sketch

October 23, 2018

We use some image manipulation to convert an image into a "sketch".

```
In [1]: import numpy as np
        import matplotlib.pylab as pl
        import scipy
        import scipy.misc as misc
```

Use an image downloaded from the interwebs.

```
In [2]: baby=misc.imread('baby.jpg')

In [3]: baby.shape

Out[3]: (275, 183, 3)

In [4]: baby.dtype

Out[4]: dtype('uint8')

In [5]: pl.imshow(baby)
        pl.show()
```

This image *may* be too small to work with. A bigger image with more detail is better.

Here is something funny you can do. It has nothing to do with what we are trying to do!

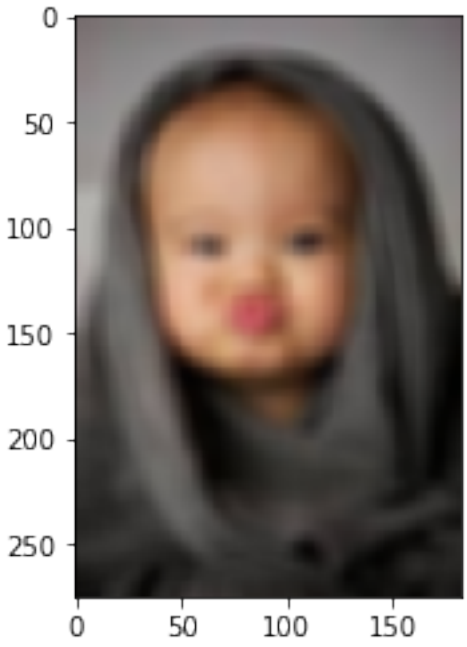
```
In [6]: strange=baby[:,:,(2,1,0)]
        pl.imshow(strange)
        pl.show()
```

Define a function that blurs an image by averaging all the pixels in square of given width around each pixel.

This is a slow operation! There is may be a way to speed it using convolution.

```
In [7]: def blur(img,w):
            xm,ym,_=img.shape
            new=np.zeros(img.shape,dtype='uint8')
            for c in range(xm):
                for d in range(ym):
                    for i in range(3):
                        new[c,d,i]=np.uint8(
                                    np.average(
                                        np.float64(img[
                                            max(c-w,0):min(c+w,xm),
                                            max(d-w,0):min(d+w,ym),
                                            i
                                ])))
            return new
```

```
In [8]: blurbaby=blur(baby,5)
        pl.imshow(blurbaby)
        pl.show()
```



Negate an image since 255 is the maximum value of any pixel.

```
In [9]: def neg(img):
            return 255-img
```

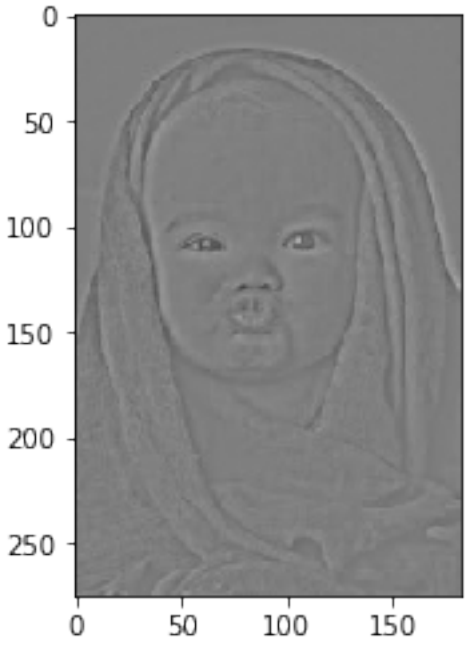
A function to convert an image to grayscale in a naive way by averaging the RGB values.

```
In [10]: def gray(img):
             new=np.float64(img[:,:,0])
             new+=np.float64(img[:,:,1])
             new+=np.float64(img[:,:,2])
             new=new/3
             new=np.uint8(new)
             return np.transpose(np.array([new,new,new]),axes=(1,2,0))
```

This is the *main* step. We take an average of the original image with the negation of the blurred image (both in grayscale). This detects edges.

```
In [11]: edge=(gray(baby)/2+gray(neg(blurbaby))/2)
```

```
In [12]: pl.imshow(edge)
         pl.show()
```



We see that we have a faint outline of the image.

The next function brings out this outline by stretch the gray levels. The `bl` value and all below it will become black and the `wh` value and all above it will become white.

```
In [13]: def stretch(img,bl,wh):
             tmp=np.minimum(np.maximum(img,bl),wh)
             return np.uint8(np.float64(tmp-bl)/(wh-bl)*255)
```

We generate a histogram of values in the `edge` image in order to find suitable black limit and white limits.

```
In [14]: [(i,np.sum(edge==i)) for i in np.unique(edge)]
```

```
Out[14]: [(61, 3),
          (65, 3),
          (73, 3),
          (74, 3),
          (75, 3),
          (77, 3),
          (79, 9),
          (80, 9),
          (81, 3),
          (82, 3),
          (83, 9),
          (84, 6),
          (85, 9),
          (86, 12),
```

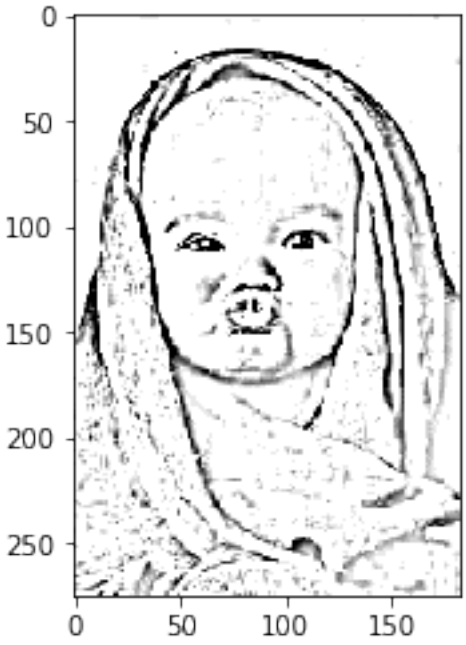
```
(87, 21),
(88, 12),
(89, 15),
(90, 21),
(91, 42),
(92, 12),
(93, 39),
(94, 30),
(95, 69),
(96, 54),
(97, 54),
(98, 99),
(99, 135),
(100, 123),
(101, 129),
(102, 162),
(103, 198),
(104, 225),
(105, 255),
(106, 270),
(107, 399),
(108, 387),
(109, 345),
(110, 417),
(111, 570),
(112, 621),
(113, 705),
(114, 828),
(115, 951),
(116, 1218),
(117, 1413),
(118, 1692),
(119, 2106),
(120, 2679),
(121, 3474),
(122, 4236),
(123, 5589),
(124, 7308),
(125, 9999),
(126, 15135),
(127, 24120),
(128, 16875),
(129, 10515),
(130, 8016),
(131, 6387),
(132, 4572),
(133, 3612),
(134, 2763),
```

```
 (135, 2052),
 (136, 1929),
 (137, 1632),
 (138, 1278),
 (139, 900),
 (140, 780),
 (141, 648),
 (142, 543),
 (143, 465),
 (144, 366),
 (145, 336),
 (146, 228),
 (147, 171),
 (148, 150),
 (149, 120),
 (150, 81),
 (151, 75),
 (152, 63),
 (153, 30),
 (154, 33),
 (155, 30),
 (156, 36),
 (157, 9),
 (158, 9),
 (159, 6),
 (160, 3),
 (161, 3),
 (167, 6),
 (168, 3),
 (170, 6),
 (172, 3),
 (176, 3),
 (177, 3)]
```

A possible heuristic is that since most of the image is gray, the value just below that with the largest number of entries (in our case 127) should be chosen as white; we choose 125. Going down to some value that occurs often enough as black; we choose 112.

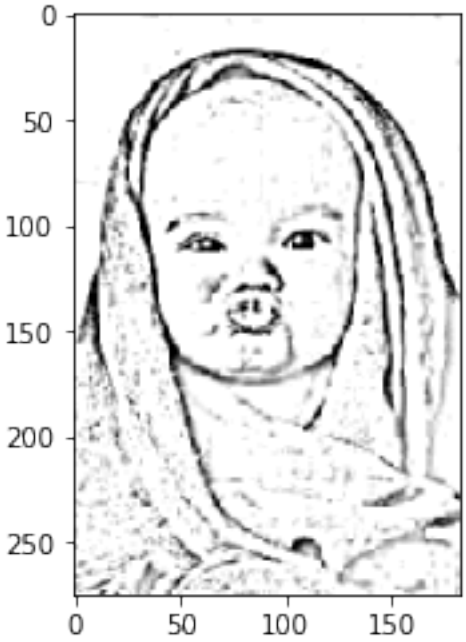
You can play with these choices till you get a nice image.

```
In [15]: sketch1=stretch(edge,113,125)
         pl.imshow(sketch1)
         pl.show()
```



Finally, all good sketch artists rub the paper with some bread crumbs to soften the edges. So we add a little blur!

```
In [16]: sketch=blur(sketch1,1)
         pl.imshow(sketch)
         pl.show()
```



Finally, save our handiwork.

```
In [17]: pl.imsave('sketch.jpg',sketch)
```

```
In [ ]:
```