Image Enhancement

Kasra Eskandari
955361005

June 27, 2020

Contents

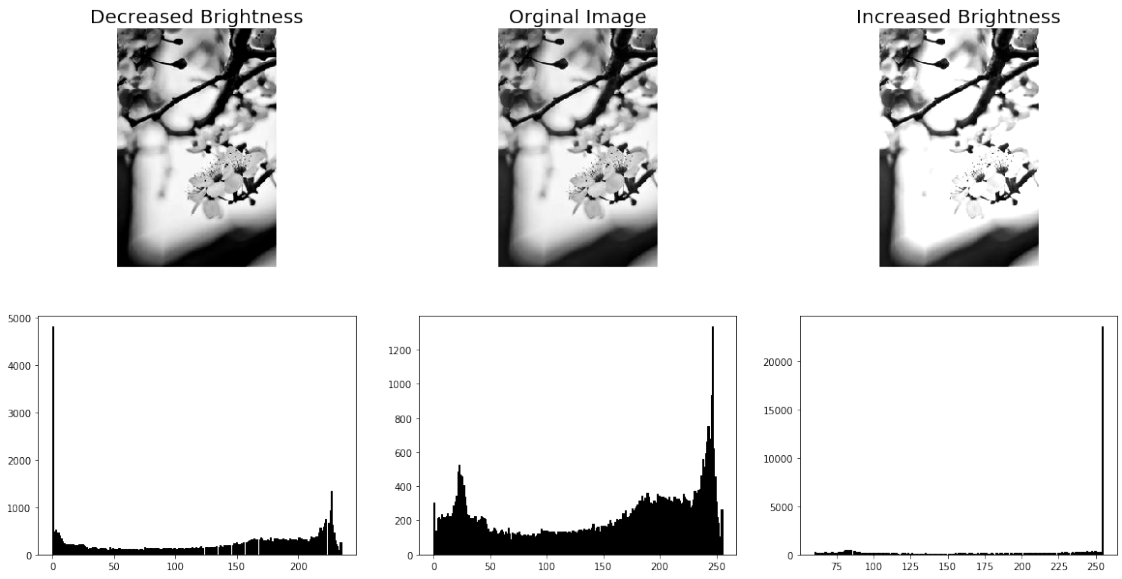
[1]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
from time import time
```

1 Point Transforms

1.1 Brightness

$$T(i) = \alpha i + \beta$$

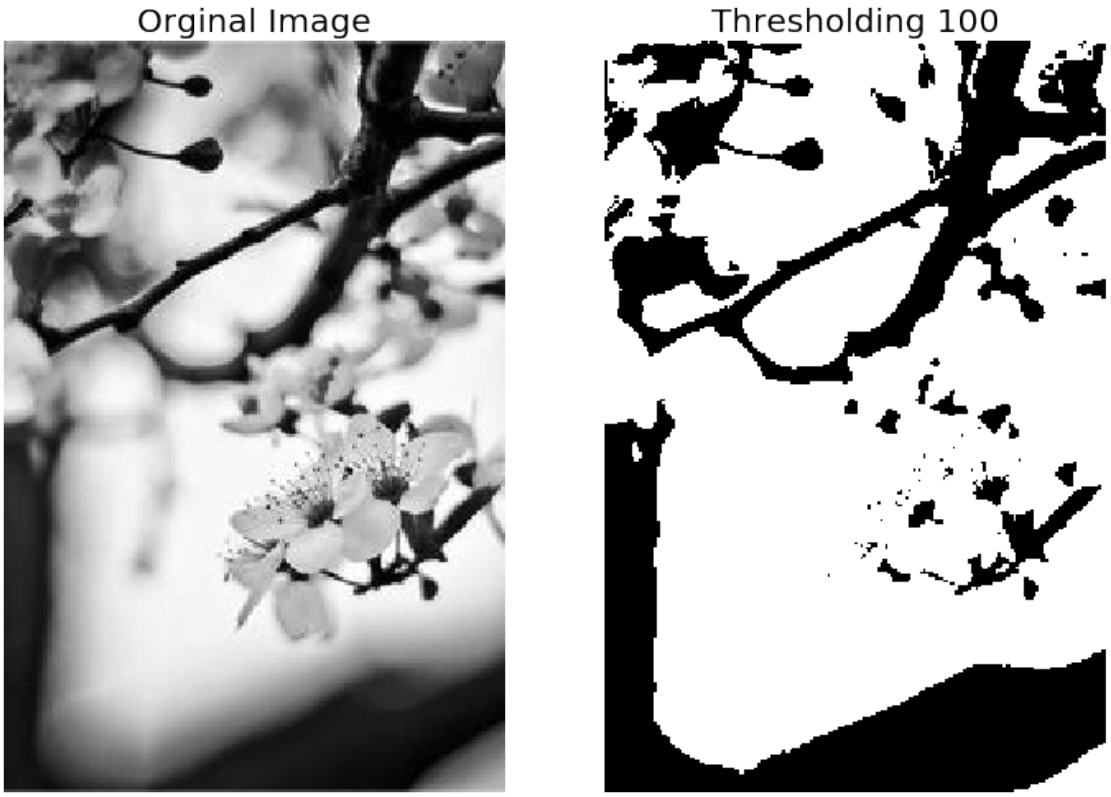
[2]:
```
def changeBrightness(img,alpha,beta):
    img=alpha*img.copy().astype('int')+beta
    img[img<0]=0
    img[img>255]=255
    return img.astype(np.uint8)
img=cv2.imread('sample.jpg',0)
assert len(img)!=0, 'Picture not found'
IncreasedBrightness=changeBrightness(img,1,60)
DecreasedBrightness=changeBrightness(img,1,-20)
plt.figure(figsize=(20,10))
plt.subplot(2,3,1)
plt.imshow(DecreasedBrightness, cmap='gray')
plt.title('Decreased Brightness',fontsize='20')
plt.axis(False)
plt.subplot(2,3,2)
plt.imshow(img, cmap='gray')
plt.title('Orginal Image',fontsize='20')
plt.axis(False)
plt.subplot(2,3,3)
plt.imshow(IncreasedBrightness, cmap='gray')
plt.title('Increased Brightness',fontsize='20')
plt.axis(False)
plt.subplot(2,3,4)
plt.hist(DecreasedBrightness.ravel(),bins=256, fc='k', ec='k')
plt.subplot(2,3,5)
plt.hist(img.ravel(),bins=256, fc='k', ec='k')
plt.subplot(2,3,6)
plt.hist(IncreasedBrightness.ravel(),bins=256, fc='k', ec='k')
plt.show()
```



1.2 Thresholding

$$T(i) = \begin{cases} V_{min} & i < t \\ V_{max} & i > t \end{cases}$$

[3]:
```
def threshold(img,t,vMin=0,vMax=255):
    outImg=img.copy()
    outImg[outImg<=t]=vMin
    outImg[outImg>t]=vMax
    return outImg
OrginalImg=cv2.imread('sample.jpg',0)
t=100
ThresholdImg=threshold(OrginalImg,t)
plt.figure(figsize=(12,10))
plt.subplot(1,2,1)
plt.title('Orginal Image',fontsize='20')
plt.imshow(OrginalImg ,cmap = 'gray')
plt.axis(False)
plt.subplot(1,2,2)
plt.title(f'Thresholding {t}', fontsize='20')
plt.imshow(ThresholdImg, cmap = 'gray')
plt.axis(False)
plt.show()
```



1.3 Linear Stretch

$$T(x) = \begin{cases} v_{min} & x < a \\ f(x) & a \leqslant x \leqslant b \\ v_{max} & x > b \end{cases}$$

$$f(x) = mx + n \Longrightarrow \begin{cases} ma + n = v_{min} \\ mb + n = v_{max} \end{cases} \Longrightarrow \begin{cases} m = \frac{v_{max} - v_{min}}{b-a} \\ n = \frac{bv_{min} - av_{max}}{b-a} \end{cases}$$

To increase the calculation time lets avoid `conditions` as we know

$$Heaviside(x, a) = \begin{cases} 0 & x < 0 \\ a & x = 0 \\ 1 & x > 0 \end{cases}$$

using `np.heaviside` function we can rewrite T function as well as:

$$T(x) = \begin{cases} v_{min} & x < a \\ f(x) & a \leqslant x \leqslant b \\ v_{max} & x > b \end{cases} = \begin{cases} v_{min} & 0 < a - x \\ f(x) & (0 \leqslant x - a \leqslant b - a \equiv\ 0 \leqslant x - a \ And\ x - a \leqslant b - a \\ v_{max} & x - b > 0 \end{cases}$$

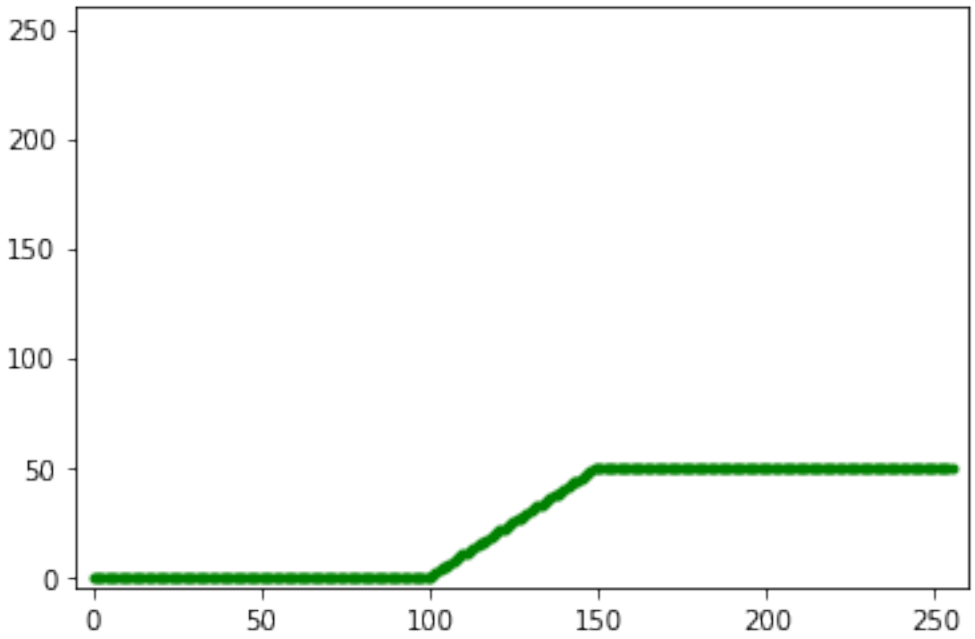
$$= \begin{cases} v_{min} & H(a - x, 0) = 1 \\ f(x) & H(x - a) \times H(\ b - x) = 1 \\ v_{max} & H(x - b) = 1 \end{cases}$$

$$\implies \boxed{T(x) = v_{min} \times H(a - x, 0) + v_{max} \times H(x - b, 0) + (mx + n) \times H(x - a, 1) \times H(b - x, 1)}$$

[4]:
```
def linearStretch(img,a,b,Vmin,Vmax):
    if a>b: a,b=b,a
    if Vmin>Vmax: Vmin,Vmax=Vmax,Vmin
    m=(Vmax-Vmin)/(b-a)
    n=(b*Vmin-a*Vmax)/(b-a)
    return np.vectorize(lambda x,a,b,m,n,Vmin,Vmax:Vmin*np.heaviside(a-x,0)\
                                                   +Vmax*np.heaviside(x-b,0)\
                                                   +(m*x+n)*np.heaviside(x-a,1)\
                                                           *np.heaviside(b-x,1))\
           (img.astype(np.int16),a,b,m,n,Vmin,Vmax).astype(np.uint8)
print(",  ".join(f"{i}->{j}" for i,j in zip(np.arange(256),
                                            linearStretch(np.arange(256),
                                                          100,150,0,50)
                                           )
               )
     )
plt.plot(np.arange(256),linearStretch(np.arange(256),100,150,0,50),'g.')
plt.xlim(-5,260)
plt.ylim(-5,260)
plt.show()
```

```
0->0,  1->0,  2->0,  3->0,  4->0,  5->0,  6->0,  7->0,  8->0,  9->0,  10->0,
11->0,  12->0,  13->0,  14->0,  15->0,  16->0,  17->0,  18->0,  19->0,  20->0,
21->0,  22->0,  23->0,  24->0,  25->0,  26->0,  27->0,  28->0,  29->0,  30->0,
31->0,  32->0,  33->0,  34->0,  35->0,  36->0,  37->0,  38->0,  39->0,  40->0,
41->0,  42->0,  43->0,  44->0,  45->0,  46->0,  47->0,  48->0,  49->0,  50->0,
51->0,  52->0,  53->0,  54->0,  55->0,  56->0,  57->0,  58->0,  59->0,  60->0,
61->0,  62->0,  63->0,  64->0,  65->0,  66->0,  67->0,  68->0,  69->0,  70->0,
71->0,  72->0,  73->0,  74->0,  75->0,  76->0,  77->0,  78->0,  79->0,  80->0,
81->0,  82->0,  83->0,  84->0,  85->0,  86->0,  87->0,  88->0,  89->0,  90->0,
91->0,  92->0,  93->0,  94->0,  95->0,  96->0,  97->0,  98->0,  99->0,  100->0,
101->1,  102->2,  103->3,  104->4,  105->5,  106->6,  107->7,  108->8,  109->9,
110->10,  111->11,  112->12,  113->13,  114->14,  115->15,  116->16,  117->17,
118->18,  119->19,  120->20,  121->21,  122->22,  123->23,  124->24,  125->25,
126->26,  127->27,  128->28,  129->29,  130->30,  131->31,  132->32,  133->33,
134->34,  135->35,  136->36,  137->37,  138->38,  139->39,  140->40,  141->41,
```

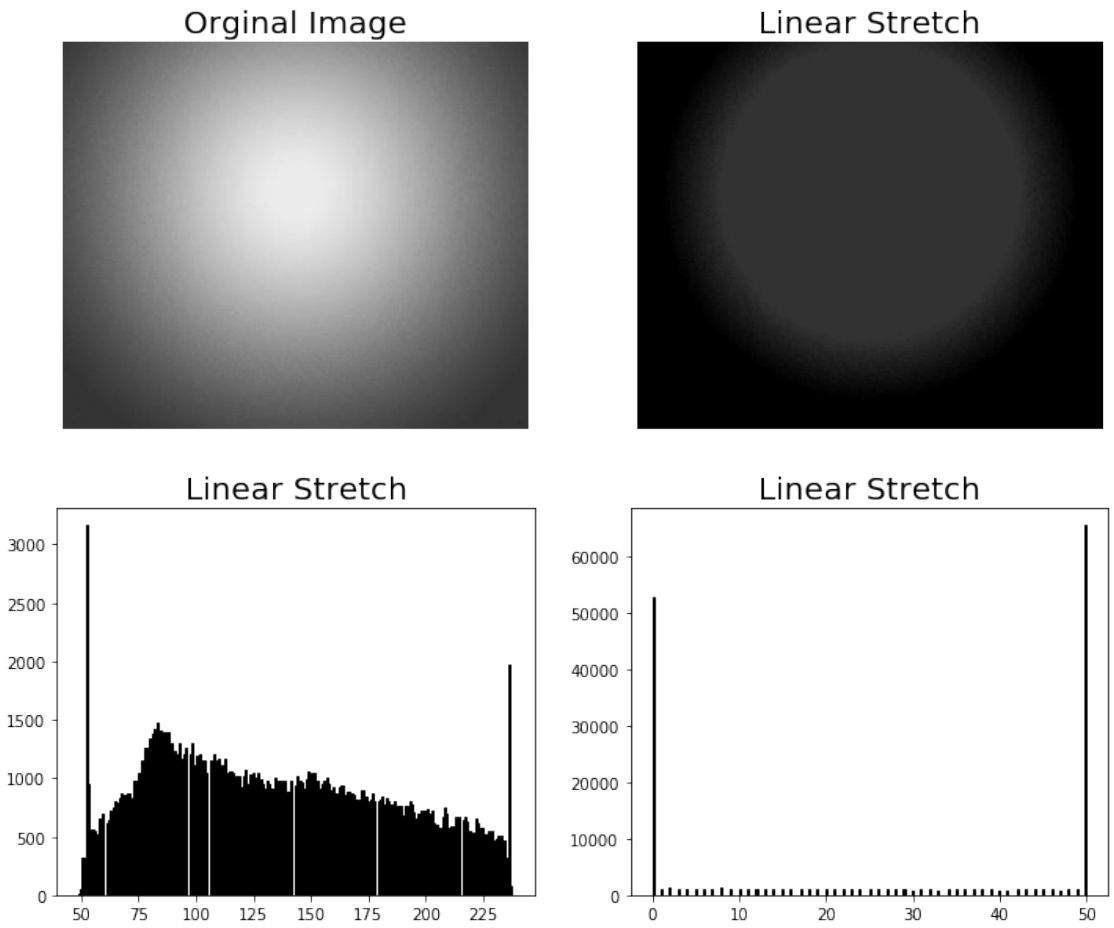
```
142->42,  143->43,  144->44,  145->45,  146->46,  147->47,  148->48,  149->49,
150->50,  151->50,  152->50,  153->50,  154->50,  155->50,  156->50,  157->50,
158->50,  159->50,  160->50,  161->50,  162->50,  163->50,  164->50,  165->50,
166->50,  167->50,  168->50,  169->50,  170->50,  171->50,  172->50,  173->50,
174->50,  175->50,  176->50,  177->50,  178->50,  179->50,  180->50,  181->50,
182->50,  183->50,  184->50,  185->50,  186->50,  187->50,  188->50,  189->50,
190->50,  191->50,  192->50,  193->50,  194->50,  195->50,  196->50,  197->50,
198->50,  199->50,  200->50,  201->50,  202->50,  203->50,  204->50,  205->50,
206->50,  207->50,  208->50,  209->50,  210->50,  211->50,  212->50,  213->50,
214->50,  215->50,  216->50,  217->50,  218->50,  219->50,  220->50,  221->50,
222->50,  223->50,  224->50,  225->50,  226->50,  227->50,  228->50,  229->50,
230->50,  231->50,  232->50,  233->50,  234->50,  235->50,  236->50,  237->50,
238->50,  239->50,  240->50,  241->50,  242->50,  243->50,  244->50,  245->50,
246->50,  247->50,  248->50,  249->50,  250->50,  251->50,  252->50,  253->50,
254->50,  255->50
```



[5]:
```
OrginalImg=cv2.imread('grayGradiant.jpg',0)
start=time()
StretchedImg=linearStretch(OrginalImg,100,150,0,50)
print(f"the time elapsed for calculation is: {time()-start}")
plt.figure(figsize=(12,10))
plt.subplot(2,2,1)
plt.title('Orginal Image',fontsize='20')
plt.imshow(OrginalImg,cmap='gray',vmin=0,vmax=255)
plt.axis(False)
plt.subplot(2,2,2)
```

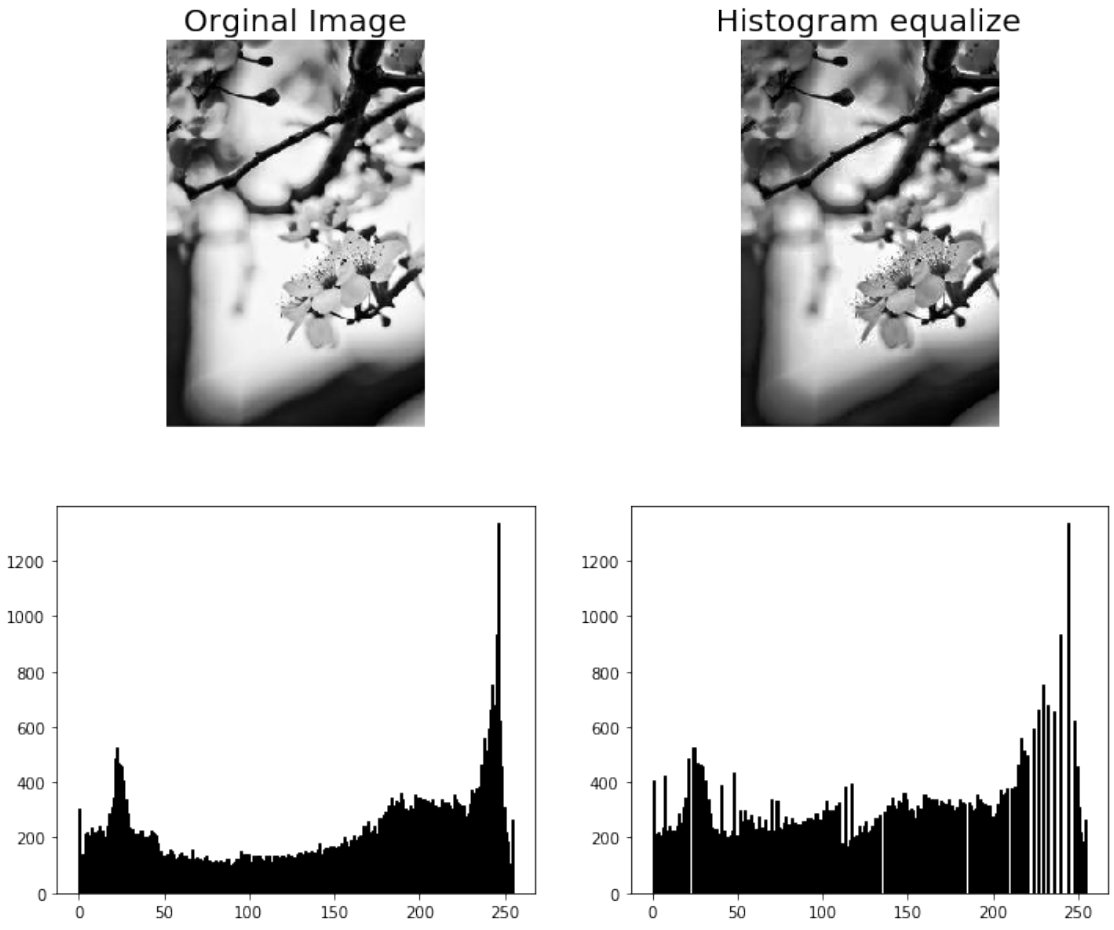
```
plt.title('Linear Stretch', fontsize='20')
plt.imshow(StretchedImg,cmap='gray',vmin=0,vmax=255)
plt.axis(False)
plt.subplot(2,2,3)
plt.title('Linear Stretch', fontsize='20')
plt.hist(OrginalImg.ravel(),bins=256, fc='k', ec='k')
plt.subplot(2,2,4)
plt.title('Linear Stretch', fontsize='20')
plt.hist(StretchedImg.ravel(),bins=256, fc='k', ec='k')
plt.show()
```

```
the time elapsed for calculation is: 1.8074896335601807
```



[6]:
```
OrginalImg=cv2.imread('sample.jpg',0)
equalizeHist = cv2.equalizeHist(OrginalImg)
plt.figure(figsize=(12,10))
plt.subplot(221)
plt.title('Orginal Image',fontsize='20')
```

```
plt.imshow(OrginalImg ,cmap = 'gray')
plt.axis('off')
plt.subplot(222)
plt.title('Histogram equalize', fontsize='20')
plt.imshow(equalizeHist, cmap = 'gray')
plt.axis('off')
plt.subplot(223)
plt.hist(OrginalImg.ravel(),bins=256, fc='k', ec='k')
plt.subplot(224)
plt.hist(equalizeHist.ravel(),bins=256, fc='k', ec='k')
plt.show()
```



[ ]:
```
Finished :)
```