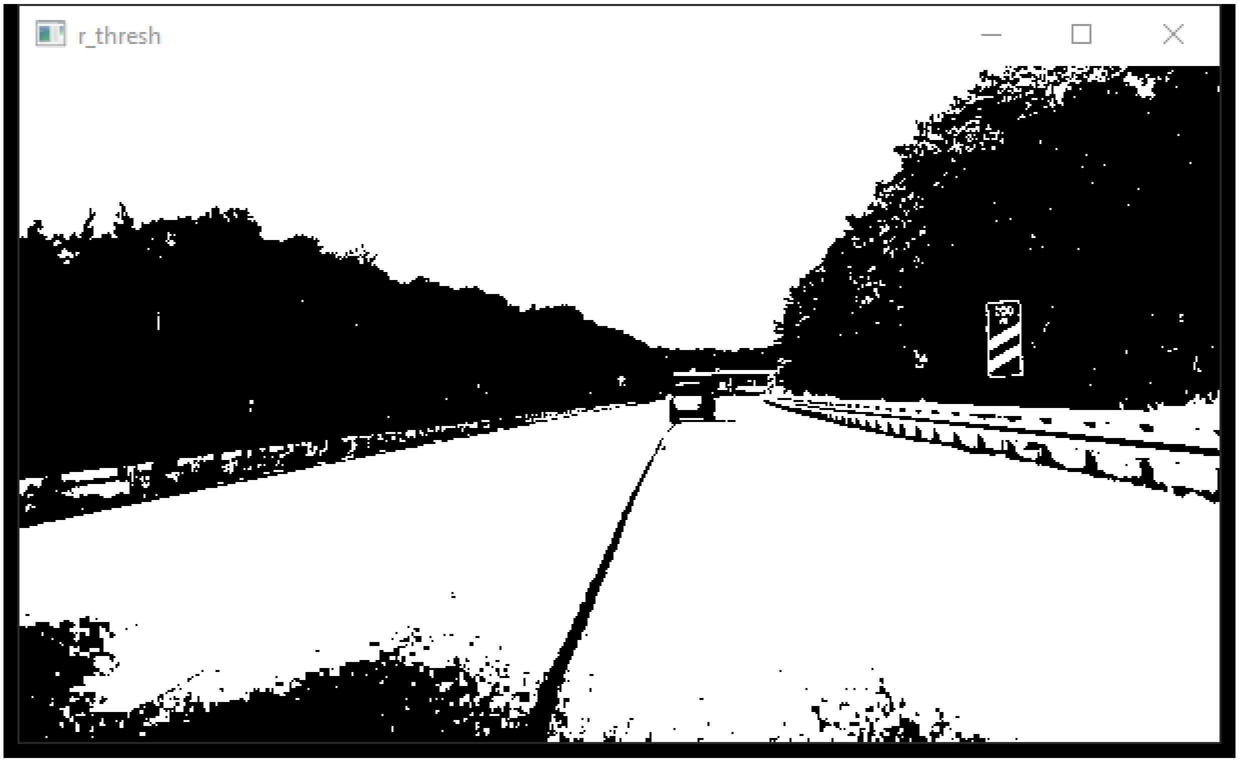




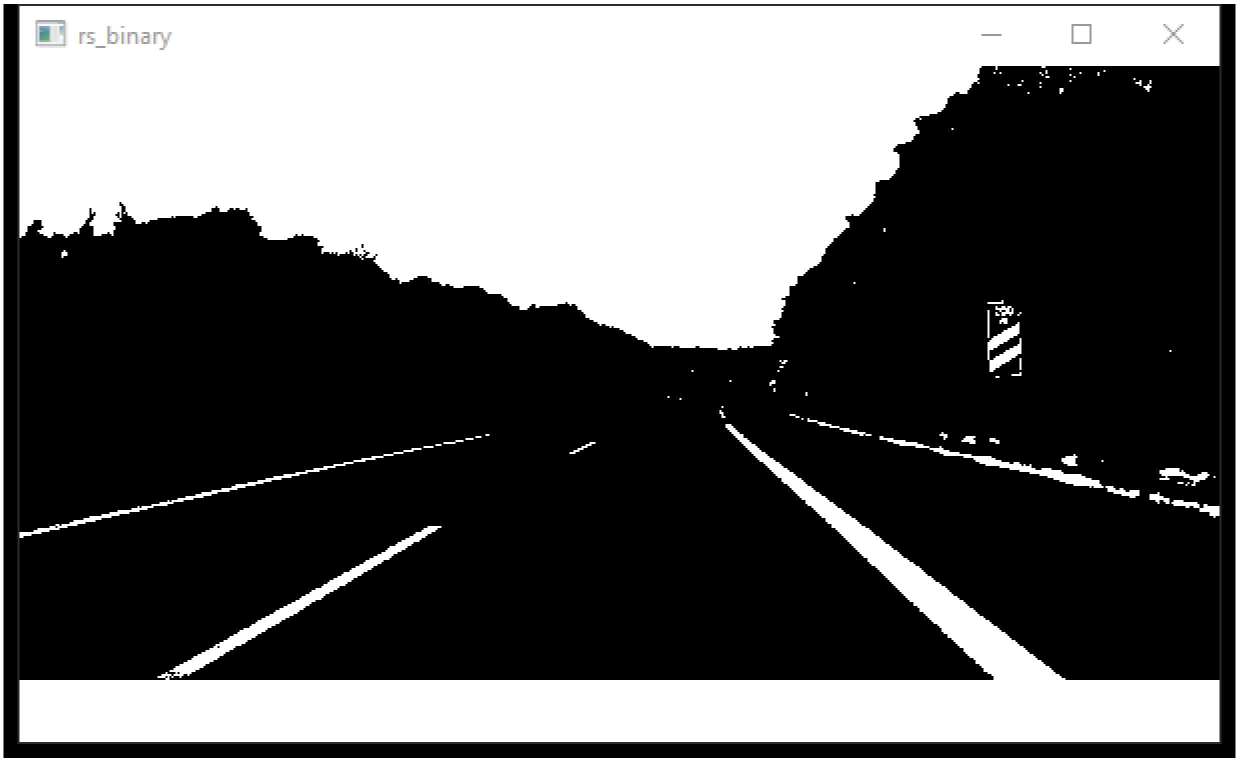
```
######################## Isolate possible lane lines ######################

# Perform binary thresholding on the S (saturation) channel
# of the video frame. A high saturation value means the hue color is pure.
# We expect lane lines to be nice, pure colors (i.e. solid white, yellow)
# and have high saturation channel values.
# s_binary is matrix full of 0s (black) and 255 (white) intensity values
# White in the regions with the purest hue colors (e.g. >80...play with
# this value for best results).
s_channel = hls[:, :, 2]  # use only the saturation channel data
_, s_binary = edge.threshold(s_channel, (80, 255))
cv2.imshow("s_binary", s_binary)
```



```
# Perform binary thresholding on the R (red) channel of the
# original BGR video frame.
# r_thresh is a matrix full of 0s (black) and 255 (white) intensity values
# White in the regions with the richest red channel values (e.g. >120).
# Remember, pure white is bgr(255, 255, 255).
# Pure yellow is bgr(0, 255, 255). Both have high red channel values.
_, r_thresh = edge.threshold(frame[:, :, 2], thresh=(120, 255))
cv2.imshow("r_thresh",r_thresh)
```



```
# Lane lines should be pure in color and have high red channel values
# Bitwise AND operation to reduce noise and black-out any pixels that
# don't appear to be nice, pure, solid colors (like white or yellow lane
# lines.)
rs_binary = cv2.bitwise_and(s_binary, r_thresh)
cv2.imshow("rs_binary", rs_binary)
```



```
lane_line_markings = cv2.bitwise_or(rs_binary, sxbinary.astype(
    np.uint8))
```