

image-processing

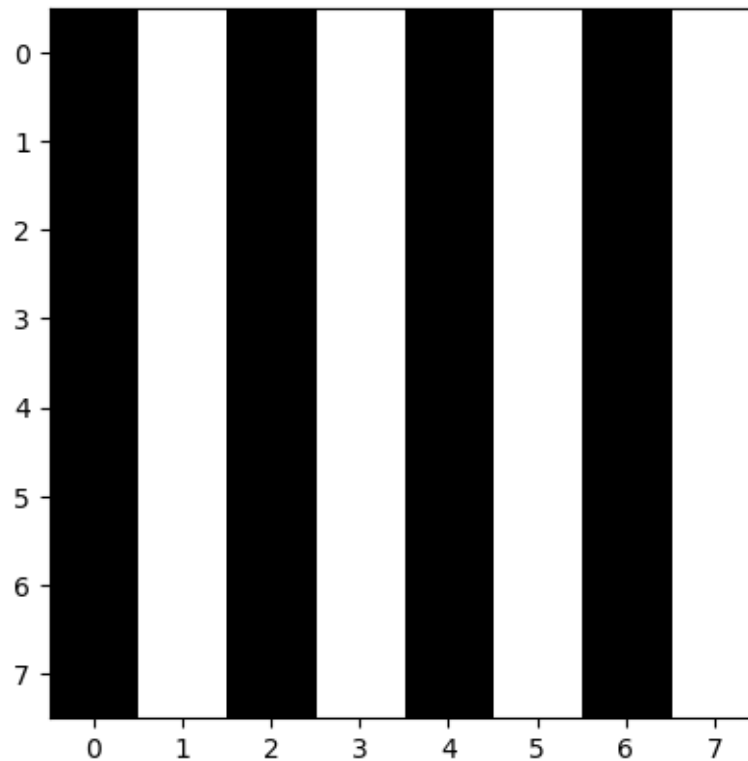
January 19, 2024

```
[1]: import numpy as np
import matplotlib.pyplot as plt
```

0.1 Gray scale images

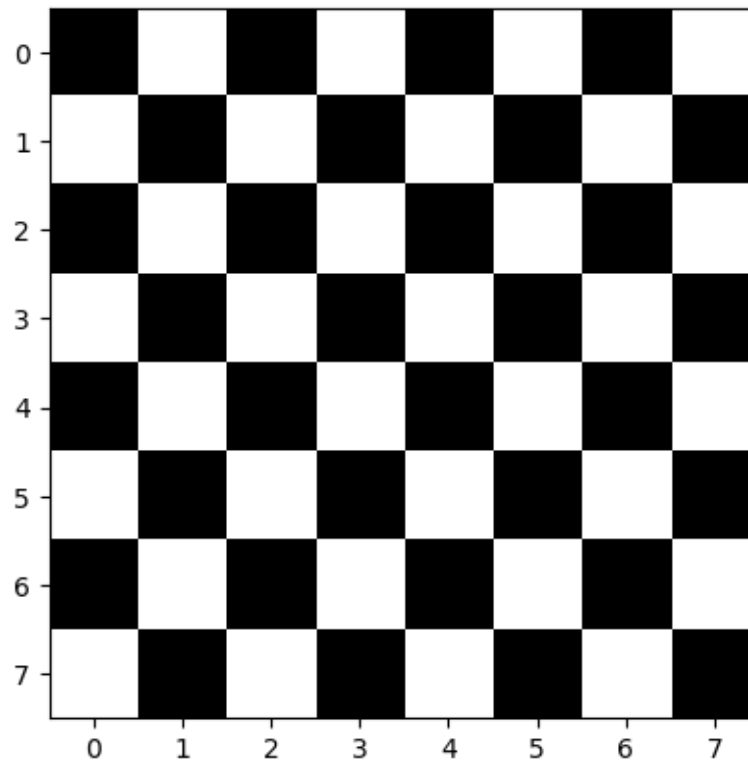
```
[3]: # This is 8 x 8 pixel gray image i.e 64 pixels total. Here, 0 is black and 255
    ↪ is white
x = np.array([ [0,255,0,255,0,255,0,255],
               [0,255,0,255,0,255,0,255],
               [0,255,0,255,0,255,0,255],
               [0,255,0,255,0,255,0,255],
               [0,255,0,255,0,255,0,255],
               [0,255,0,255,0,255,0,255],
               [0,255,0,255,0,255,0,255],
               [0,255,0,255,0,255,0,255] ])
plt.imshow(x, cmap='gray')
x.shape
```

```
[3]: (8, 8)
```



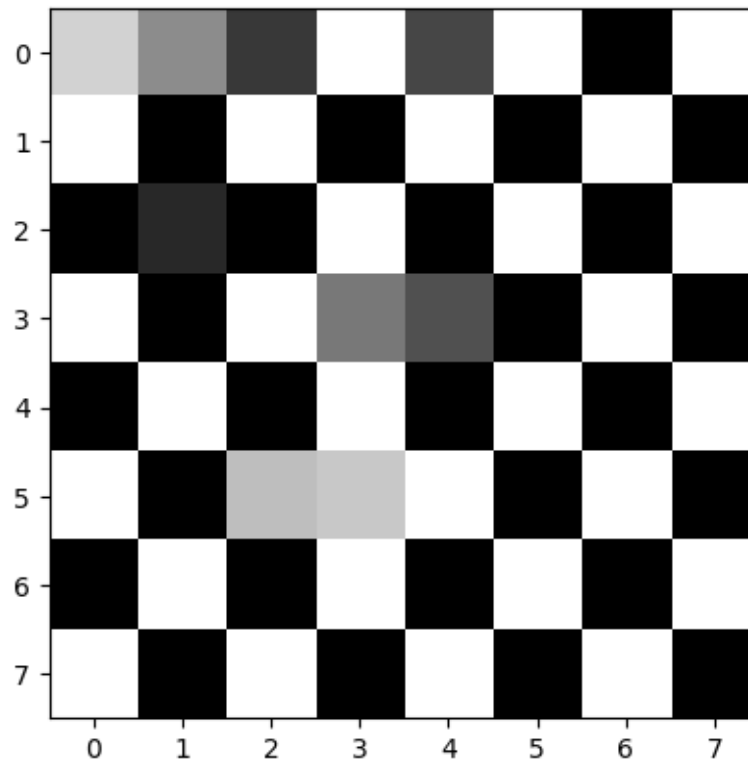
```
[4]: # This is 8 x 8 pixel gray image. If we want to make this as chess board type
      ↪ image. Here, 0 is black and 255 is white
x = np.array([ [0,255,0,255,0,255,0,255],
                [255,0,255,0,255,0,255,0],
                [0,255,0,255,0,255,0,255],
                [255,0,255,0,255,0,255,0],
                [0,255,0,255,0,255,0,255],
                [255,0,255,0,255,0,255,0],
                [0,255,0,255,0,255,0,255],
                [255,0,255,0,255,0,255,0] ])
plt.imshow(x, cmap='gray')
x.shape
```

```
[4]: (8, 8)
```



```
[5]: # This is 8 x 8 pixel gray image. Now let us twick values of various pixel
      ↪ between 0 to 255. Hear, 0 is black and 255 is white.
x = np.array([ [210,140,56,255,70,255,0,255],
               [255,0,255,0,255,0,255,0],
               [0,40,0,255,0,255,0,255],
               [255,0,255,120,80,0,255,0],
               [0,255,0,255,0,255,0,255],
               [255,0,190,200,255,0,255,0],
               [0,255,0,255,0,255,0,255],
               [255,0,255,0,255,0,255,0] ])
plt.imshow(x, cmap='gray')
x.shape
```

```
[5]: (8, 8)
```



1 Colour images

In colour image each pixel is represented by 3 values - RGB

[0,0,0]: Black

[255, 255,255]: White

[255,0,0]: Red

[0,255,0]: Green

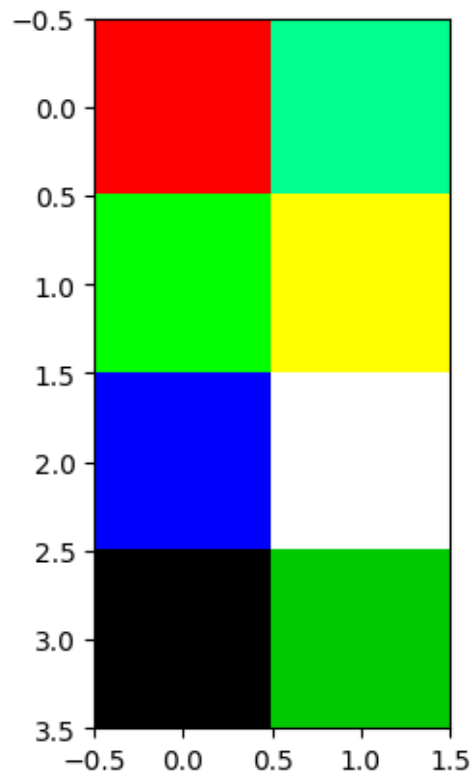
[0,0,255]: Blue

[100,0,0]: Dark Red

```
[6]: # Let us generate colour image of 4 x 2 pixels
y=np.array([[ [255,0,0], [0,255,140]],
             [[0,255,0], [255,255,0]],
             [[0,0,255], [255,255,255]],
             [[0,0,0], [0,200,2]]])
plt.imshow(y)
```

```
y.shape
```

```
[6]: (4, 2, 3)
```



1.1 Image Processing

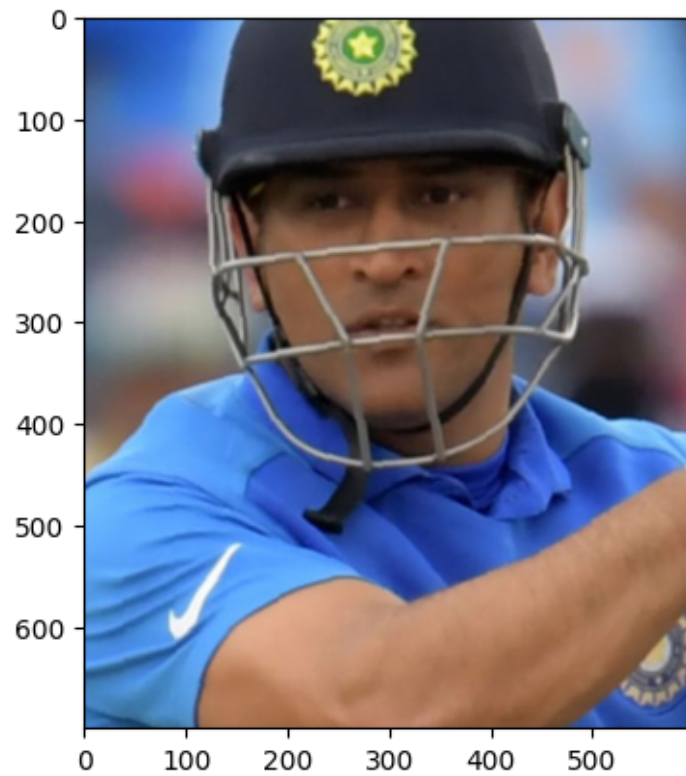
```
[7]: img=plt.imread("/kaggle/input/msdhoni/ms-dhoni.jpg")
print(img.shape) # printing shape of image. It is colour so 427 x 630 image
             ↳having 3 channel - RGB
plt.imshow(img)
plt.show()
```

```
(900, 1600, 3)
```



1.2 Crop Image

```
[30]: # Crop image
img2=img[100:800,400:1000]
plt.imshow(img2)
plt.show()
```



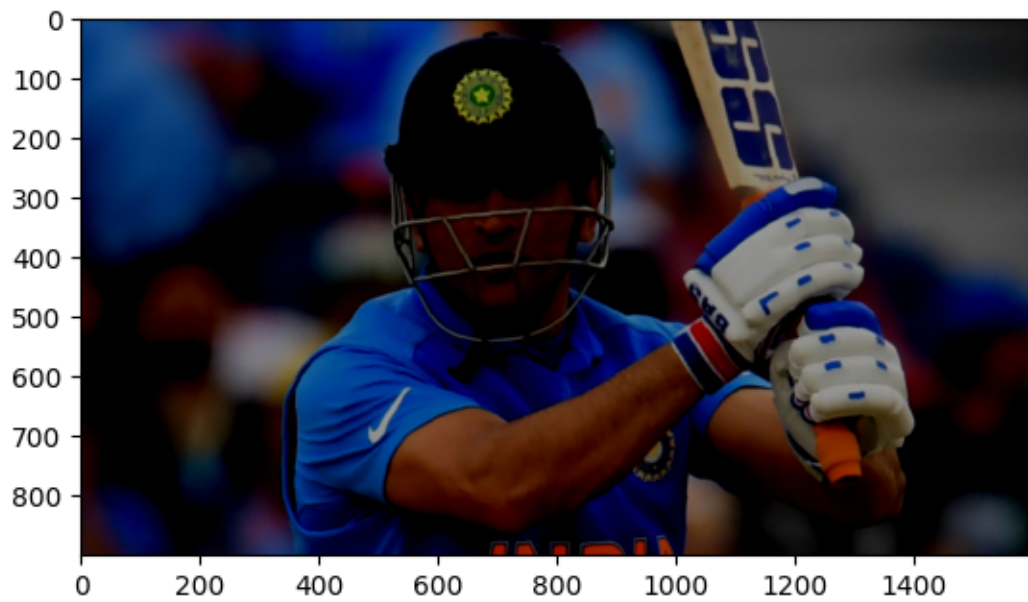
1.3 Increase Brightness

```
[31]: # Increase the brightness  
BrightnessUp = 100 * np.ones((img.shape), dtype='int32')  
img3=img+BrightnessUp  
plt.imshow(img3)  
plt.show()
```



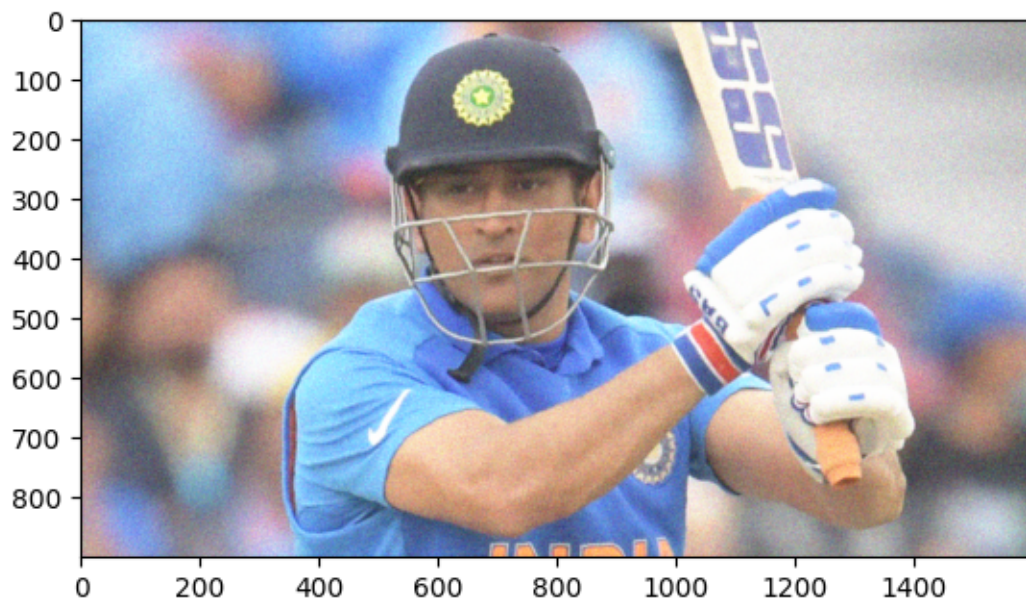
1.4 Decrease the brightness

```
[32]: # Decrease the brightness  
BrightnessDown = -100 * np.ones((img.shape),dtype='int32')  
img3=img+BrightnessDown  
plt.imshow(img3)  
plt.show()
```



1.5 Add noise

```
[33]: # Add noise to the image
AddNoise=np.random.randint(0,100,img.shape) # generating random number between 0
↳ 0 to 100 for the noise value
img4=img+AddNoise
plt.imshow(img4)
plt.show()
```



2 Displaying image using OpenCV

```
[34]: import cv2
```

2.1 Reading image using OpenCv, this is default BGR format

```
[35]: imgOCV=cv2.imread("/kaggle/input/msdhoni/ms-dhoni.jpg")

plt.imshow(imgOCV)
plt.show()
```



2.2 convert image read by openCV in BRG format to RGB format

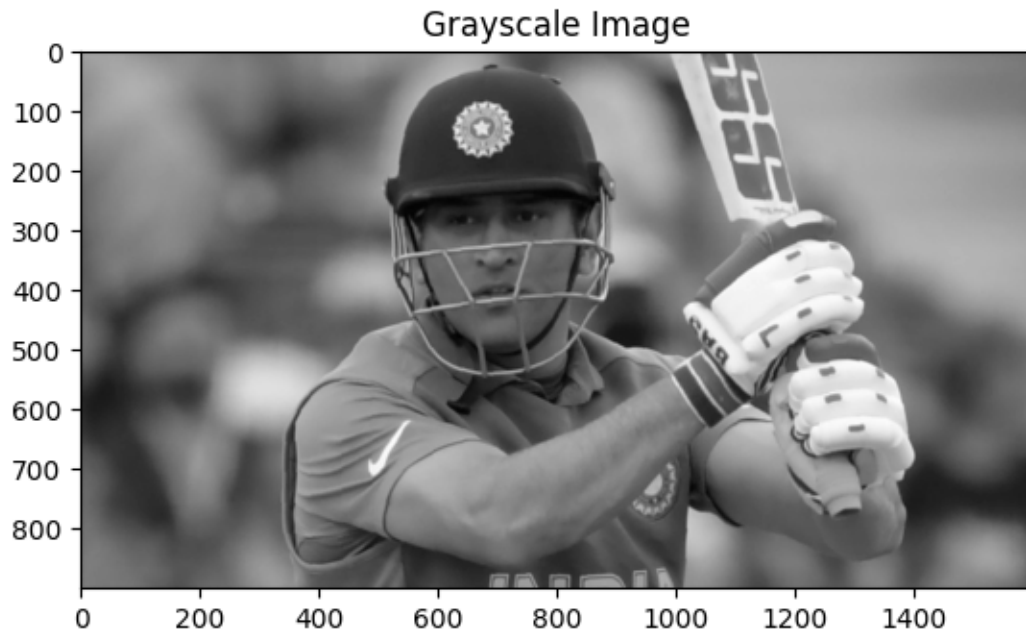
```
[36]: imgOCV1=cv2.cvtColor(imgOCV,cv2.COLOR_BGR2RGBA)  
plt.imshow(imgOCV1)  
plt.show()
```



2.3 Reading image in grayscale using openCV

```
[39]: # Convert to grayscale
imgGray = cv2.cvtColor(imgOCV, cv2.COLOR_BGR2GRAY)

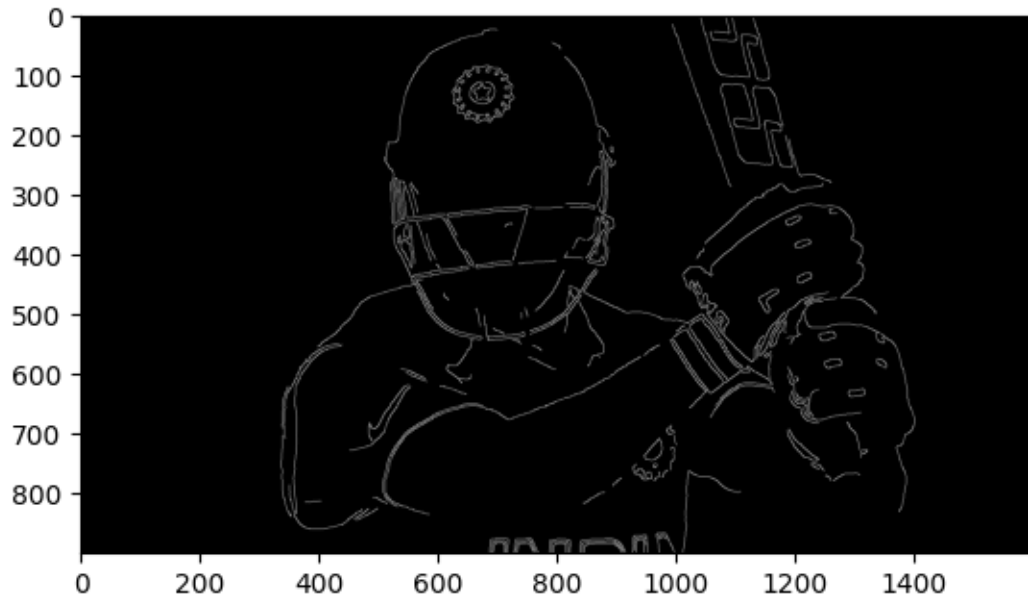
# Display the grayscale image
plt.imshow(imgGray, cmap='gray')
plt.title('Grayscale Image')
plt.show()
```



2.4 Detecting boundries of the objects in image

```
[40]: imgOCV3 = cv2.Canny(imgOCV,150,255)

plt.imshow(imgOCV3, cmap='gray')
plt.show()
```



2.5 Image Blurring

```
[46]: # Display the original image
plt.imshow(cv2.cvtColor(imgOCV, cv2.COLOR_BGR2RGB))
plt.title('Original Image')
plt.show()

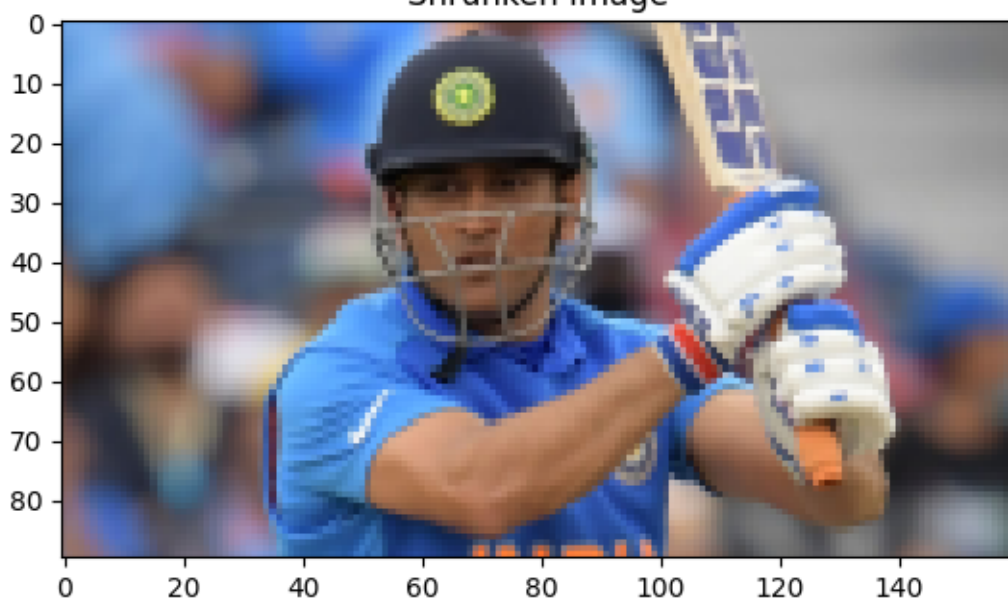
# Resize the image (shrink by a factor of 0.5 in both dimensions)
shrunk_img = cv2.resize(imgOCV, None, fx=0.1, fy=0.1, interpolation=cv2.
    ↪ INTER_AREA)

# Display the shrunken image
plt.imshow(cv2.cvtColor(shrunk_img, cv2.COLOR_BGR2RGB))
plt.title('Shrunken Image')
plt.show()
```

Original Image



Shrunk Image

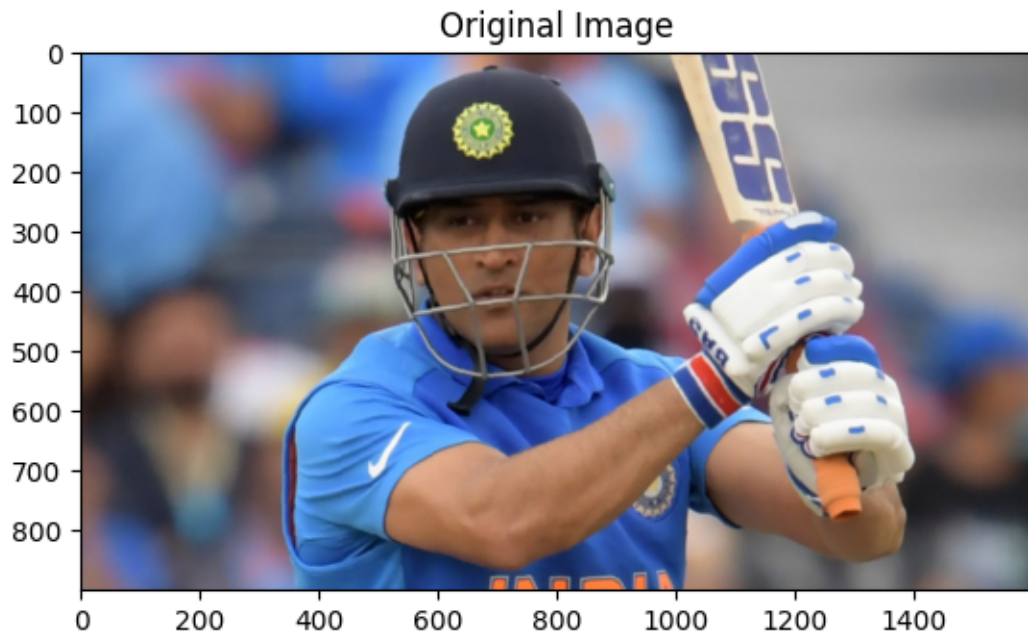


```
[49]: # Blurring - smoothes the image out
imgOCV8 = imgOCV1

blur = cv2.blur(imgOCV8,(18, 18))
```

```
gblur = cv2.GaussianBlur(imgOCV8,(7,7),0)

plt.imshow(imgOCV8), plt.title('Original Image')
plt.show()
plt.imshow(blur), plt.title('Blurred Image')
plt.show()
plt.imshow(gblur),plt.title('Gaussian Blurred Image')
plt.show()
```



Blurred Image



Gaussian Blurred Image

