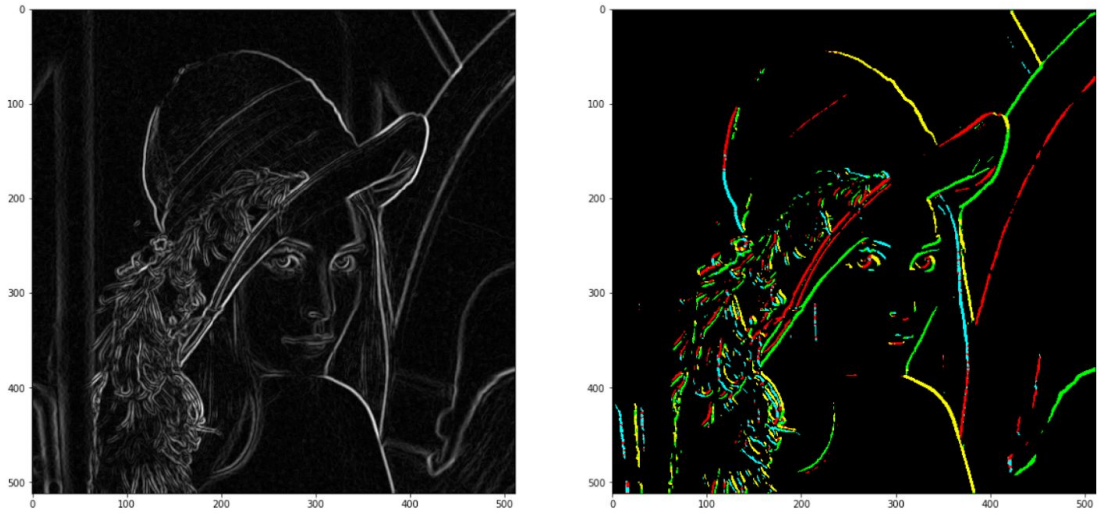
In [11]:

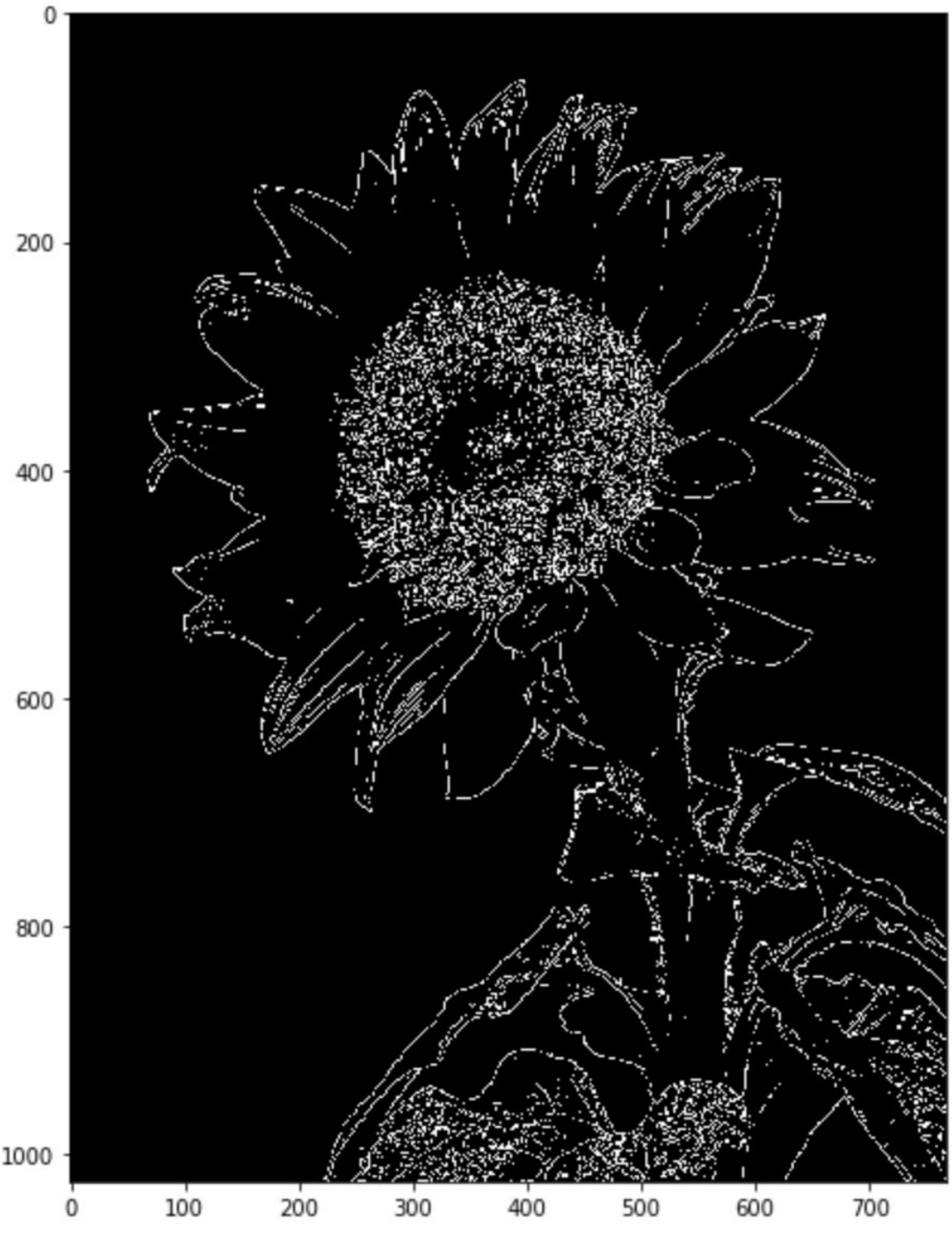
```
f, (ax3, ax4) = plt.subplots(1, 2, figsize=(20,10))

gradient_mag = ((dx)**2 + (dy)**2)**0.5
ax3.imshow(gradient_mag, cmap='gray')

orien = cv2.phase(dy, dx, angleInDegrees = True)
_, mask = cv2.threshold(gradient_mag, 150, 255, cv2.THRESH_BINARY)
image_map = np.zeros((orien.shape[0], orien.shape[1], 3), dtype=np.int16)

red = np.array([255, 0, 0])
cyan = np.array([0, 255, 255])
green = np.array([0, 255, 0])
yellow = np.array([255, 255, 0])

image_map[(mask == 255) & (orien <= 90)] = red
image_map[(mask == 255) & (orien > 90) & (orien <= 180)] = cyan
image_map[(mask == 255) & (orien > 180) & (orien <= 270)] = green
image_map[(mask == 255) & (orien > 270) & (orien <= 360)] = yellow

ax4.imshow(image_map)
```

Out[11]: <matplotlib.figure.AxesImage at 0x1df732bb6a0>

1.

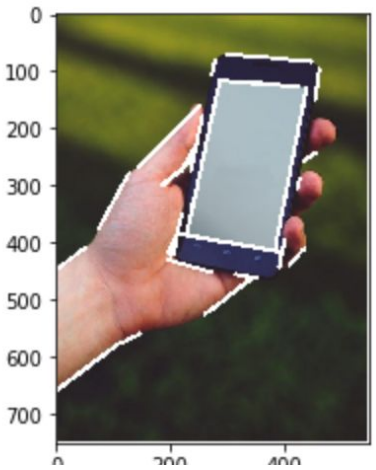


2.



```
rho = 2
theta = np.pi/180
threshold = 50
min_line_length = 10
max_line_gap = 20

# Run Hough on the edge-detected image, get coordinates of lines
lines = cv2.HoughLinesP( edges, rho, theta, threshold,  np.array([]), 10, 20)
# Iterate over the output "lines" and draw lines on the image copy

for line in lines:
    for x1,y1,x2,y2 in line:
        cv2.line(line_image,(x1,y1),(x2,y2),(255,255,255),5)

plt.imshow(line_image)

## TODO
#Try line detection on different images
```

```
<matplotlib.image.AxesImage at 0xb31b5b0>
```

3. 

```
rho = 1
theta = np.pi/180
threshold = 200
min_line_length = 100
max_line_gap = 5

# Run Hough on the edge-detected image, get coordinates of lines
hall_lines = cv2.HoughLinesP( edges_hall, rho, theta, threshold,  np.array([]), min_line_length, max_line_gap)
# Iterate over the output "lines" and draw lines on the image copy

for line in hall_lines:
    for x1,y1,x2,y2 in line:
        cv2.line(hall_image,(x1,y1),(x2,y2),(255,255,255),5)

plt.imshow(hall_image)
```

```
<matplotlib.image.AxesImage at 0x268467f0>
```