Image Segmentation

June 17, 2022

1 Image Segmentation

This notebook documents my exploration of basic image segmentation techniques. I will be working specifically on instance segmenatio, as opposed to semantic segmentation. This is a recreation of the article found here.

1.1 Threshold Segmentation

We will first work to divide the image into two regions, namely the background and the foreground by choosing one threshold value, n. Every pixel with a brightness value $p >= n$ will be marked black, and $p < n$ will be turned white.

First, we will import some libraries.

[1]:
```
from skimage.color import rgb2gray
import numpy as np
import cv2
import matplotlib.pyplot as plt
import os

%matplotlib inline

from scipy import ndimage
```

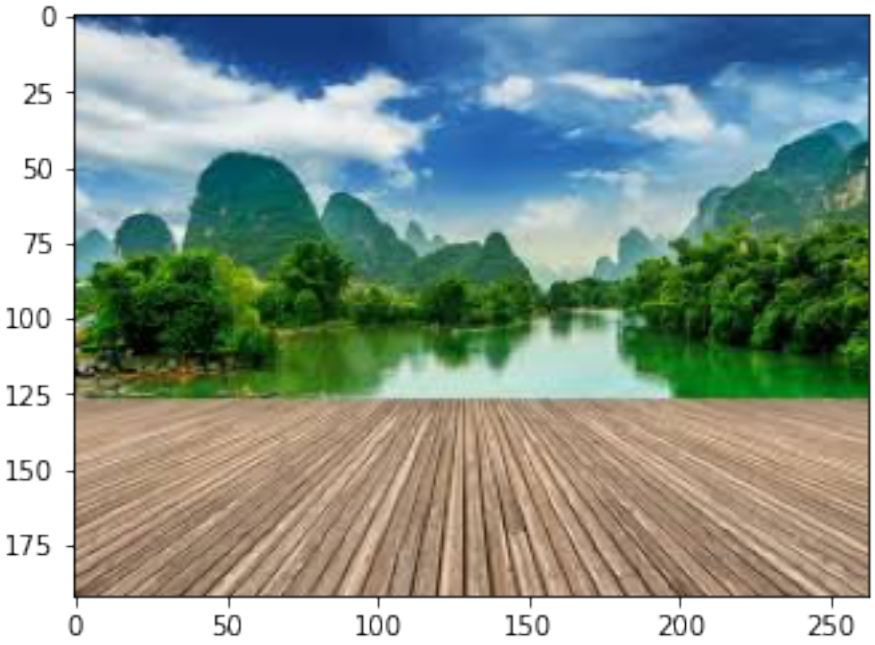
[2]:
```
filepath = os.getcwd() + '/ImageSegmentation/'

image = plt.imread(filepath + 'original.jpeg')

print(image.shape)
plt.imshow(image)
```

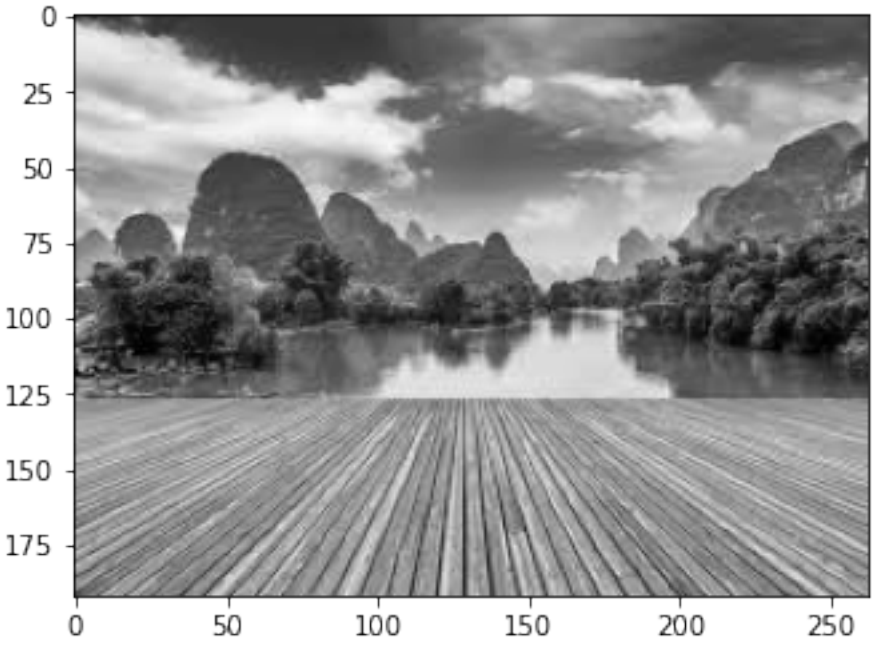
```
(192, 263, 3)
```

[2]:
```
<matplotlib.image.AxesImage at 0x7f64b4e3e7c0>
```



[3]:
```
gray = rgb2gray(image)
plt.imshow(gray, cmap='gray')
```

[3]: <matplotlib.image.AxesImage at 0x7f64b45798e0>



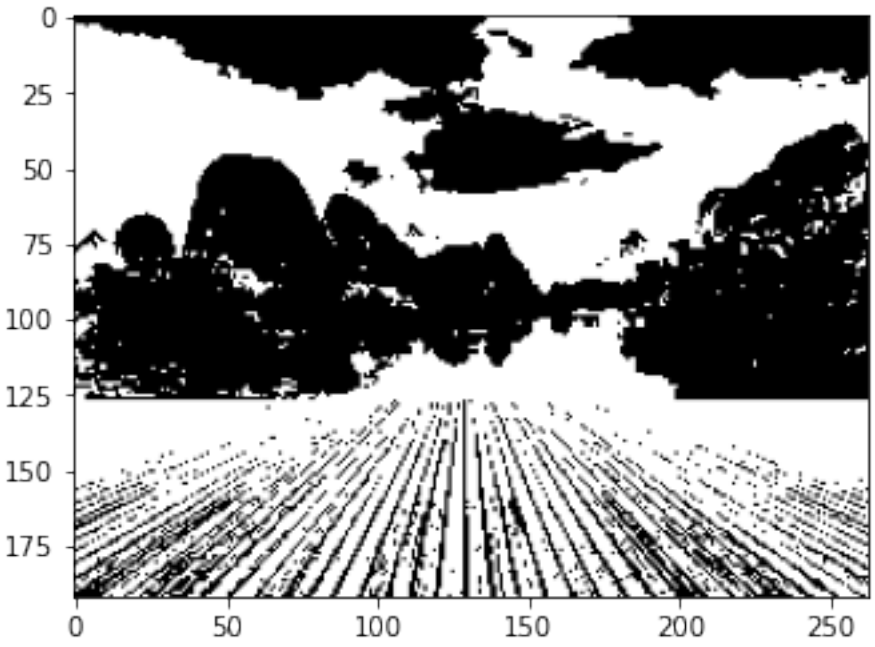
We will now iterate through the pixels of the image and manually turn each one black or white. First, we must compute the mean of the pixel values.

[4]:
```
gray_r = gray.reshape(gray.shape[0] * gray.shape[1])

for i in range(gray_r.shape[0]):
    if gray_r[i] > gray_r.mean():
        gray_r[i] = 1
    else:
        gray_r[i] = 0

gray = gray_r.reshape(gray.shape[0], gray.shape[1])
plt.imshow(gray, cmap = 'gray')
```

[4]:
```
<matplotlib.image.AxesImage at 0x7f64b44f6580>
```

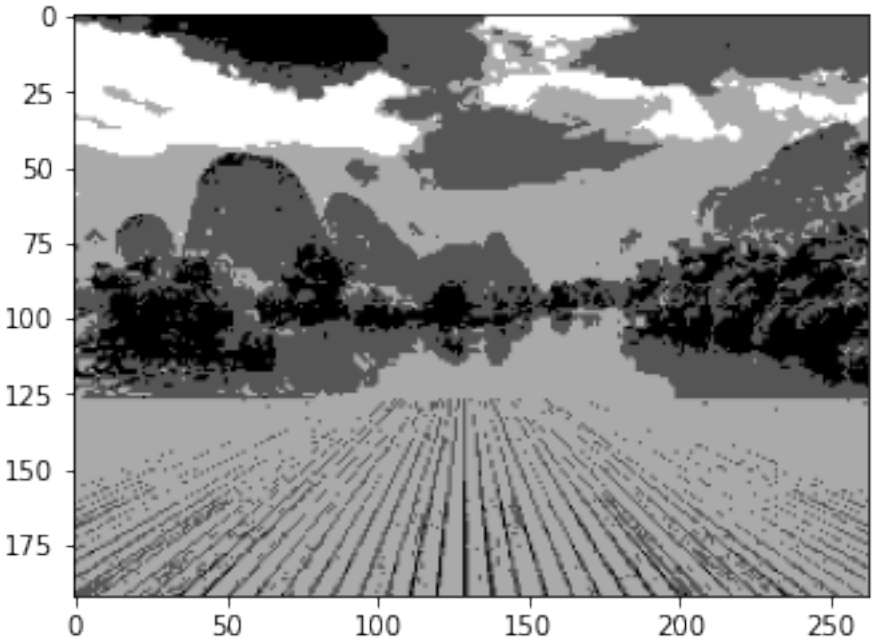


We can do further segmentation by having more threshold values as follows.

[5]:
```
gray = rgb2gray(image)
gray_r = gray.reshape(gray.shape[0]*gray.shape[1])
for i in range(gray_r.shape[0]):
    if gray_r[i] > gray_r.mean():
        gray_r[i] = 3
    elif gray_r[i] > 0.5:
        gray_r[i] = 2
    elif gray_r[i] > 0.25:
```

```
        gray_r[i] = 1
    else:
        gray_r[i] = 0
gray = gray_r.reshape(gray.shape[0],gray.shape[1])
plt.imshow(gray, cmap='gray')
```

[5]: <matplotlib.image.AxesImage at 0x7f64b44e07c0>

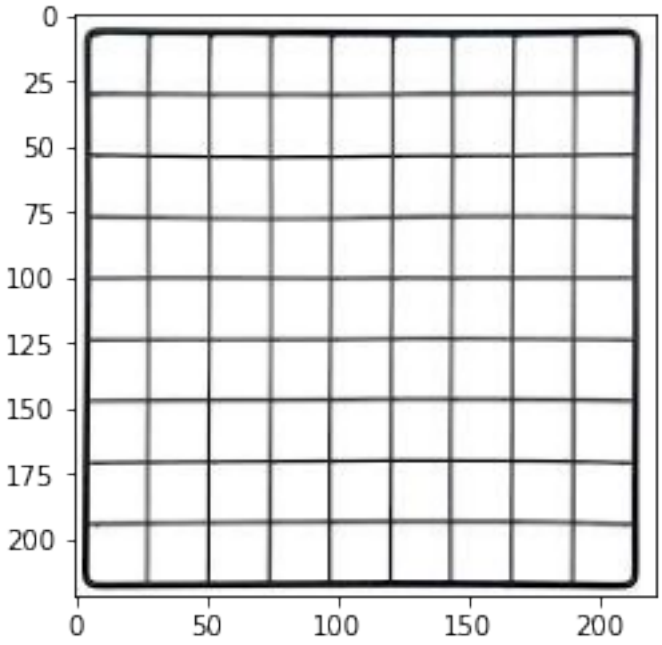


1.2 Edge Detection Through Convolution by Kernels

We will be using kernels and convolutions on an image to perform basic feature extraction, in this case edges.

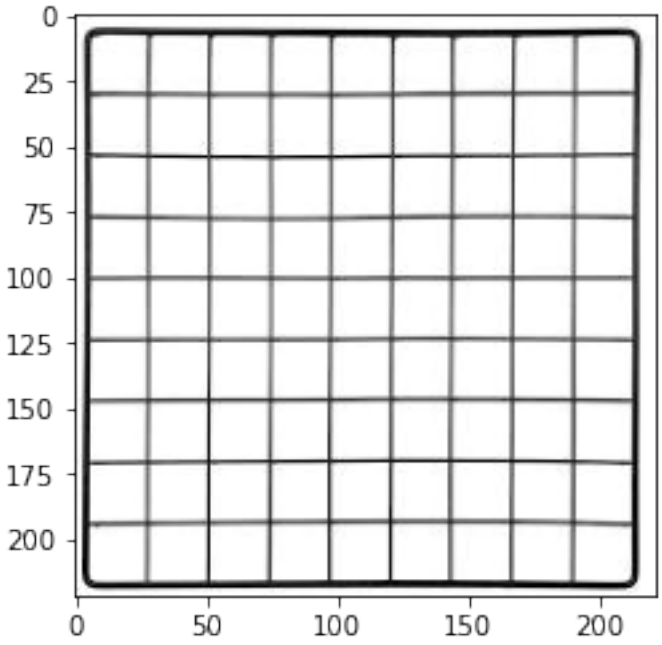
[6]:
```
grid = plt.imread(filepath + 'grid.jpeg')
plt.imshow(grid)
```

[6]: <matplotlib.image.AxesImage at 0x7f64b4437c10>



[7]:
```
grid_g = rgb2gray(grid)
plt.imshow(grid_g, cmap = 'gray')
```

[7]: <matplotlib.image.AxesImage at 0x7f64b442b8e0>



we will now define the kernels of the filters.

[8]:
```
horizontal = np.array([[1,2,1],[0,0,0],[-1,-2,-1]])
vertical = np.array([[-1,0,1],[-2,0,2],[-1,0,1]])
```

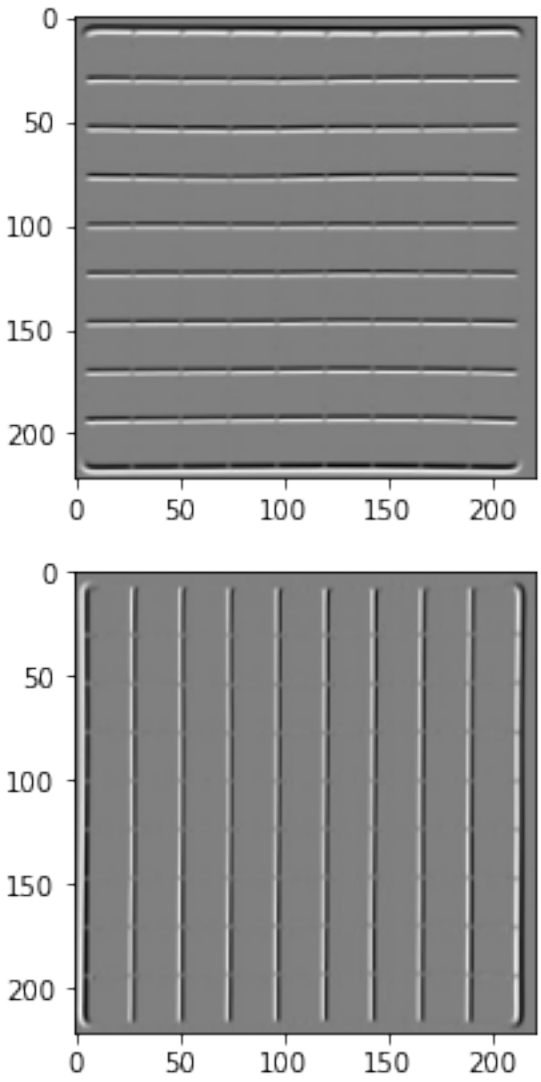
We will now convolve this filter over the image.

[9]:
```
out_h = ndimage.convolve(grid_g, horizontal, mode = 'reflect')
out_v = ndimage.convolve(grid_g, vertical, mode = 'reflect')


fig = plt.figure(figsize = (5,7))
plt.subplot(211)
plt.imshow(out_h, cmap = 'gray')

plt.subplot(212)
plt.imshow(out_v, cmap = 'gray')
```

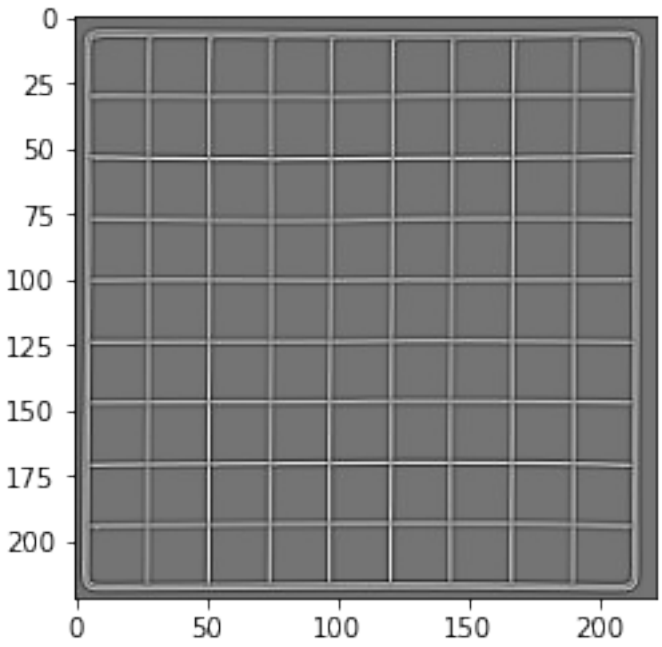
[9]:
```
<matplotlib.image.AxesImage at 0x7f64b4332640>
```



The laplace filter can detect both horizontal and vertical edges.

[10]:
```
laplace = np.array([[1,1,1],[1,-8,1],[1,1,1]])
out_l = ndimage.convolve(grid_g, laplace, mode = 'reflect')
plt.imshow(out_l, cmap = 'gray')
```

[10]: `<matplotlib.image.AxesImage at 0x7f64b426fcd0>`



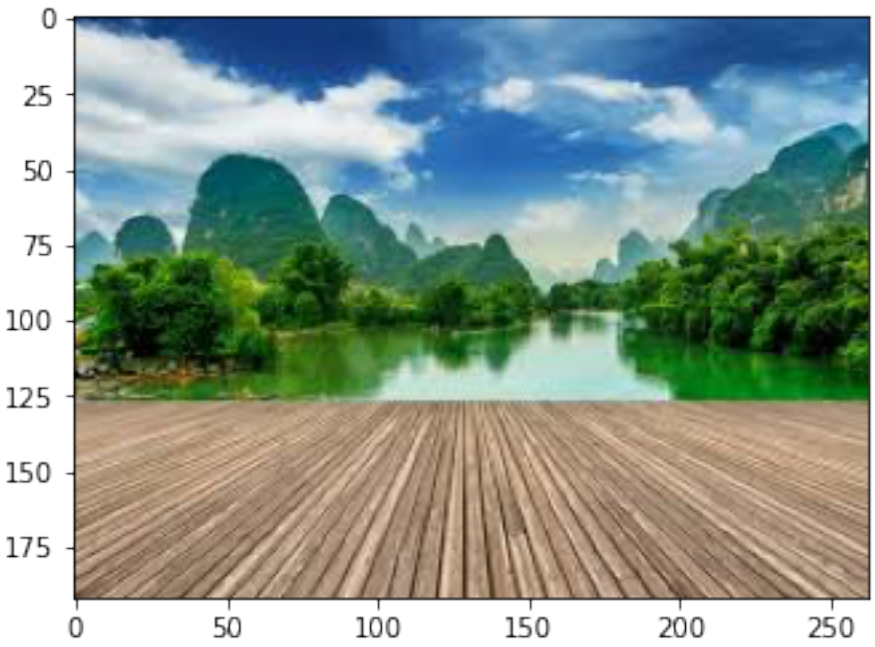
1.3 Clustering Based Image Segmentation

This will just be a simple implementation of k-means on an image to segment it.

[11]:
```
pic = plt.imread(filepath + 'original.jpeg')/255  # dividing by 255 to bring␣
↪the pixel values between 0 and 1
print(pic.shape)
plt.imshow(pic)
```

```
(192, 263, 3)
```

[11]:
```
<matplotlib.image.AxesImage at 0x7f64b425d520>
```



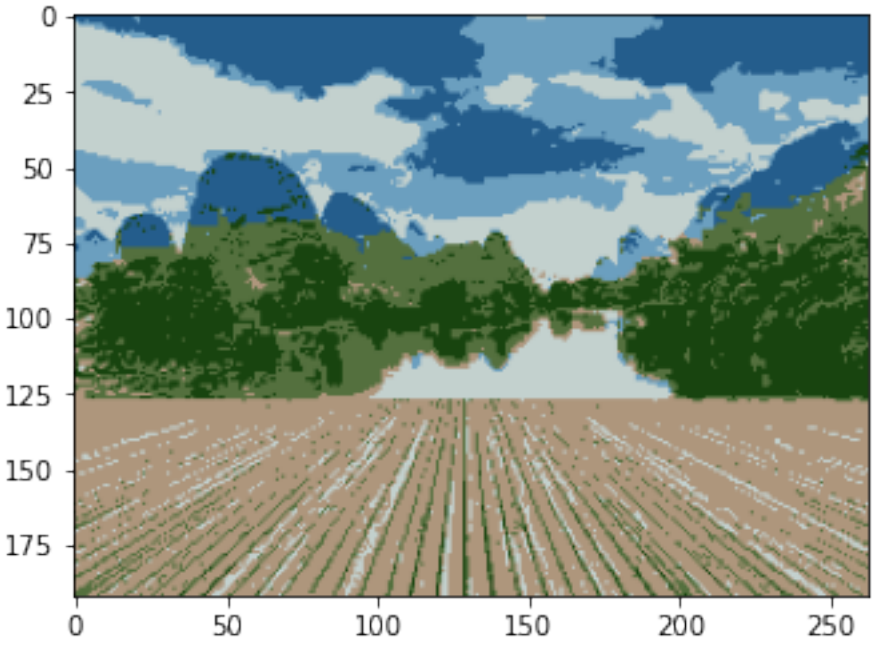
[12]:
```
pic_n = pic.reshape(pic.shape[0]*pic.shape[1], pic.shape[2])
pic_n.shape
```

[12]:
```
(50496, 3)
```

[13]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 6, random_state = 3).fit(pic_n)
clustered = kmeans.cluster_centers_[kmeans.labels_]
```

[14]:
```
cluster_pic = clustered.reshape(pic.shape[0], pic.shape[1], pic.shape[2])
plt.imshow(cluster_pic)
```

[14]:
```
<matplotlib.image.AxesImage at 0x7f64a98a11c0>
```


0
25
50
75
100
125
150
175
0
50
100
150
200
250