**Detector de carril.**

El objetivo del detector de carril es detectar, únicamente, los carriles de una carretera a partir de la imagen original de la cámara de un coche, se usan librerías de openCV para cumplir este objetivo. Se puede ver con claridad la imagen “RGB Image” de la figura 1, que las líneas son de color blanco, se puede aprovechar esta propiedad para segmentar los carriles, ya que son los unicos que tienen la mayor cantidad de pixeles juntos de color blanco que cualquier otro objeto.

Para comenzar, usamos la imagen original de la cámara con la que puedes realizar cualquier tipo de programa que tenga que ver con la información que está provee. La imagen original es “BGR Image” de la figura 1.

La imagen “Binary image” muestra en color blanco todos los pixeles de la imagen original que son de color blanco y en negro todo lo que no es de color blanco. Para segmentar únicamente este color, es necesario usar la función cv2.inRange(original\_image, lower\_white, upper\_white) en la cual indicamos la imagen original, el umbral máximo y mínimo en el que se encuentra el color blanco.

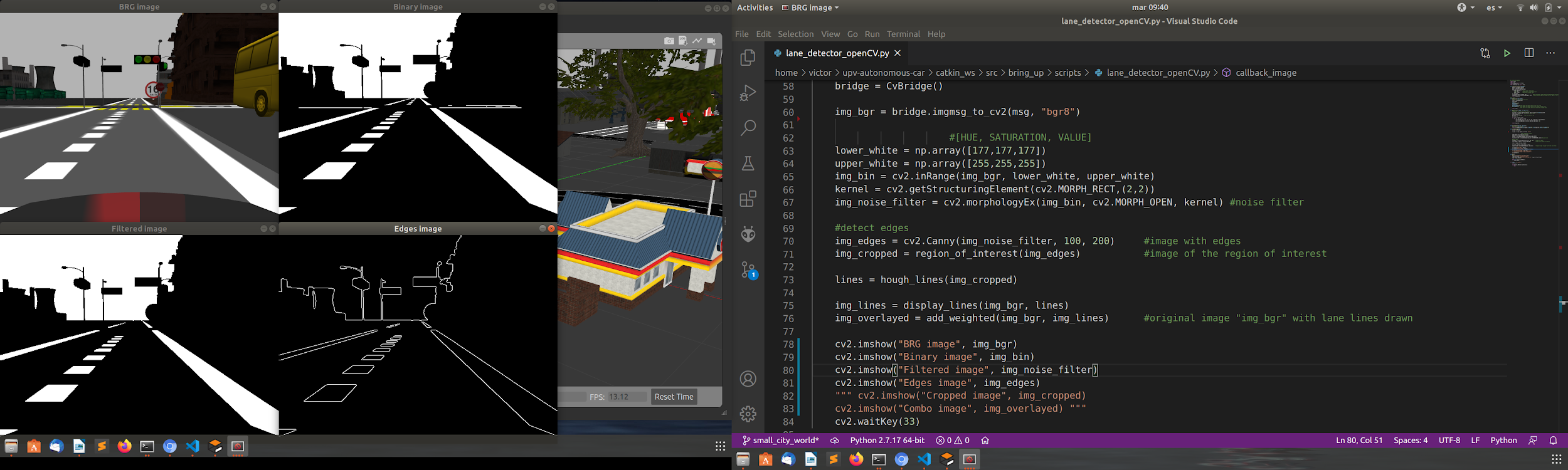


Figura 1. “BGR Image”,imagen original que provee la cámara. “Binary Image”, imagen binaria en blanco y negro. “Filtered Image”, imagen binaria con filtro de ruido. “Edges Image”, imagen de contornos.

Después de hacer algunas pruebas con herramientas web como “color picker” para saber el código del color blanco en hsv que nos proporciona la imagen de la cámara, se descubrió que en algunas partes de la carretera el color de la división de los carriles no es perfectamente blanco, es un color que tiende a ser gris, por lo que es necesario ajustar el umbral mínimo, pues el máximo es completamente blanco. De acuerdo a la imagen original, se llegó a la conclusión de que el valor de los umbrales debe ser el siguiente:

H S V

Threshold = [HUE, SATURATION, VALUE]

Lower\_white = [177,177,177] gris claro

Upper\_white = [255,255,255] blanco

Para continuar con la segmentación de los carriles, es recomendable aplicar un filtro a la imagen “Binary image” con el objetivo de despreciar aquellos pixeles color blanco cuya densidad es mínima, es decir, despreciar pixeles blancos que están muy dispersos entre sí. En la imagen “Filtered image” de la figura 1 se puede ver el resultado de haber aplicado el filtro, las líneas de los carriles horizontales ya no aparecen en la imagen filtrada, ya que la cantidad de pixeles blancos que están juntos es mucho menor que la de los pixeles de los carriles verticales. Para obtener la imagen filtrada puedes usar la función cv2.morphologyEx(binary\_image, cv2.MORPH\_OPEN, kernel). Es necesario que en el primer parámetro de la función indiques la imagen que será filtrada y que en los dos últimos uses los datos que se recomiendan en la documentación de openCV.

Con el fin de identificar los carriles por medio de líneas, el primer paso es obtener los contornos de la imagen filtrada, el contorno nos proporciona información más próxima a una línea. Para obtener la imagen de contornos “Edges image” de la figura 1, es necesario usar la función cv2.Canny(filtered\_image, threshold1, threshold2). Se requiere indicar la imagen filtrada y los umbrales correspondientes. Después de hacer algunas pruebas, se llegó a la conclusión de que los umbrales recomendados son:

Threshold1 = 100

Threshold2 = 200

Si cambiamos estos umbrales aumentando o disminuyendo el valor recomendado, el comportamiento tiende a ser el siguiente. Figura 2.



Figura 2. Parametro Threshold1 > 100 o Threshold1 < 100. Threshold2 >200 o

Threshold2 < 200

Se puede ver en la figura 2 que aparecen más contornos de color blanco y además comienzan a distorsionar el contorno original de la división de los carriles.

Podemos ver que desde la posición en la que se encuentra el coche, es posible interpretar la perspectiva y ver que el camino termina en un punto, formando un triángulo como el que se muestra en la figura 3.



Figura 3. Perspectiva de inicio y final de la carretera en forma de triángulo.

Lo que realmente interesa es segmentar los carriles de la carretera, entonces se puede aprovechar esta geometría para reducir el campo de visión de estudio y obtener únicamente la imagen en la que aparecen los carriles. Observa la figura 4.

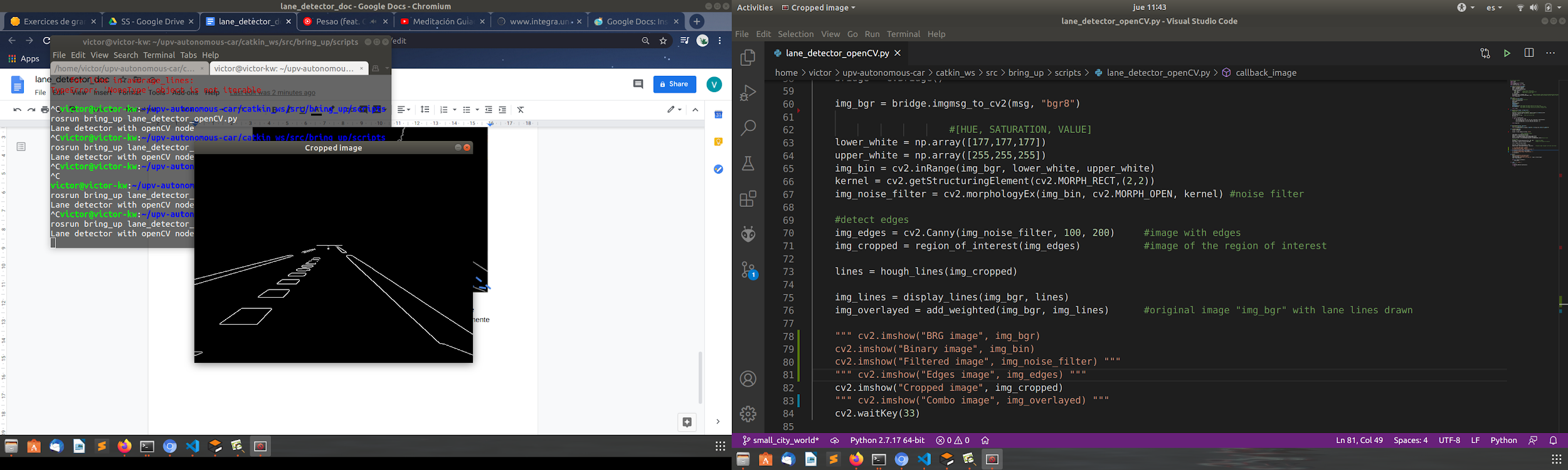


Figura 4. Imagen de los contornos de la sección de interés.

En la figura 4 se muestra la región de interés segmentada por un triángulo cuyos puntos están definidos en pixeles por coordenadas en el eje “x” y “y” de la imagen original. Los puntos con los que se construye el triángulo se definen de forma experimental de acuerdo a las dimensiones de la imagen, basta con indicar 3 de ellos para construirlo. Se puede ver únicamente la imagen contenida en el interior del triángulo y se descarta todo lo que está en el exterior.

Esta imagen es la que necesitamos para calcular y dibujar las líneas de los carriles sobre la imagen original “BGR image”. Lo primero que hay que hacer, es calcular las líneas por medio de la función cv2.HoughLinesP(). Los parámetros que se usan son los siguientes:

cv2.HoughLinesP(

cropped\_image, #imagen con la región de interés

rho = 1, #valor recomendado en la documentación

theta = 𝝅/180, #valor recomendado en la documentación

threshold =60, #valor recomendado por experimentación

lines = None, #valor recomendado en la documentación

minLineLength = 175, #valor recomendado por experimentación

maxLineGap = 150 #valor recomendado por experimentación

)

Cuando aumentamos el valor del parámetro “threshold” el número de líneas calculadas disminuye, es decir, si dibujamos todas las líneas calculadas por la función cv2.HoughLinesP() con un valor pequeño de “threshold”, por ejemplo 2, el número de líneas es mayor que si usamos un valor igual a 200. Se recomienda un valor de 60 para que el número de líneas sea parecido al número de divisiones de los carriles que hay en la imagen. En la figura 5 se muestra la diferencia entre un valor de “threshold” igual a 2, 60 y 200, respectivamente.

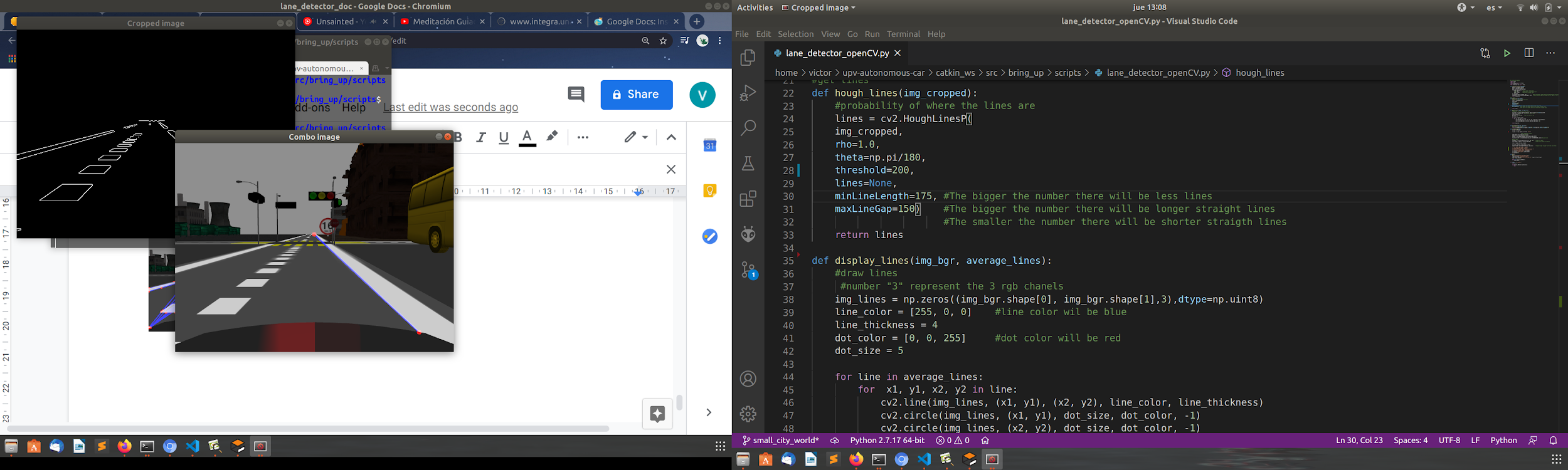
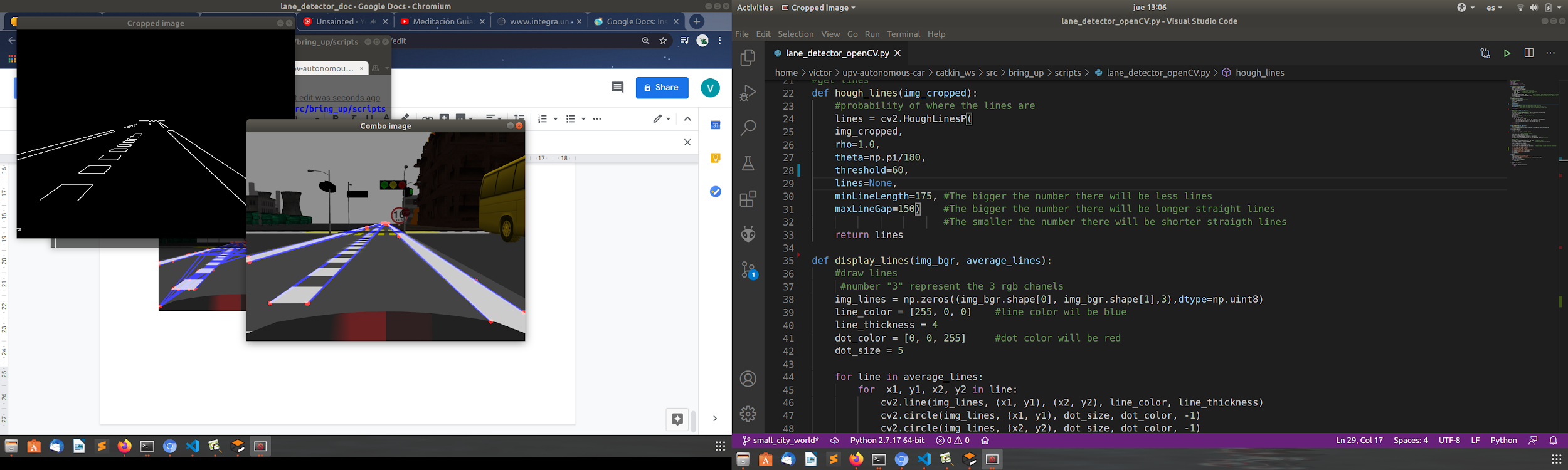


Figura 5. Uso de los valores del parámetro “threshold” de 2, 60 y 200 respectivamente.

Cuando aumentamos el valor del parámetro “minLineLength” el número de líneas disminuye. Entre más pequeño sea este valor, el número de líneas aumenta. Tienen un comportamiento similar al del parámetro “threshold”. El valor que se ajusta bien en cuanto al número de líneas dibujadas y el número de líneas de división de carriles es de 175.

En cuanto al parámetro “maxLineGap”, entre menor sea el valor, las líneas tienden a se más largas, entonces solo se calculan y dibujan líneas largas. Si el valor es grande solo se calculan y dibujan líneas cortas. Este parámetro también afecta el número de líneas, es necesario encontrar un equilibrio entre longitud y número de líneas. Tenga en cuenta que cada línea se dibuja con respecto a dos puntos con coordenadas “x” y “y” cada uno. Vea la diferencia entre usar un valor de “maxLineGap” de 10, 150 y 500 en la figura 6.

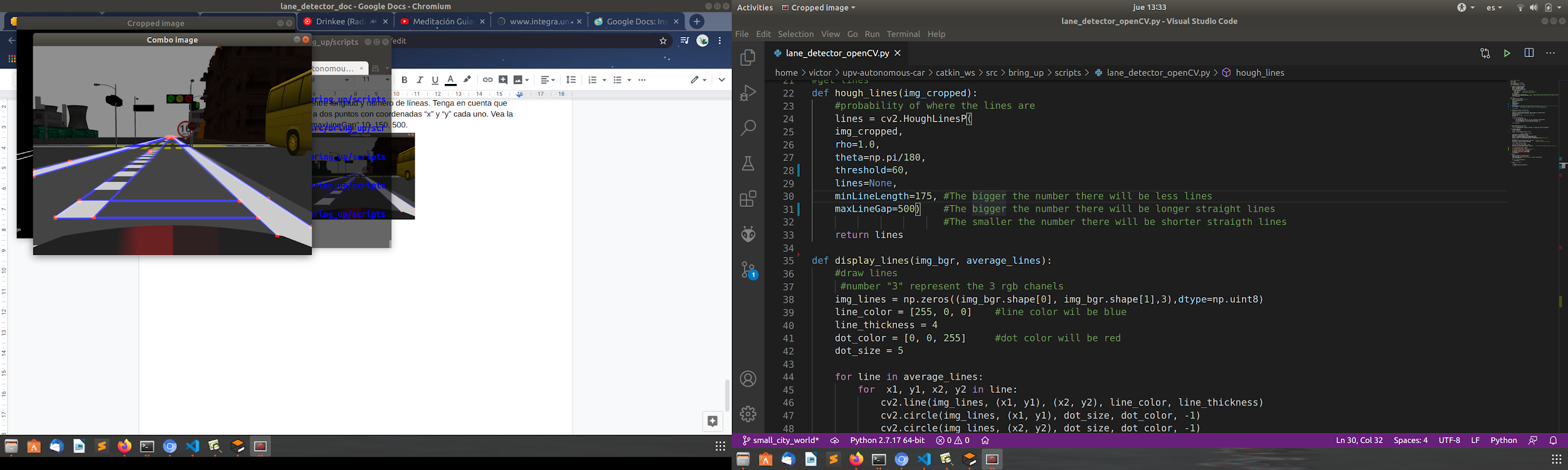
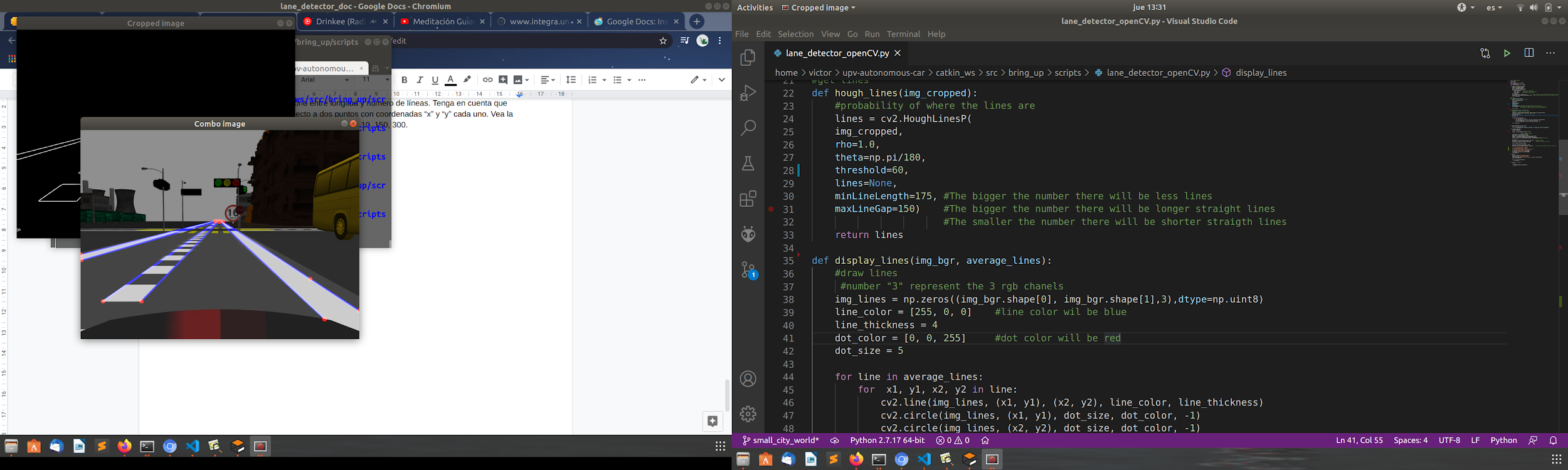
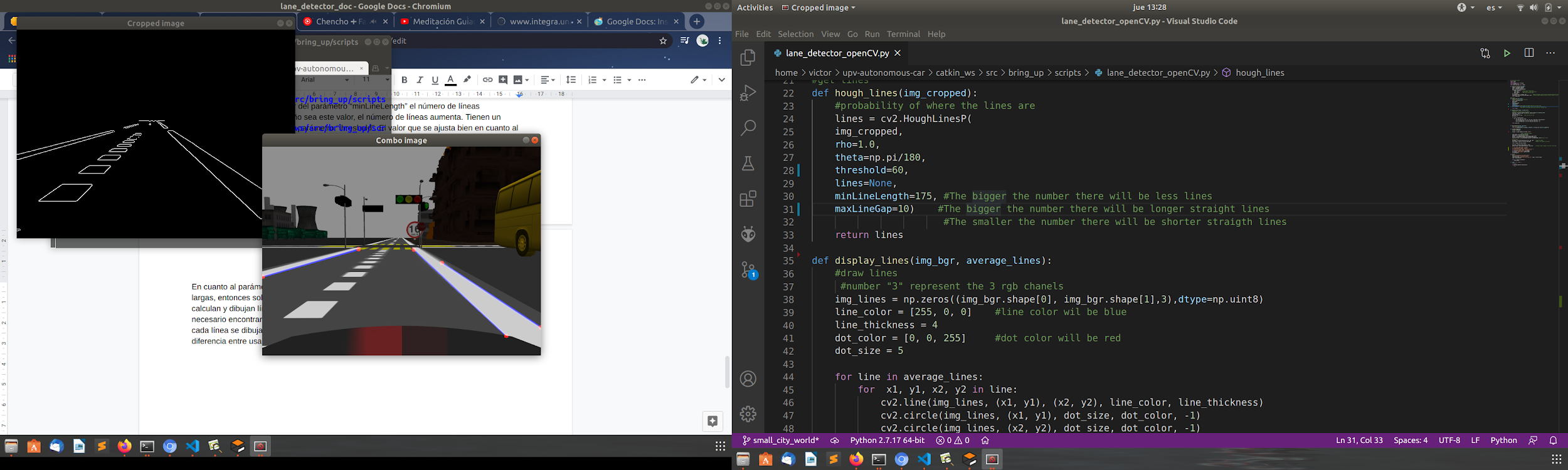


Figura 6. Uso de los valores del parámetro “maxLineGap” de 10, 150 y 500, respectivamente.

Se recomienda usar un valor de 150 para que la longitud y el número de líneas se ajuste al número de líneas de división de los carriles.

Con el procedimiento anterior y usando los valores recomendados, el resultado final es el siguiente:

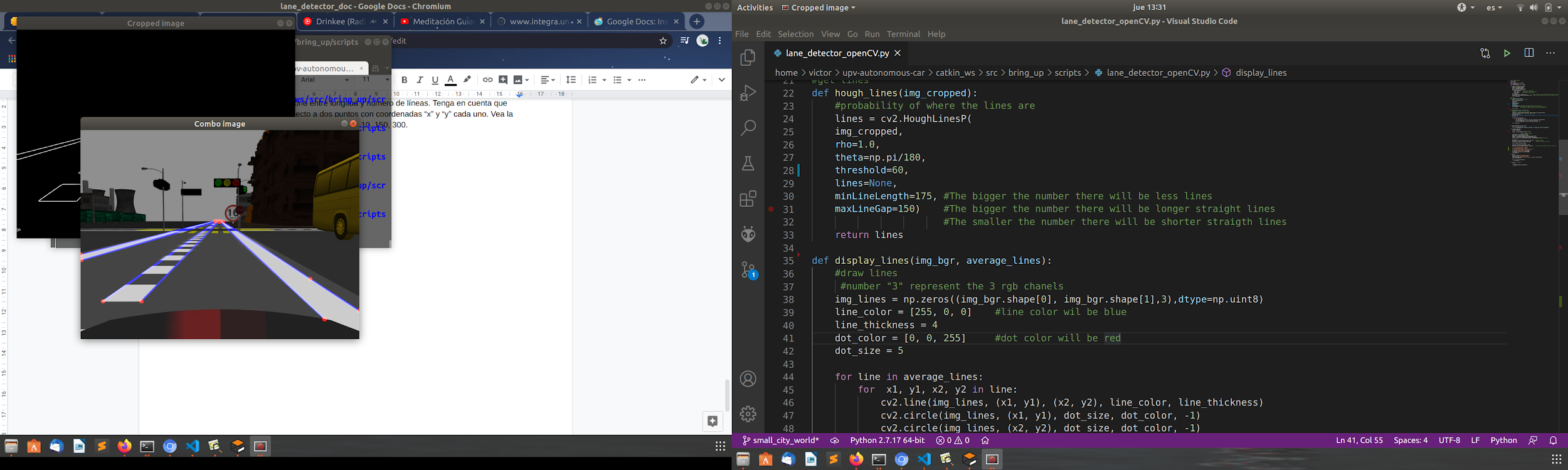


Figura 7. Segmentación de los carriles de la carretera.