***How to turn any image into a pencil sketch with 10 lines of code***

Sakshi Agrawal

Sakshi.agrawal8800@gmail.com

I have always been fascinated by computer vision, and especially by its power to manipulate images.

An image is basically an array of numbers to Python. So we can perform a variety of matrix manipulations to get some very interesting results. In this post, I talk about how to reduce an image into a ‘pencil’ outline.

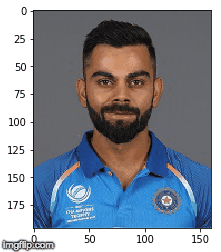
## The Steps

The process is pretty simple:

1. Grayscale the image
2. Invert it
3. Blur the inverted image
4. Dodge blend the blurred and grayscale image.

We can pick any image from the Internet. I will go with this image of Indian cricketer Virat Kohli:

https://miro.medium.com/freeze/max/33/1*TG0CeJvuv1iwskaCQSEyNA.gif?q=20



## **1. Load image**

import imageio  
img="[http://static.cricinfo.com/db/PICTURES/CMS/263600/263697.20.jpg](http://static.cricinfo.com/db/PICTURES/CMS/263600/263697.20.jpg?source=post_page---------------------------)"  
start\_img = imageio.imread(img)

https://miro.medium.com/max/30/1*v-q4is6XGAQ6XLYI2m7NzQ.jpeg?q=20



Initial image

You can see how Python sees this image with the shape attribute:

start\_img.shape  
(196, 160, 30)

So this is a three channel image of size 196x160.

## **2. Greyscale**

We then make the image black and white.

Numpy doesn’t have any in-built function for grayscaling, but we can easily convert the image using the formula. You can learn why this formula works right [here](https://www.w3.org/Graphics/Color/sRGB?source=post_page---------------------------).

Y= 0.299 R + 0.587 G + 0.114 B

So our function will look like:

import numpy as np  
def grayscale(rgb):  
 return np.dot(rgb[...,:3], [0.299, 0.587, 0.114])

Applying grayscale:

gray\_img = grayscale(start\_img)

https://miro.medium.com/max/33/1*XOYmVdB9JvBtPAIkpnPVRw.png?q=20



Grayscaled Image

## **3. Invert image**

We can invert images simply by subtracting from 255, as grayscale images are 8 bit images or have a maximum of 256 tones.

inverted\_img = 255-gray\_img

https://miro.medium.com/max/33/1*k47Y4d4kAFP3tTydH2xlcQ.png?q=20



Inverted Image

## **4. Blur Image**

We now blur the inverted image. Blurring is done by applying a [Gaussian filter](https://en.wikipedia.org/wiki/Gaussian_blur?source=post_page---------------------------)to the inverted image. The key here is the variance of the Gaussian function or sigma.

As sigma increases, the image becomes more blurred. Sigma controls the extent of the variance and thus, the degree of blurring.

import scipy.ndimage  
blur\_img = scipy.ndimage.filters.gaussian\_filter(inverted\_img,sigma=5)

https://miro.medium.com/freeze/max/30/1*sS1PIpHogxUKPNDRkOBLuQ.gif?q=20



More blurring on increasing sigma

## **5. Dodge and Merge**

The [Colour Dodge](https://en.wikipedia.org/wiki/Blend_modes?source=post_page---------------------------) blend mode divides the bottom layer by the inverted top layer. This lightens the bottom layer depending on the value of the top layer. We have the blurred image, which highlights the boldest edges.

As all our images are read using Numpy, all the matrix calculations are super fast.

def dodge(front,back):  
 result=front\*255/(255-back)   
 result[result>255]=255  
 result[back==255]=255  
 return result.astype(‘uint8’)final\_img= dodge(blur\_img,gray\_img)

https://miro.medium.com/max/33/1*K8GCJlziwZm3NmONoy3B7g.png?q=20



Final image

And that’s it!

## **6. Plot and save**

We can plot our final image using plt.imgshow. Note that we need to keep the cmap argument equal to “gray”.

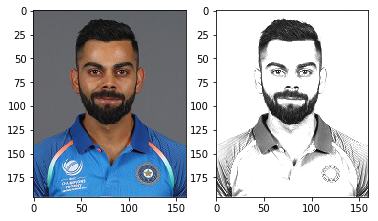
import matplotlib.pyplot as plt  
plt.imshow(final\_img, cmap=”gray”)

We can save the image using:

plt.imsave(‘img2.png’, final\_img, cmap=’gray’, vmin=0, vmax=255)

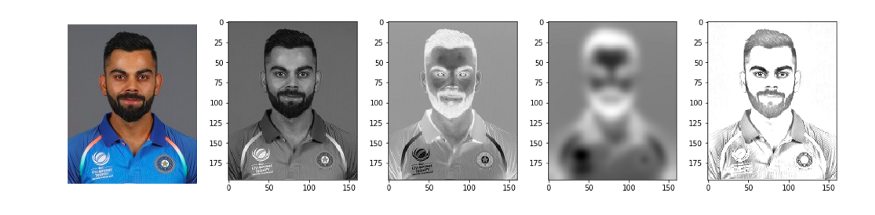
## Final result

https://miro.medium.com/max/38/1*cFcDUhcYTBx4AtGpXzeyXw.png?q=20



## Entire code in action

https://miro.medium.com/max/38/1*h94l7xWnXk_6dbzDUuPamw.png?q=20



Each stage of the algorithm

Here we don’t have much room to play with, except with the sigma parameter while blurring.

As sigma increases, the pic becomes clearer but run time also increases. So a sigma of 5 works well for us.

https://miro.medium.com/freeze/max/30/1*Xdeu0DckWzCXMX6P1TwGzA.gif?q=20



effect of increasing sigma

## Condensed code:

I promised 10 lines or less, so here you go:

As always, you can find the entire detailed code on my [GitHub](https://github.com/rra94/sketchify/tree/master?source=post_page---------------------------).

PS this is how I created my Medium DP. If you like this blog, show some ❤️ :)

Also I don’t own this image of Virat. I hope he doesn’t mind!