Image Data Analysis

August 8, 2021

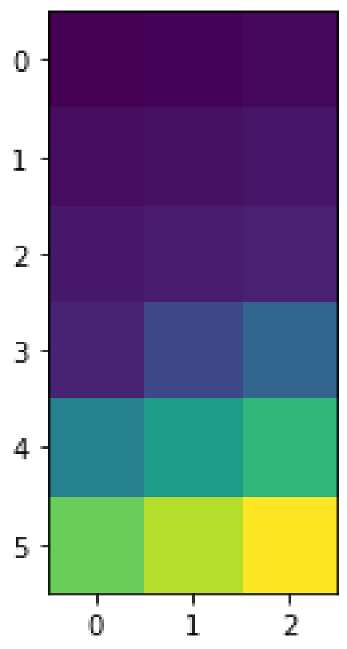
1 Image Data Analysis

In this post, I will analyze image data using an open-source library OpenCV. An image is nothing but a matrix. First, I will show matrix data as an image. Second, I will use a tire thread image data and apply different functions and filtering methods to that image.

[ ]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
```

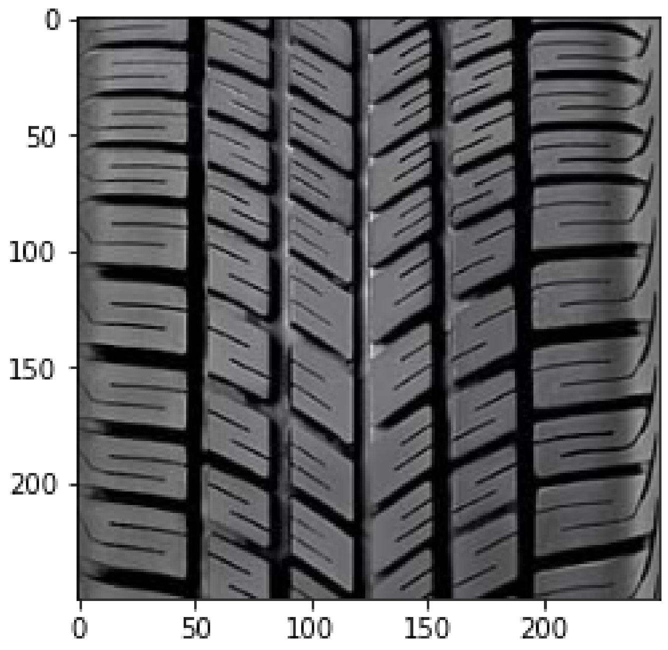
[9]:
```
# Say A and B are two matrices. We can concatenate these two matrices and␣
↪display as an image.
A = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
B = [[10, 20, 30], [40, 50, 60], [70, 80, 90]]
X= A+B
plt.imshow(X)
```

[9]:
```
<matplotlib.image.AxesImage at 0x7f9fd5a41c10>
```



[18]:
```python
# load the input image and display it to our screen
image = cv.imread('tread.jpeg')
image_2 = cv.cvtColor(image, cv.COLOR_BGR2RGB)
plt.imshow(image_2)
```

[18]:
```
<matplotlib.image.AxesImage at 0x7f9fd5ef2af0>
```

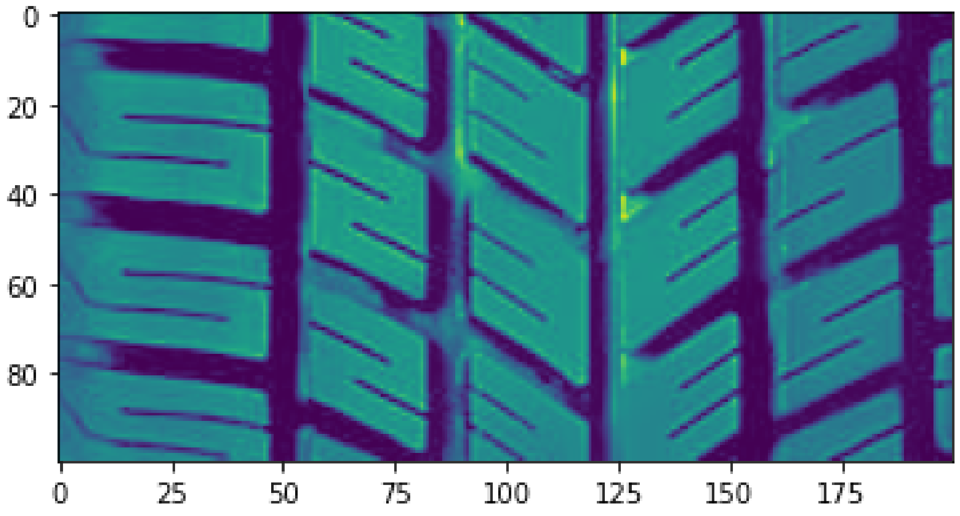


[19]:
```python
# Check the shape of the image
image.shape
```

[19]:
```
(250, 250, 3)
```

[20]:
```python
# segment of an image
img_segment = image[100:200,0:200,0]
plt.imshow(image[100:200,0:200,0])
```

[20]:
```
<matplotlib.image.AxesImage at 0x7f9fd60fe8e0>
```



```
[21]: %matplotlib inline
```

2 Basis Functions

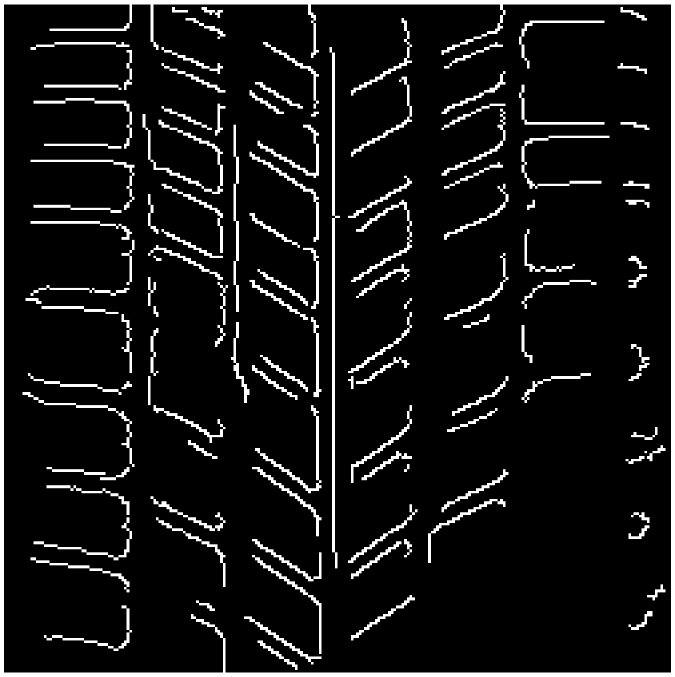
```
[ ]: # Read in an image
     img = cv.imread('tread.jpeg')
     plt.imshow(img)
```

```
[ ]: blank = np.zeros(img.shape, dtype='uint8')
     #cv.imshow('Blank', blank)
     cv2_imshow(blank)
```

```
[ ]: # Converting to grayscale
     gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
     #cv.imshow('Gray', gray)
     cv2_imshow(gray)
```

```
[ ]: # Blur
     blur = cv.GaussianBlur(img, (7,7), cv.BORDER_DEFAULT)
     #cv.imshow('Blur', blur)
     cv2_imshow(blur)
```

```
[ ]: # Edge Cascade
     canny = cv.Canny(blur, 125, 175)
     #cv.imshow('Canny Edges', canny)
     cv2_imshow(canny)
```

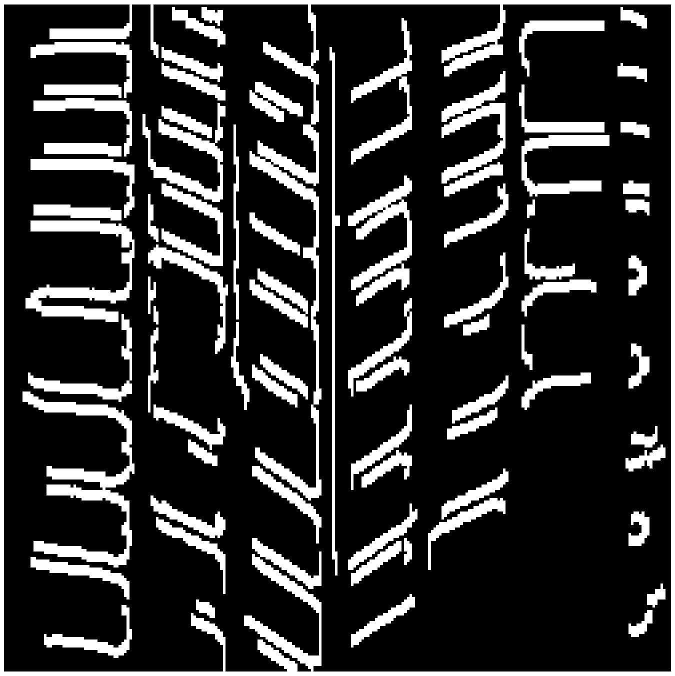


```
[ ]: contours, hierarchies = cv.findContours(canny, cv.RETR_LIST, cv.
      ↪CHAIN_APPROX_SIMPLE)
     print(f'{len(contours)} contour(s) found!')
```

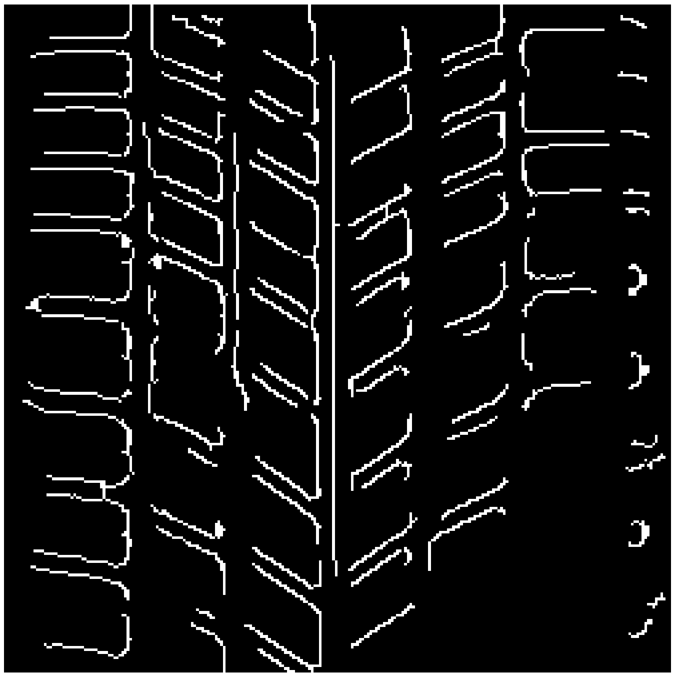
```
127 contour(s) found!
```

```
[ ]: cv.drawContours(canny, contours, -1, (0,255,0), 3)
     #cv.drawContours(img, contours, -1, (0,255,0), 3)
     cv2_imshow(canny)
```

```
[ ]: # Dilating the image
     dilated = cv.dilate(canny, (7,7), iterations=3)
     #cv.imshow('Dilated', dilated)
     cv2_imshow(dilated)
```



[ ]:
```
# Eroding
eroded = cv.erode(dilated, (7,7), iterations=3)
#cv.imshow('Eroded', eroded)
cv2_imshow(eroded)
```



[ ]:
```
# Resize an image
resized = cv.resize(img, (100,100), interpolation=cv.INTER_CUBIC)
#cv.imshow('Resized', resized)
cv2_imshow(resized)
```

[ ]:
```
# Cropping an image
cropped = img[50:200, 200:400]
#cv.imshow('Cropped', cropped)
cv2_imshow(cropped)
```