Republic of Iraq
Baghdad University
Collage of Science
Computer department

Low Image Lminance Ehancement Bsed Histogram Modification Techniques

A project submitted to
the collage of science, university of Baghdad
Department of computer science
as a partial fulfillment of the of the requirement for the degree Bask in computer science.

By

Duaa A.j Mohammed
Amna Azher Selman

Supervised by
Ass. Prof. Dr. Faisel Ghazee Mohammed

May/2009

بِسْمِ اللّهِ الرَّحْمَنِ الرَّحِيمِ

"وَإِذَا سَأَلَكَ عِبَادِي عَنِّي فَإِنِّي قَرِيبٌ أُجِيبُ دَعْوَةَ الدَّاعِ إِذَا دَعَانِ فَلْيَسْتَجِيبُواْ لِي وَلْيُؤْمِنُواْ بِي لَعَلَّهُمْ يَرْشُدُونَ"

سورة البقرة اية رقم186

اهداء ...

لا يسعنا ونحن نقدم هذا البحث الا ان نتقدم بالشكر الجزيل والوافر للدكتور فيصل غازي لما ابداه من دعم وتشجيع ومساندة في انجاز مشروع البحث كما نتقدم بالشكر الجزيل لقسم الحاسبات وكل من ساعدنا في انجاز هذا البحث راجين من الله العلي القدير ان يحفظهم ويوفقهم لما فيه خير البلاد

الطالبتان :

دعاء عبد الجبار محمد

آمنة ازهر سلمان

TABELS OF CONTENTS

References

[1] Scott E. Umbaugh "Computer Vision and Image processing" prentice hall[1998]

[2] http:/www.cambridgeincolour.com/tutorials/histogram.htm

[3] http:/www.prenhall.com/divisions/ptr

CHAPTER ONE

INTRODUCTION

Image enhancement processes consist of a collection of techniques that seek to Improve the visual appearance of an image or to convert the image to a form better Suited for analysis by a human or a machine. In an image enhancement system, there Is no conscious effort to improve the fidelity of a reproduced image with regard to some ideal form of the image, as is done in image restoration. Actually, there is some evidence to indicate that often a distorted image, for example, an image with amplitude overshoot and undershoot about its object edges, is more subjectively pleasing than a perfectly reproduced original.

For image analysis purposes, the definition of image enhancement stops short of information extraction. As an example, an image enhancement system might emphasize the edge outline of objects in an image by high-frequency filtering. This edge-enhanced image would then serve as an input to a machine that would trace the outline of the edges, and perhaps make measurements of the shape and size of the outline. In this application, the image enhancement processor would emphasize salient features of the original image and simplify the processing task of a data extraction machine.

There is no general unifying theory of image enhancement at present because there is no general standard of image quality that can serve as a design criterion for an image enhancement processor. Consideration is given here to a variety of techniques that have proved useful for human observation improvement and image analysis.

1.1. CONTRAST MANIPULATION

One of the most common defects of photographic or electronic images is poor contrast resulting from a reduced, and perhaps nonlinear, image amplitude range. Image contrast can often be improved by amplitude rescaling of each pixel (1,2). Figure (1-1*a)* illustrates a transfer function for contrast enhancement of a typical Continuous amplitude low-contrast image. For continuous amplitude images, the transfer function operator can be implemented by photographic techniques, but it is often difficult to realize an arbitrary transfer function accurately. For quantized amplitude images, implementation of the transfer function is a relatively simple task. However, in the design of the transfer function operator, consideration must be given to the effects of amplitude quantization. With reference to Figure (1-1*b*), suppose that an original image is quantized to *(J)* levels, but it occupies a smaller range. The output image is also assumed to be restricted to (*J)* levels, and the mapping is linear. In the mapping strategy indicated in Figure (1-1*b)*, the output level chosen is that level closest to the

exact mapping of an input level. It is obvious from the diagram that the output image will have unoccupied levels within its range, and some of the gray scale transitions will be larger than in the original image. The latter effect may result in noticeable gray scale contouring. If the output image is quantized to more levels than the input image, it is possible to approach a linear placement of output levels, and hence, decrease the gray scale contouring Effect.

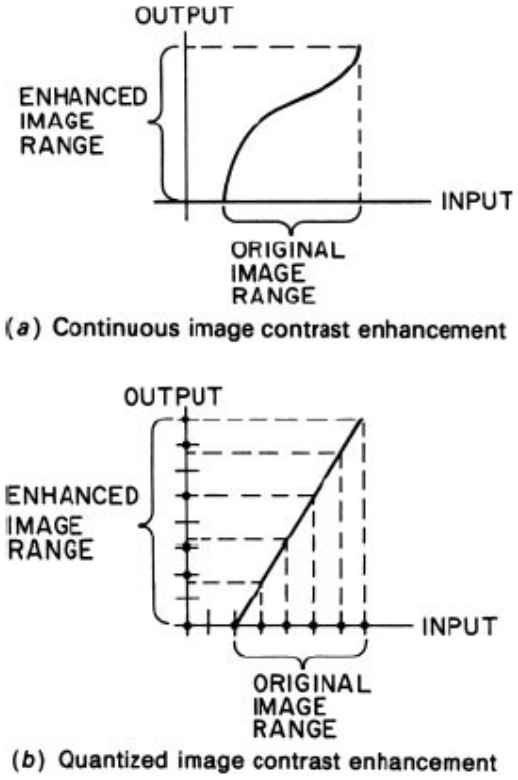
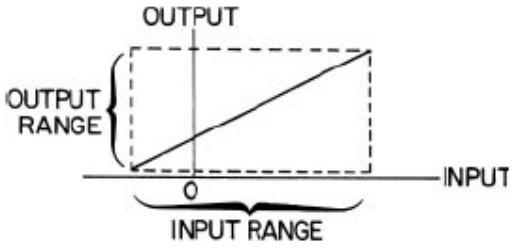
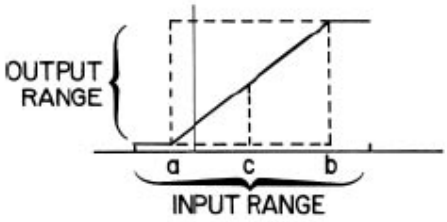


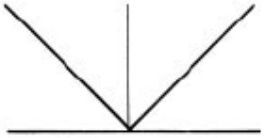
FIGURE.1-1. Continuous and quantized image contrast enhancement.



(*a*) Linear image scaling



(*b*) Linear image scaling with clipping



(*c*) Absolute value scaling

FIGURE1-2. Image scaling methods.

(a) Linear image scaling

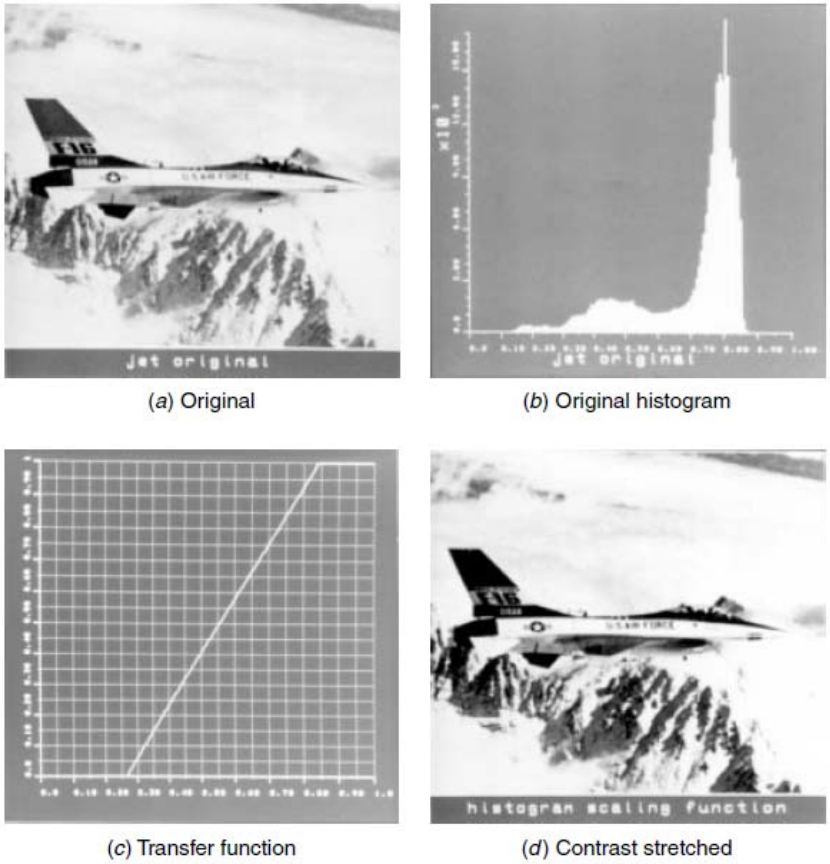
(b) Linear image scaling with clipping

(c) Absolute value scaling

1.2. Contrast Modification

Section 1.1 dealt with amplitude scaling of images that do not properly utilize the dynamic range of a display; they may lie partly outside the dynamic range or occupy only a portion of the dynamic range. In this section, attention is directed to point transformations that modify the contrast of an image within a display's dynamic range.

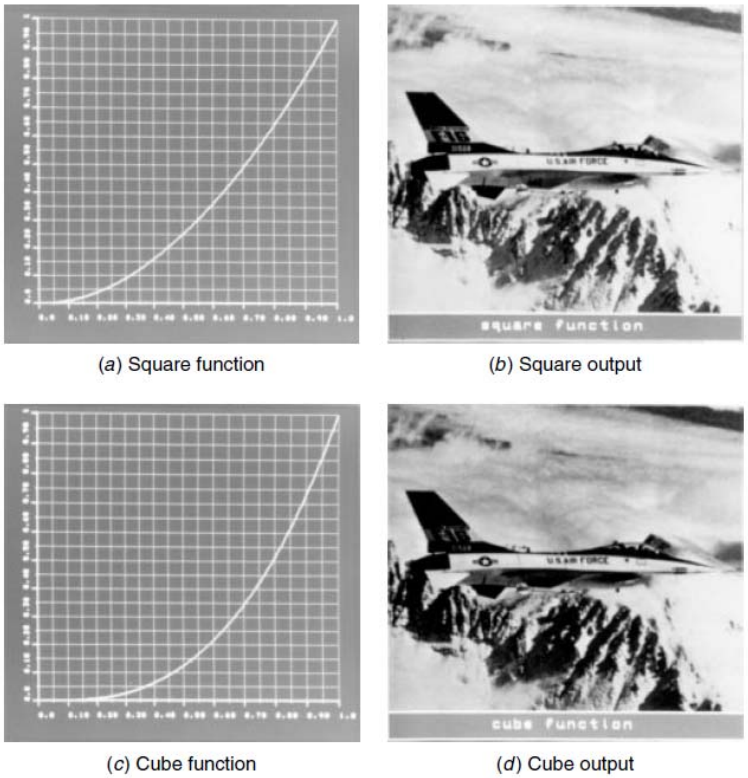
Figure (1-3*a)* contains an original image of a jet aircraft that has been digitized to 256 gray levels and numerically scaled over the range of 0.0 (black) to 1.0 (white).



(*a*) Original (*b*) Original histogram

(*c*) Transfer function (*d*) Contrast stretched

FIGURE 1-3. Window-level contrast stretching of the jet_mon image.



(*a*) Square function (*b*) Square output

(*c*) Cube function (*d*) Cube output

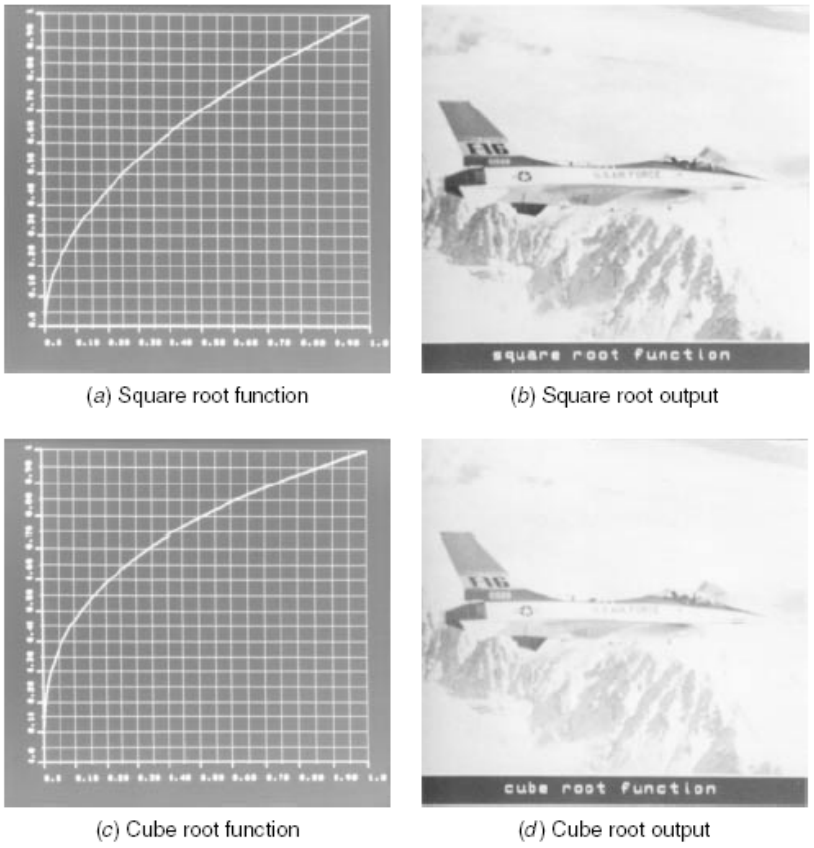
FIGURE 1-4. Square and cube contrast modification of the jet_mon image

The histogram of the image is shown in Figure (1-3*b*). Examination of the histogram of the image reveals that the image contains relatively few low- or high amplitude pixels. Consequently, applying the window-level contrast stretching function of Figure (1-3*c)* results in the image of Figure (1-3*d*), which possesses better visual contrast but does not exhibit noticeable visual clipping.

Consideration will now be given to several nonlinear point transformations, some of which will be seen to improve visual contrast, while others clearly impair visual contrast.

Figures (1-3)and (1-4) provide examples of power law point transformations in which the processed image is defined by

$$G(j,k) = [F(j,k)]^p \tag{1-1}$$



(*a*) Square root function (*b*) Square root output

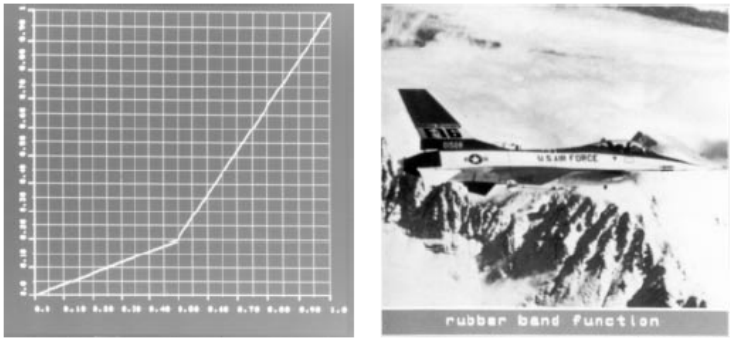
(*c*) Cube root function (*d*) Cube root output

FIGURE 1-5. Square root and cube root contrast modification of the jet_mon image.

Where represents the original image and *p* is the power law variable . It is important that the amplitude limits of Eq. 1-1 be observed; processing of the integer code (e.g., 0 to 255) by Eq. 1-1 will give erroneous results. The square function provides the best visual result. The rubber band transfer function shown in Figure (1-6*a)* provides a simple piecewise linear approximation to the power law curves. It is often useful in interactive enhancement machines in which the inflection point is interactively placed. The Gaussian error function behaves like

a square function for low-amplitude pixels and like a square root function for high- amplitude pixels. It is defined as

$$G(j,k) = \frac{\operatorname{erf}\left\{\dfrac{F(j,k)-0.5}{a\sqrt{2}}\right\} + \dfrac{0.5}{a\sqrt{2}}}{2\ \operatorname{erf}\left\{\dfrac{0.5}{a\sqrt{2}}\right\}} \qquad (1\text{-}2\text{-a})$$



(*a*) Rubber-band function (*b*) Rubber-band output

FIGURE 1-6. Rubber-band contrast modification of the jet_mon image

Where

$$\operatorname{erf}\{x\} = \frac{2}{\sqrt{\pi}} \int_0^x \exp\{-y^2\}\, dy \qquad (1\text{-}2b)$$

and *a* is the standard deviation of the Gaussian distribution.

The logarithm function is useful for scaling image arrays with a very wide dynamic range. The logarithmic point transformation is given by

$$G(j,k) = \frac{\log_e\{1.0 + aF(j,k)\}}{\log_e\{2.0\}} \qquad (1\text{-}3)$$

under the assumption that where *a* is a positive scaling factor.

Figure 8.2-4 illustrates the logarithmic transformation applied to an array of Fourier

transform coefficients.

There are applications in image processing in which monotonically decreasing and nonmonotonic amplitude scaling is useful. For example, contrast reverse and contrast inverse transfer functions, as illustrated in Figure 10.1-9, are often helpful in visualizing detail in dark areas of an image. The reverse function is defined as

$$G(j,k) = 1.0 - F(j,k) \tag{1-4}$$

1.3. HISTOGRAM MODIFICATION

The luminance histogram of a typical natural scene that has been linearly quantized is usually highly skewed toward the darker levels; a majority of the pixels possess a luminance less than the average. In such images, detail in the darker regions is often not perceptible. One means of enhancing these types of images is a technique called *histogram modification,* in which the original image is rescaled so that the histogram of the enhanced image follows some desired form. Andrews, Hall, and others (3–5) have produced enhanced imagery by a *histogram equalization* process for which the histogram of the enhanced image is forced to be uniform. Frei (6) has explored the use of histogram modification procedures that produce enhanced images possessing exponential or hyperbolic-shaped histograms. Ketcham (7) and Hummel (8) have demonstrated improved results by an adaptive histogram modification procedure.

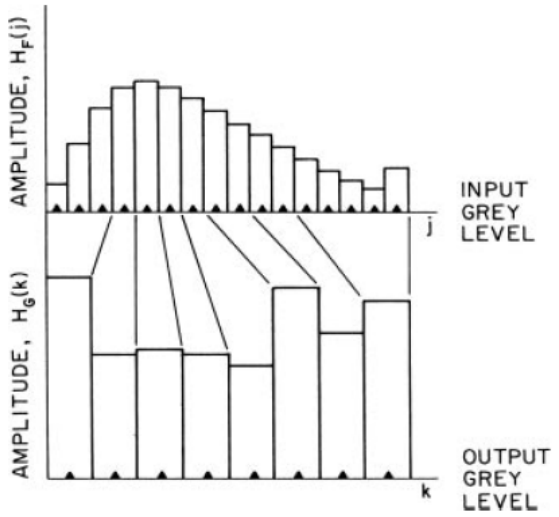


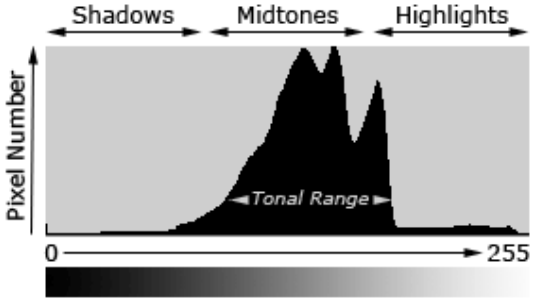
FIGURE 1.3. Approximate gray level histogram equalization with unequal number of quantization levels.

CHAPTER TWO

IMAGE HISTOGRAMS

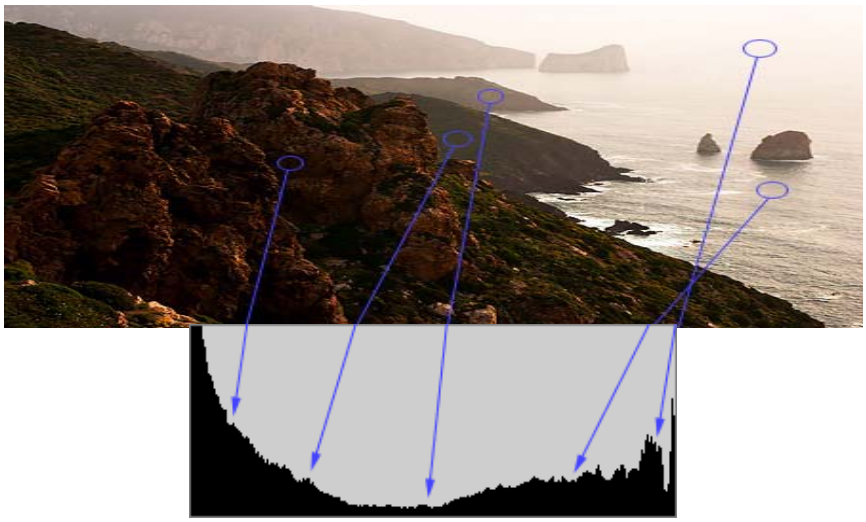
Understanding image histograms is probably the single most important concept to become familiar with when working with pictures from a digital camera. A histogram can tell you whether or not your image has been properly exposed, whether the lighting is harsh or flat, and what adjustments will work best. It will not only improve your skills on the computer, but as a photographer as well.

Each pixel in an image has a color which has been produced by some combination of the primary colors red, green, and blue (RGB). Each of these colors can have a brightness value ranging from 0 to 255 for a digital image with a bit depth of 8-bits. A RGB histogram results when the computer scans through each of these RGB brightness values and counts how many are at each level from 0 through 255. Other types of histograms exist, although all will have the same basic layout as the histogram example shown below.



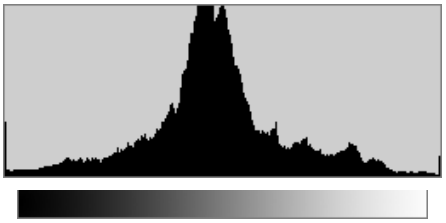
The region where most of the brightness values are present is called the "tonal range." Tonal range can vary drastically from image to image, so developing an intuition for how numbers map to actual brightness values is often critical—both before and after

the photo has been taken. There is no one "ideal histogram" which all images should try to mimic; histograms should merely be representative of the tonal range in the scene and what the photographer wishes to convey.



The above image is an example which contains a very broad tonal range, with markers to illustrate where regions in the scene map to brightness levels on the histogram. This coastal scene contains very few midtones, but does have plentiful shadow and highlight regions in the lower left and upper right of the image, respectively. This translates into a histogram which has a high pixel count on both the far left and right-hand sides.

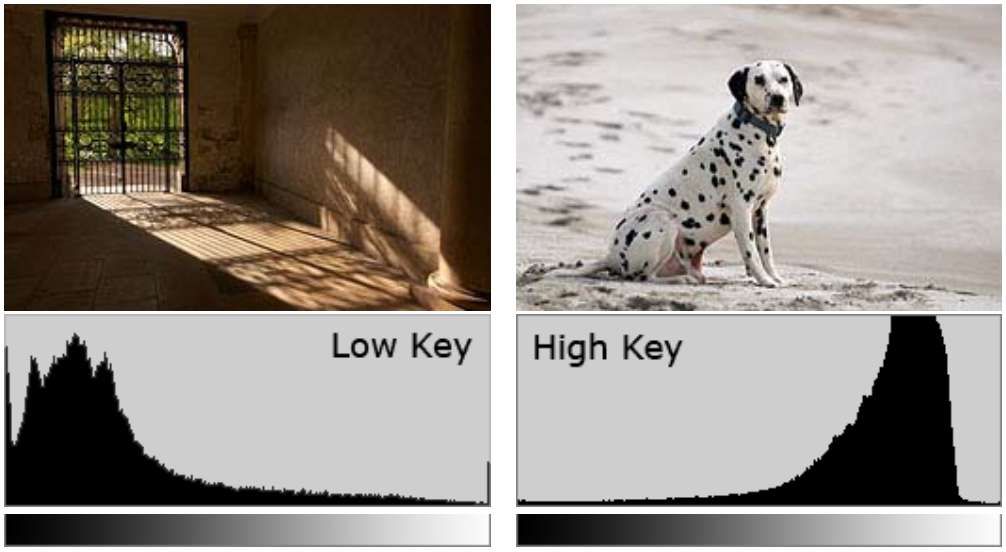
Lighting is often not as extreme as the last example. Conditions of ordinary and even lighting, when combined with a properly exposed subject, will



usually produce a histogram which peaks in the centre, gradually tapering off into the shadows and highlights. With the exception of the direct sunlight reflecting off the top of the building and off some windows, the boat scene to the right is quite evenly lit. Most cameras will have no trouble automatically reproducing an image which has a histogram similar to the one shown below

HIGH AND LOW KEY IMAGES

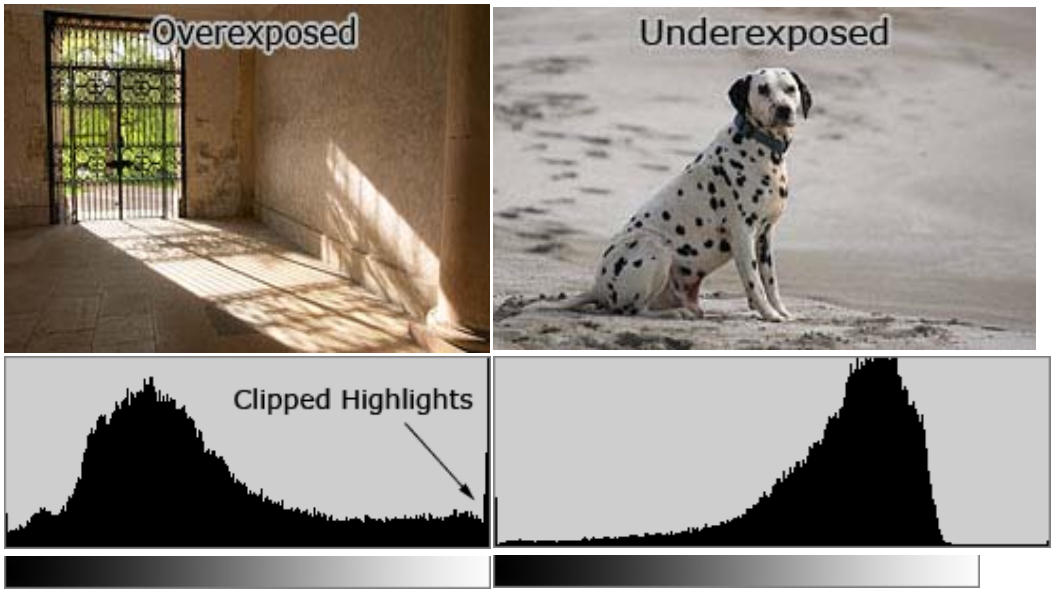
Although most cameras will produce midtone-centric histograms when in an automatic exposure mode, the distribution of peaks within a histogram also depends on the tonal range of the subject matter. Images where most of the tones occur in the shadows are called "low key," whereas with "high key" images most of the tones are in the highlights.



Before the photo has been taken, it is useful to assess whether or not your subject matter qualifies as high or low key. Since cameras measure reflected as opposed to incident light, they are unable to assess the absolute brightness of their subject. As a result, many cameras contain sophisticated algorithms which try

to circumvent this limitation, and estimate how bright an image should be. These estimates frequently result in an image whose average brightness is placed in the midtones. This is usually acceptable, however high and low key scenes frequently require the photographer to manually adjust the exposure, relative to what the camera would do automatically. A good rule of thumb is that you will need to manually adjust the exposure whenever you want the average brightness in your image to appear brighter or darker than the midtones.

The following set of images would have resulted if I had used my camera's auto exposure setting. Note how the average pixel count is brought closer to the midtones.



Most digital cameras are better at reproducing low key scenes since they prevent any region from becoming so bright that it turns into solid white, regardless of how dark the rest of the image might become as a result. High key scenes, on the other hand, often produce images which are significantly

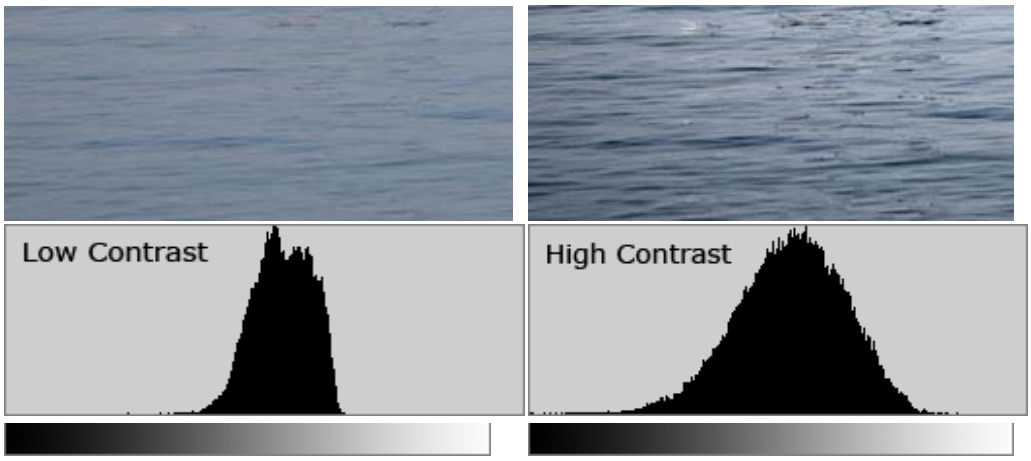
underexposed. Fortunately, underexposure is usually more

forgiving than overexposure (although this compromises your signal to noise ratio). Detail can never be recovered when a region becomes so overexposed that it becomes solid white. When this occurs the highlights are said to be "clipped" or "blown."

The histogram is a good tool for knowing whether clipping has occurred since you can readily see when the highlights are pushed to the edge of the chart. Some clipping is usually ok in regions such as specular reflections on water or metal, when the sun is included in the frame or when other bright sources of light are present. Ultimately, the amount of clipping present is up to the photographer and what they wish to convey.

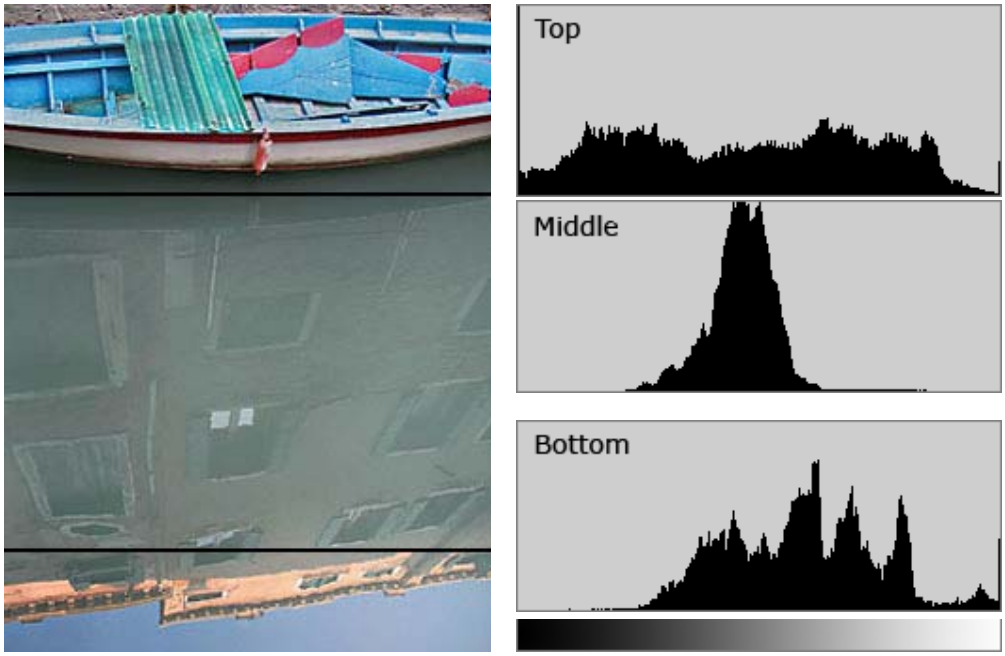
CONTRAST

A histogram can also describe the amount of contrast. Contrast is a measure of the difference in brightness between light and dark areas in a scene. Broad histograms reflect a scene with significant contrast, whereas narrow histograms reflect less contrast and may appear flat or dull. This can be caused by any combination of subject matter and lighting conditions. Photos taken in the fog will have low contrast, while those taken under strong daylight will have higher contrast.



Contrast can have a significant visual impact on an image by emphasizing texture, as shown in the image above. The high contrast water has deeper shadows and more pronounced highlights, creating texture which "pops" out at the viewer.

Contrast can also vary for different regions within the same image due to both subject matter and lighting. We can partition the previous image of a boat into three separate regions—each with its own distinct histogram.



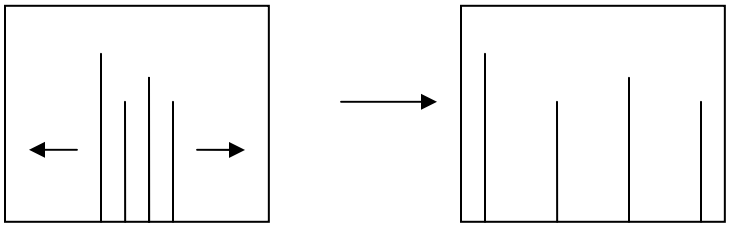
The upper region contains the most contrast of all three because the image is created from light which does not first reflect off the surface of water. This produces deeper shadows underneath the boat and its ledges, and stronger highlights in the upward-facing and directly exposed areas. The middle and bottom regions are produced entirely from diffuse, reflected light and thus have lower contrast; similar to if one were taking photographs in the fog. The bottom region has more contrast than the middle—despite the smooth and monotonic blue sky—because it contains a combination of shade and more intense sunlight. Conditions in the bottom region create more pronounced highlights, but it still lacks the deep shadows of the top region. The sum of the histograms in all three regions creates the overall histogram shown before.

CHAPTER THREE

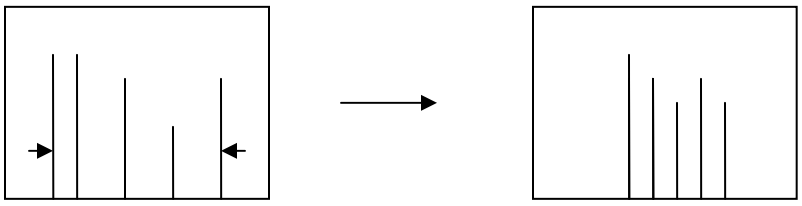
3.1.Histogram Modification

An alternative perspective to gray-level modification that performs a similar function is referred to as histogram modification. The gray –level histogram of an image is the distribution of the gray levels in an image.

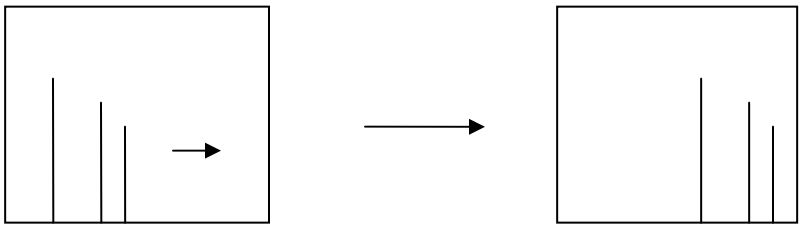
In general, a histogram with a small spread has low contrast, and a histogram with a wide spread has high contrast, whereas an image with its histogram clustered at the low end of the range is dark, and a histogram with the values clustered at the high end of the range corresponds to a bright image (see figure).



a Histogram stretch



b Histogram shrink



c Histogram slide

1.1 HISTOGRAM STRETCH

The histogram can also be modified by a mapping function, which will either stretch, shrink(compress), or slide the histogram.

Histogram stretching and histogram shrinking are forms of gray-scale modification, sometimes referred to as *histogram scaling*.

The mapping function for a histogram stretch can be found by the following

equation:

streach(I(r,c))= [(I(r,c)-I(r,c) min)/(I(r,c)max-I(r,c)min)]

where

I(r,c)max is the largest gray-level value in the image I(r,c)
I(r,c)min is the smallest gray-level value in I(r,c)
Max and min correspond to the maximum and minimum gray-level values
Possible(for an 8-bit image these are 0 and 255)

This equation will take an image and stretch the histogram across the entire gray-level range, which has the effect of increasing the contrast of a low contrast image.
If a stretch is desired over a smaller range, different max and min values can be specified.
If most of the pixel values in a image fall within small range, but a few outliers force the histogram to span the entire range, a pure histogram stretch will not improve the image.
In this case it is useful to allow a small percentage of the pixel values to be clipped at the low and high end of the range.

3.2 HISTOGRAM SHRINK

The opposite of a histogram stretch is a histogram shrink, which will decrease image contrast by compressing the gray levels.

The mapping function for a histogram shrink can be found by the following equation:

$$\text{Shrink}\,(I(r,c)) = \left[\frac{shrink_{\max} - shrink_{\min}}{I(r,c)_{\max} - I(r,c)_{\min}}\right]\left[I(r,c) - I(r,c)_{\min}\right] + shrink_{\min}$$

Where

$I(r,c)_{\max}$ is the largest gray-level value in the image $I(r,c)$

$I(r,c)_{\max}$ is the smallest gray- level value in $I(r,c)$

Shrink max and Shrink min correspond to the maximum and minimum desired in the Compressed histogram .

In general, this process produces an image of reduces contrast and may not seem to be useful as an image enhancement tool. However, we will see an image-sharpening algorithm that uses the histogram shrink process as part of an enhancement technique.

3.3Histogram Slide :

The histogram slide technique can be used to make an image either darker or lighter but retain the relationship between gray-level values. This can be accomplished by Simply adding or subtracting a fixed number from all the gray-level values, as follows:

$$slide(I(r,c)) = I(r,c) + offset$$

where the offset value is the amount to slide the histogram. In this equation we assume that any values slide past the minimum and maximum values will be clipped to the respective minimum or maximum .A positive offset value will increase the overall brightness, where as a negative offset will create a darker image.

CHAPTER FOUR

The code

```
Dim Streth(), Shrink(), slide()
Dim maxred, minred
--------------------------------------------------------------------------------
Private Sub eixt1_Click ()
End
End Sub
--------------------------------------------------------------------------------
Private Sub Form_Load()

  Label1 = "original image"
  Label3 = "image after processing"
  Label5 = "original histogram"
  Label6 = "histogram after processing"
End Sub
--------------------------------------------------------------------------------
Private Sub equalization_Click()

 maxred = 0
 minred = 200

 Size = Wid * Hgt

 ReDim HistR(255)
 ReDim HistG(255)
 ReDim HistB(255)
 ReDim Sum_HistR(0 To 255)
 ReDim sumg_histg(0 To 255)
 ReDim Sumb_Histb(0 To 255)
 1)*************************
  HistR(kr) = 0:
  HistG(kg) = 0:
  HistB(kb) = 0
   For r = 0 To Wid - 1
    For c = 0 To Hgt - 1

     If Red(r, c) < minred Then minred = Red(r, c)
     If Red(r, c) > maxred Then maxred = Red(r, c)

     kr = Red(r, c)
     HistR(kr) = HistR(kr) + 1  'Iteration

     If green(r, c) < minred Then minred = green(r, c)
     If green(r, c) > maxred Then maxred = green(r, c)

     kg = green(r, c)
     HistG(kg) = HistG(kg) + 1 'Iteration

    If Blue(r, c) < minred Then minred = Blue(r, c)
    If Blue(r, c) > maxred Then maxred = Blue(r, c)

   kb = Blue(r, c)
```

```
  HistB(kb) = HistB(kb) + 1 Iteration
   Next
 Next

 2)*******calculate the sum of histogram *********
 gray_level = 255
Sum = 0: Sumg = 0: Sumb = 0
 For i = 0 To gray_level
   Sum = Sum + HistR(i)
   Sum_HistR(i) = Sum
   Sumg = Sumg + HistG(i)
   sumg_histg(i) = Sumg
   Sumb = Sumb + HistB(i)
   Sumb_Histb(i) = Sumb
 Next

'3)*** transform input image to output image

area = Wid * Hgt
dm = gray_level
For i = 0 To 255
  HistR(i) = 0: HistG(i) = 0: HistB(i) = 0
Next i

 For r = 0 To Wid - 1
    For c = 0 To Hgt - 1

     rr = (dm / area) * Sum_HistR(Red(r, c))
     gg = (dm / area) * sumg_histg(green(r, c))
     bb = (dm / area) * Sumb_Histb(Blue(r, c))
     HistR(rr) = HistR(rr) + 1  'Iteration
     HistG(gg) = HistG(gg) + 1 'Iteration
     HistB(bb) = HistB(bb) + 1 'Iteration

     Picture5.PSet (r, c), RGB(rr, 0, 0)
     Picture7.PSet (r, c), RGB(0, gg, 0)
     Picture8.PSet (r, c), RGB(0, 0, bb)
     Picture9.PSet (r, c), RGB(rr, gg, bb)
   Next
  Next
     n = Hgt * Wid
   If maxred < n Then n = maxred
   ratio = CInt(Picture6.ScaleHeight / n)

 ' now plot each bin on the picture box
    Picture6.AutoRedraw = True
    Picture6.CurrentX = 0
    Picture6.CurrentY = ScaleHeight
    Picture6.Cls

    For i = 0 To 255
      Value = Abs(HistR(i) * ratio)
       Picture6.Line -(i + 1, Picture6.ScaleHeight - Value), RGB(200, 0,0)

      Valueg = Abs(HistG(i) * ratio)
```

```
     Picture6.Line -(i + 1, Picture6.ScaleHeight - Valueg), RGB(0,200,0)

     Valueb = Abs(HistB(i) * ratio)
     Picture6.Line -(i + 1, Picture6.ScaleHeight - Valueb), RGB(0,0,200)
   Next i
c = 0
For i = 0 To 9
       c = c + ((i + 1) * (n \ 10))
       Label7(i) = c
   Next i

End Sub

---------------------------------------------------------------------------------------


Private Sub OPen_Click()

  CommonDialog1.InitDir = App.Path
  CommonDialog1.ShowOpen
  fil = CommonDialog1.FileName

  Picture1.Picture = LoadPicture(fil)

 Wid = Picture1.Width
 Hgt = Picture1.Height

 Picture2.Width = Wid: Picture2.Height = Hgt
     Picture3.Width = Wid: Picture3.Height = Hgt
   Picture4.Width = Wid: Picture4.Height = Hgt

 ReDim Red(Wid, Hgt), green(Wid, Hgt), Blue(Wid, Hgt)
 maxred = 0
 minred = 1000
 Size = Wid * Hgt
 ReDim HistR(Size), HistG(Size), HistB(Size)
 ReDim Sum_HistR(0 To 255), Sum_HistG(0 To 255), Sum_HistB(0 To 255)

 For r = 0 To Wid - 1
  For c = 0 To Hgt - 1
   pixel = Picture1.Point(r, c)

    Red(r, c) = Abs(pixel Mod 256)
    green(r, c) = (pixel \ 256) Mod 256
    Blue(r, c) = (pixel \ 65536) Mod 256

     If Red(r, c) < minred Then minred = Red(r, c)
     If Red(r, c) > maxred Then maxred = Red(r, c)

     If green(r, c) < minred Then minred = green(r, c)
     If green(r, c) > maxred Then maxred = green(r, c)

     If Blue(r, c) < minred Then minred = Blue(r, c)
     If Blue(r, c) > maxred Then maxred = Blue(r, c)
```

```
    HistR(Red(r, c)) = HistR(Red(r, c)) + 1    'Iteration
    HistG(green(r, c)) = HistG(green(r, c)) + 1  'Iteration
    HistB(Blue(r, c)) = HistB(Blue(r, c)) + 1  'Iteration


   Picture2.PSet (r, c), RGB(Red(r, c), 0, 0)
   Picture3.PSet (r, c), RGB(0, green(r, c), 0)
   Picture4.PSet (r, c), RGB(0, 0, Blue(r, c))
  Next
 Next
 Picture5.Cls
 Picture6.Cls
 Picture7.Cls
 Picture8.Cls
 Picture9.Cls


  End Sub
---------------------------------------------------------------------------------------

Private Sub plot_Histogram_Click()

    n = Hgt * Wid

 If maxred < n Then n = maxred
    ratio = CInt(picHistogram.ScaleHeight / n)


  'now plot each bin on the picture box
    picHistogram.AutoRedraw = True
    picHistogram.CurrentX = 0
    picHistogram.CurrentY = ScaleHeight

    picHistogram.Cls

    For i = 0 To 255
        Value = Abs(HistR(i) * ratio)
        picHistogram.Line -(i + 1, picHistogram.ScaleHeight - Value), RGB(200, 0, 0)
        Valueg = Abs(HistG(i) * ratio)
        picHistogram.Line -(i + 1, picHistogram.ScaleHeight - Valueg), RGB(0, 200, 0)
        Valueb = Abs(HistB(i) * ratio)
        picHistogram.Line -(i + 1, picHistogram.ScaleHeight - Valueb), RGB(0, 0, 200)
    Next i
c = 0
    For i = 0 To 9
        c = c + ((i + 1) * (n / 10))
        yLabels(i) = c
    Next i
End Sub
---------------------------------------------------------------------------------------

Private Sub review_Click()

Form2.Visible = True
```

```
End Sub
-------------------------------------------------------------------------------

Private Sub slideing_Click()

ReDim Slider(256)
ReDim Slideg(256)
ReDim Slideb(256)
Size = Wid * Hgt
 ReDim HistR(Size), HistG(Size), HistB(Size)
 ReDim Sum_HistR(0 To 255), Sum_HistG(0 To 255), Sum_HistB(0 To 255)

 Offset = Val(InputBox("enter the value offset"))
 maxred = 0: minred = 2000

For r = 0 To Wid - 1
  For c = 0 To Hgt - 1

   Slider(Red(r, c)) = Red(r, c) + Offset

   If ((Slider(Red(r, c)) > 255)) Then Slider(Red(r, c)) = 255
   If ((Slider(Red(r, c)) < 0)) Then Slider(Red(r, c)) = 0

   rr = Slider(Red(r, c))
   If rr < minred Then minred = rr
   If rr > maxred Then maxred = rr

    HistR(rr) = HistR(rr) + 1     'Iteration

   Slideg(green(r, c)) = green(r, c) + Offset
   If ((Slideg(green(r, c)) > 255)) Then Slideg(green(r, c)) = 255
   If ((Slideg(green(r, c)) < 0)) Then Slideg(green(r, c)) = 0
   gg = Slideg(green(r, c))
   If gg < minred Then minred = gg
   If gg > maxred Then maxred = gg
   HistG(gg) = HistG(gg) + 1     'Iteration

   Slideb(Blue(r, c)) = Blue(r, c) + Offset
   If ((Slideb(Blue(r, c)) > 255)) Then Slideb(Blue(r, c)) = 255
   If ((Slideb(Blue(r, c)) < 0)) Then Slideb(Blue(r, c)) = 0

    bb = Slideb(Blue(r, c))
    If bb < minred Then minred = bb
    If bb > maxred Then maxred = bb
    HistB(bb) = HistB(bb) + 1     'Iteration

Picture5.PSet (r, c), RGB(rr, 0, 0)
       Picture7.PSet (r, c), RGB(0, gg, 0)
       Picture8.PSet (r, c), RGB(0, 0, bb)
       Picture9.PSet (r, c), RGB(rr, gg, bb)

       Next
      Next

n = Hgt * Wid
```

```
   If maxred < n Then n = maxred
   ratio = CInt(Picture6.ScaleHeight / n)

 ' now plot each bin on the picture box
   Picture6.AutoRedraw = True
   Picture6.CurrentX = 0
   Picture6.CurrentY = ScaleHeight
   Picture6.Cls


         Value = Abs(HistR(0) * ratio)
        Picture6.PSet (1, Picture6.ScaleHeight - Value), RGB(200, 0, 0)

   For i = 0 To 255

      Value = Abs(HistR(i) * ratio)


     Picture6.Line -(i + 1, Picture6.ScaleHeight - Value), RGB(200, 0, 0)
     Valueg = Abs(HistG(i) * ratio)

   Picture6.Line -(i + 1, Picture6.ScaleHeight - Valueg), RGB(0, 200, 0)     Valueb =
Abs(HistB(i) * ratio)

   Picture6.Line -(i + 1, Picture6.ScaleHeight - Valueb), RGB(0, 0, 200)
   Next i
   c = 0

For i = 0 To 9
      c = c + ((i + 1) * (n \ 10))
      Label7(i) = c
   Next i

End Sub

Private Sub Stretching_Click()

 ReDim Streth(256), Strethg(256), Strethb(256)
 Size = Wid * Hgt
 ReDim HistR(Size), HistG(Size), HistB(Size)
 ReDim Sum_HistR(0 To 255), Sum_HistG(0 To 255), Sum_HistB(0 To 255)

 wminred = Val(InputBox("enter value for minimum red.minimum value shoud be less than
maximum value"))
 wmaxred = Val(InputBox("enter value for maximum red"))

Do While (wminred > wmaxred)
MsgBox ("the value is error")
wminred = Val(InputBox("enter value for minimum red.minimum value shoud be less than
maximum value"))
 wmaxred = Val(InputBox("enter value for maximum red"))
Loop
For i = 0 To 255
  HistR(i) = 0: HistG(i) = 0: HistB(i) = 0
Next i
```

```
For r = 0 To Wid - 1
  For c = 0 To Hgt - 1
    Streth(Red(r, c)) = ((Red(r, c) - minred) / (maxred - minred)) * (wmaxred - wminred) +
wminred
    rr = Streth(Red(r, c))
    Strethg(green(r, c)) = ((green(r, c) - minred) / (maxred - minred)) * (wmaxred -
wminred) + wminred
    gg = Strethg(green(r, c))
    Strethb(Blue(r, c)) = ((Blue(r, c) - minred) / (maxred - minred)) * (wmaxred - wminred)
+ wminred
    bb = Strethb(Blue(r, c))

    HistR(rr) = HistR(rr) + 1     'Iteration
    HistG(gg) = HistG(gg) + 1 'Iteration
    HistB(bb) = HistB(bb) + 1   'Iteration

    Picture5.PSet (r, c), RGB(rr, 0, 0)
     Picture7.PSet (r, c), RGB(0, gg, 0)
      Picture8.PSet (r, c), RGB(0, 0, bb)

      Picture9.PSet (r, c), RGB(rr, gg, bb)
  Next
Next

   n = Hgt * Wid
   If wmaxred < n Then n = wmaxred
   ratio = CInt(Picture6.ScaleHeight / n)

 ' now plot each bin on the picture box
   Picture6.AutoRedraw = True
   Picture6.CurrentX = 0
   Picture6.CurrentY = ScaleHeight
   Picture6.Cls

   For i = 0 To wmaxred
      Value = Abs(HistR(i) * ratio)
       If i = 0 Then
     Picture6.PSet (i + 1, Picture6.ScaleHeight - Value), RGB(200, 0, 0)
      End If
      Picture6.Line -(i, Picture6.ScaleHeight - Value), RGB(200, 0, 0)
      Valueg = Abs(HistG(i) * ratio)
   Picture6.Line -(i + 1, Picture6.ScaleHeight - Valueg), RGB(0, 200, 0)
    Valueb = Abs(HistB(i) * ratio)

   Picture6.Line -(i + 1, Picture6.ScaleHeight - Valueb), RGB(0, 0, 200)
   Next i
c = 0

For i = 0 To 9
      c = c + ((i + 1) * (n \ 10))
      Label7(i) = c
   Next i

 End Sub
```

```
Private Sub Shrinking_Click()


  ReDim Shrink(256), Shrinkg(256), Shrinkb(256)
  Size = Wid * Hgt
  ReDim HistR(Size), HistG(Size), HistB(Size)
  ReDim Sum_HistR(0 To 255), Sum_HistG(0 To 255), Sum_HistB(0 To 255)

  kminred = Val(InputBox("enter value for minimum red.mimimum value should be less
than maximum value"))
  kmaxred = Val(InputBox("enter value for maximum red"))

Do While (kminred > kmaxred)
 MsgBox ("the Value Is Error")
 kminred = Val(InputBox("enter value for minimum red.mimimum value should be less
than maximum value"))
 kmaxred = Val(InputBox("enter value for maximum red"))
Loop

For i = 0 To 255
  HistR(i) = 0:
  HistG(i) = 0:'
  HistB(i) = 0
Next i

For r = 0 To Wid - 1
   For c = 0 To Hgt - 1
    Shrink(Red(r, c)) = ((kmaxred - kminred) / (maxred - minred)) * (Red(r, c) - minred) +
kminred
    rr = Shrink(Red(r, c))

    Shrinkg(green(r, c)) = ((kmaxred - kminred) / (maxred - minred)) * (green(r, c) - minred)
+ kminred
    gg = Shrinkg(green(r, c))

    Shrinkb(Blue(r, c)) = ((kmaxred - kminred) / (maxred - minred)) * (Blue(r, c) - minred)
+ kminred
    bb = Shrinkb(Blue(r, c))

    HistR(rr) = HistR(rr) + 1     'Iteration
    HistG(gg) = HistG(gg) + 1 'Iteration
    HistB(bb) = HistB(bb) + 1   'Iteration

    Picture5.PSet (r, c), RGB(rr, 0, 0)
    Picture7.PSet (r, c), RGB(0, gg, 0)
    Picture8.PSet (r, c), RGB(0, 0, bb)
    Picture9.PSet (r, c), RGB(rr, gg, bb)

   Next
Next

   n = Hgt * Wid
   If kmaxred < n Then n = kmaxred
```

```
   For i = 0 To 9
     c = c + ((i + 1) * (n \ 10))
     Label7(i) = c
   Next i

   ratio = 0.7 'CInt(Picture6.ScaleHeight / n)

 ' now plot each bin on the picture box
   Picture6.AutoRedraw = True
   Picture6.CurrentX = 0
   Picture6.CurrentY = ScaleHeight
   Picture6.Cls

   For i = 0 To 255
       Value = Abs(HistR(i) * ratio)
       If i = 0 Then
      Picture6.PSet (i + 1, Picture6.ScaleHeight - Value), RGB(200, 0, 0)
       End If
     Picture6.Line -(i + 1, Picture6.ScaleHeight - Value), RGB(200, 0, 0)

      Valueg = Abs(HistG(i) * ratio)
   Picture6.Line -(i + 1, Picture6.ScaleHeight - Valueg), RGB(0, 200, 0)

   Valueb = Abs(HistB(i) * ratio)
   Picture6.Line -(i + 1, Picture6.ScaleHeight - Valueb), RGB(0, 0, 200)
   Next i
   c = 0


  ' MsgBox c

End Sub
```