LUT:

8 bit image → 256 entries

1111 1111 = 255 ← 8 bit image
0 to 255 is 256 — 256 values

Treshold:

grayscale image → · if input pixel's gray lvl is > some treshold → output pixel = 1
· otherwise → output pixel = 0

0 = black
255 = white

$$I'(x, y) = \begin{cases} 1 & \text{if } I(x, y) > \tau \\ 0 & \text{otherwise} \end{cases}$$

anything below a certain threshold is black else its white

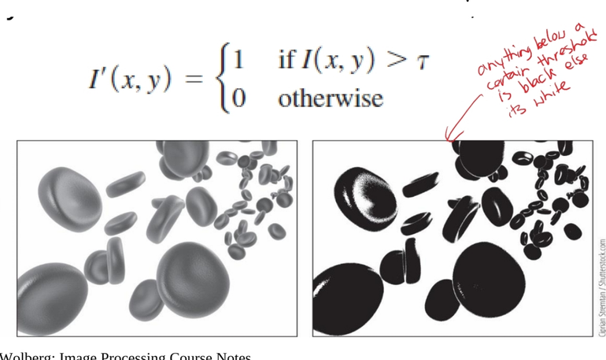


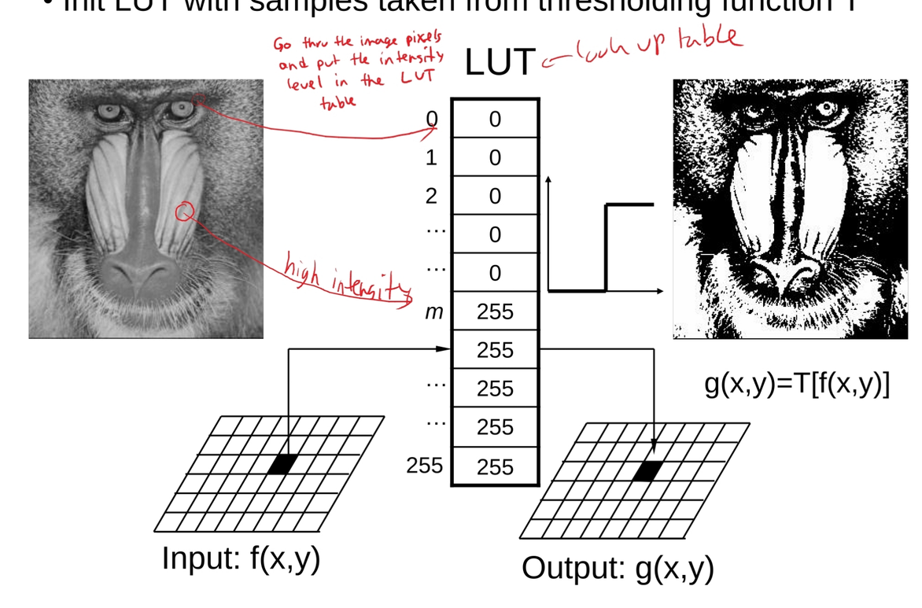
Figure 3.14 An 8-bit grayscale image (left), and the binarized result obtained by thresholding with τ = 150 (right).

Wolberg: Image Processing Course Notes

Lookup table

go thru input image pxl by pxl → store grayscale level on LUT → use the LUT value to make output: F(LUT value of the pixel) → output pixel
thresholding: T(f(x,y)) = g(x,y)

• Init LUT with samples taken from thresholding function T



• Better approach: exploit LUT to avoid **total** comparisons:

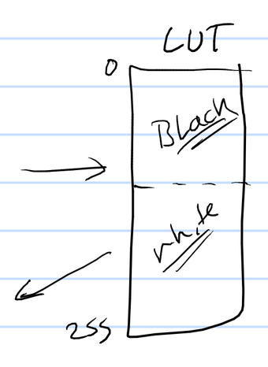
```
// init lookup tables
for(i=0; i<thr; i++) lut[i] = BLACK;
for(; i<MXGRAY; i++) lut[i] = WHITE;

// iterate over all pixels
for(i=0; i<total; i++) out[i] = lut[in[i]];
```

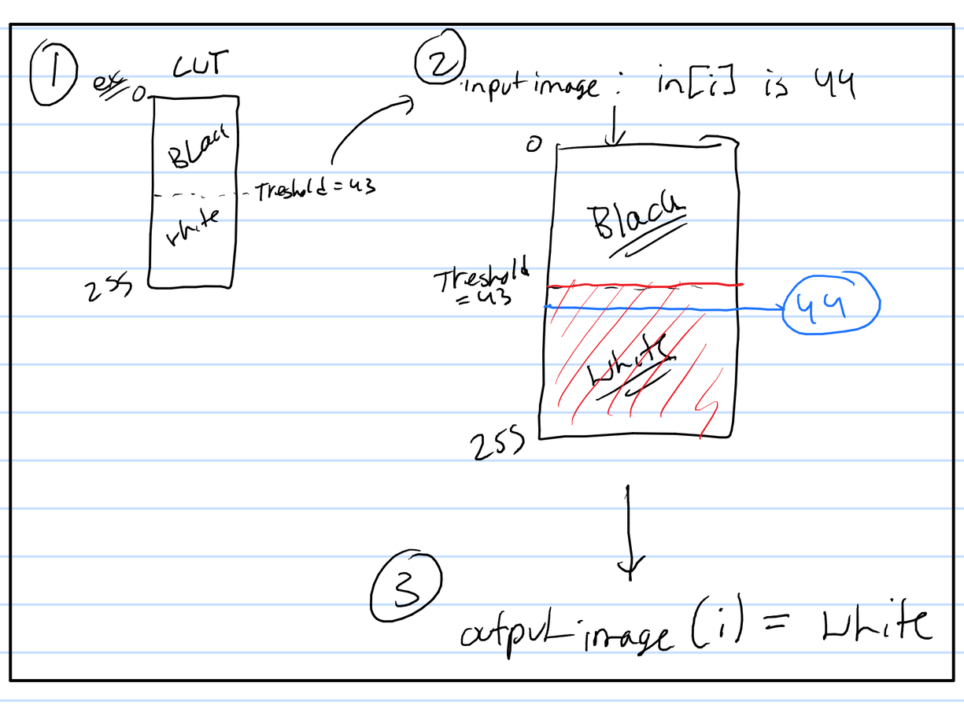
(MXGRAY = 256) — assign black or white to look up table

Smart way

· If less than treshold → BLACK
If more than treshold → White

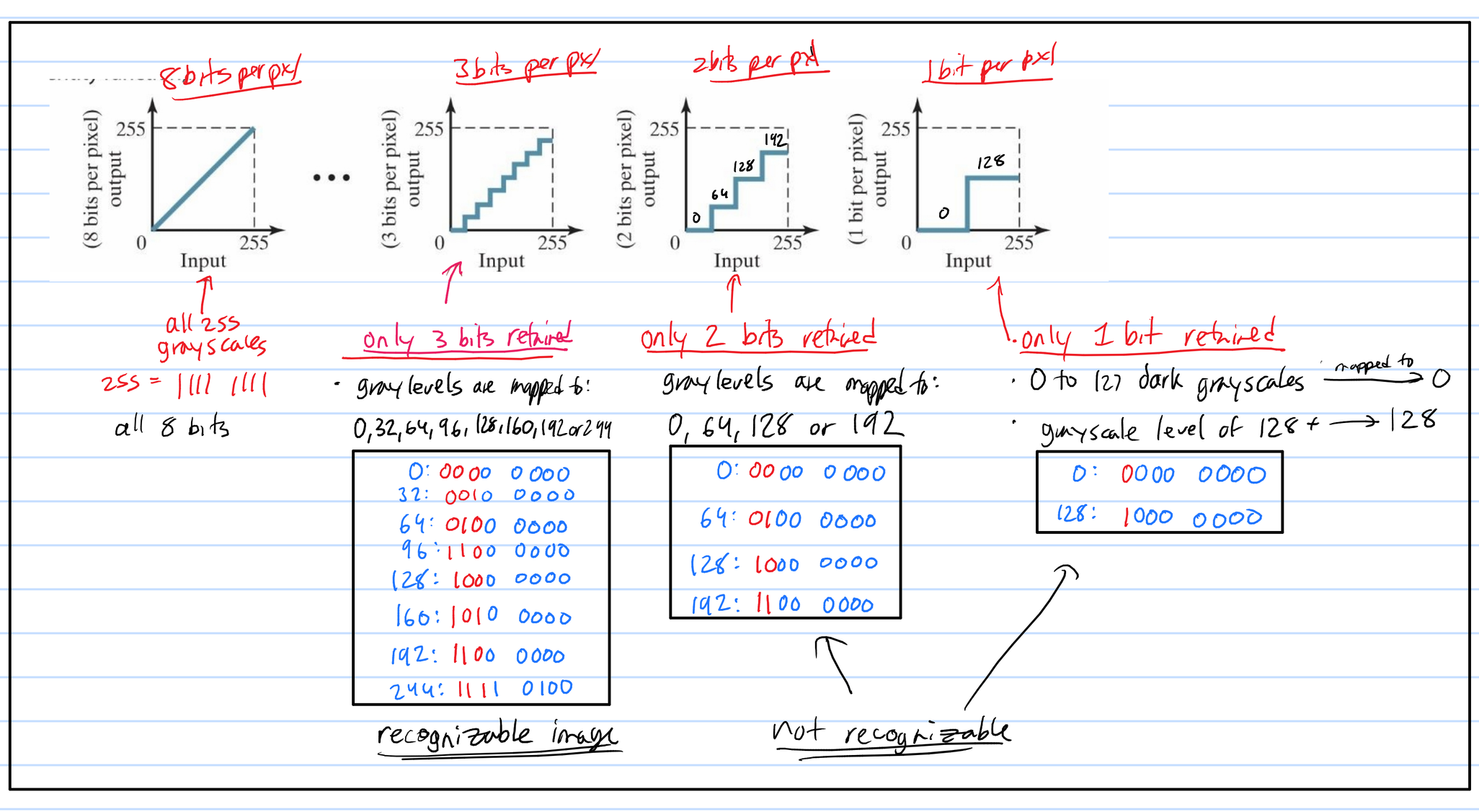
· go over all input pixels and use the value of the input pixel for the LUT



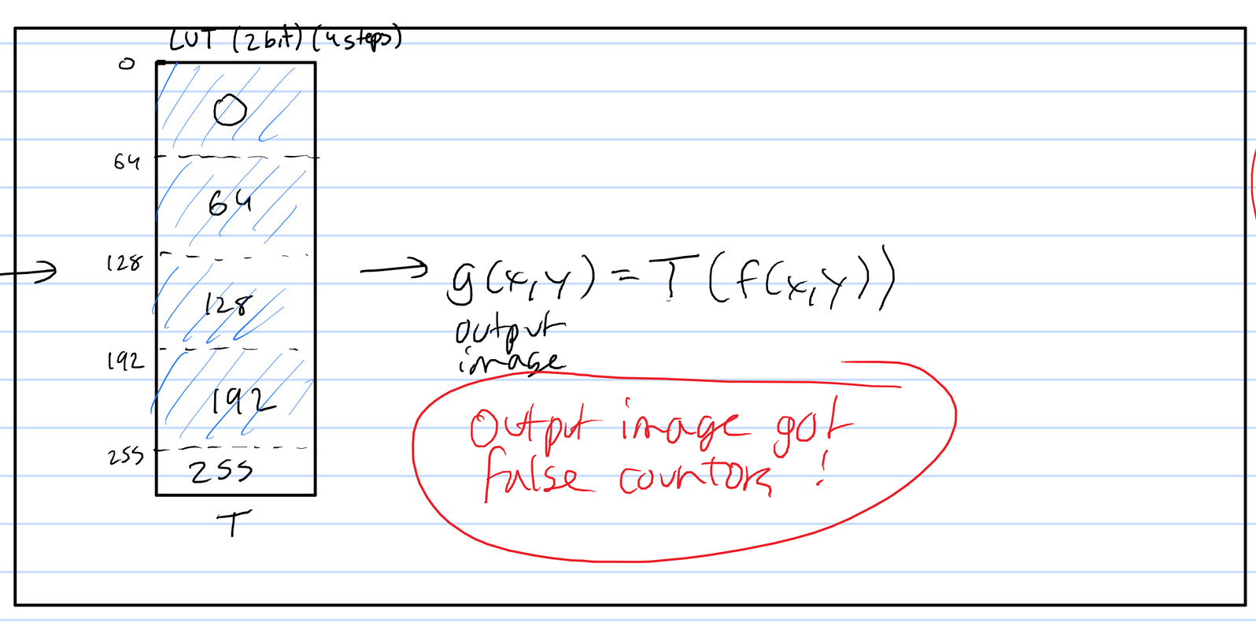


Quantization:

8 bit images takes up space
quantization gets rid of the lower order bits via a staircase func
∴ # of stairs ↑ = # of bits retained ↑



Issue! image → LUT created using 2 bit quantization (4 steps)



With Quantization info will obviously be lost bec we are grouping a lot of values and setting them as one value to save space
→ so will get false contours

code

• Better approach: exploit LUT to avoid **total** mults/divisions:

```
// init lookup tables
scale = MXGRAY / levels;
for(i=0; i<MXGRAY; i++)
       lut[i] = scale * (int) (i/scale);

// iterate over all pixels
for(i=0; i<total; i++) out[i] = lut[in[i]];
```

ex 2bit, 4 step Quantization:
MXGRAY = 256
· scale = 256 levels/4 = 64

① MAKE LUT
· Loop thru all levels 0 → 255:
· for each: scale · int(lut/scale)
ex 201 grayscale
201/64 = 3.14 → int 3
64 · 3 = 192

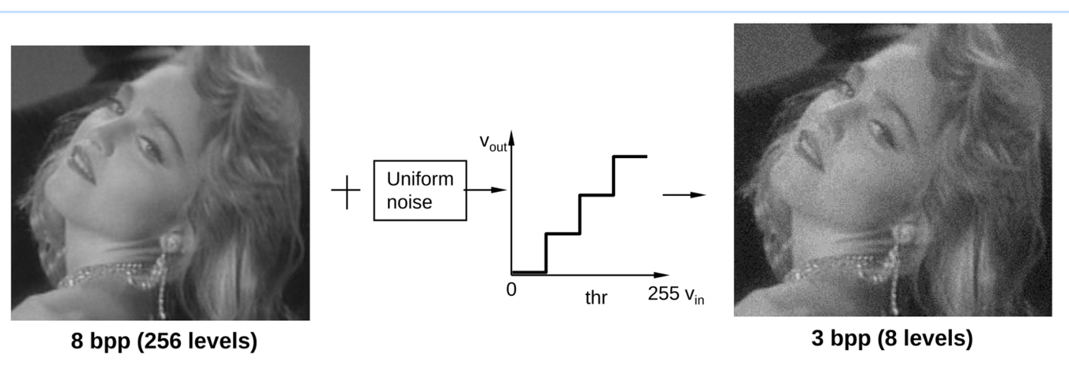
LUT (2bit): 0, 64, 128, 192, 255

② go thru each pixel val in image → use the LUT:
ex input pixel val = 200
in[i] → 200
output is 192
out[i] = lut[in[i] = 200]
= 192

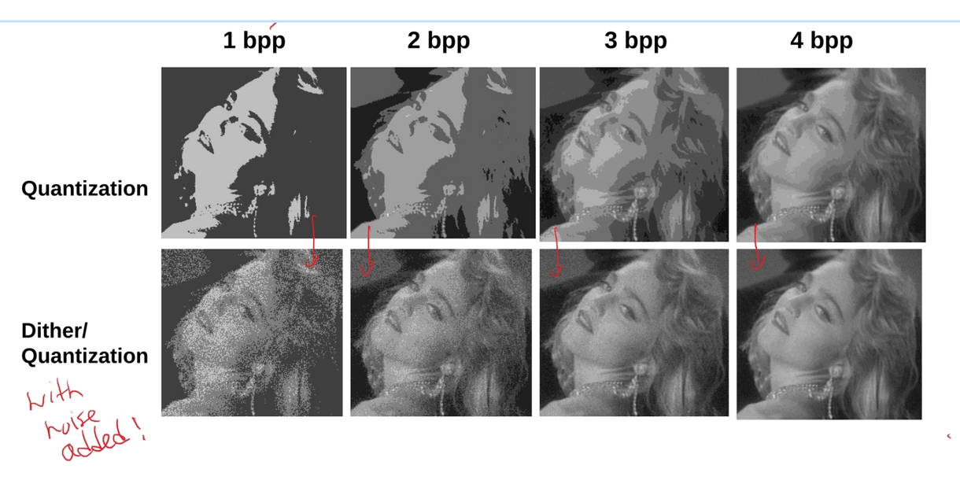
Fix to false contours when quantizing

FIX · add dither (uniformly distributed white noise) to the input image BEFORE quantization → LUT not affected

· dither = some random # from [−m to m)
$m = \frac{MXGRAY}{quantization\text{-}lvls} = \frac{256}{q\text{-}lvls}$



8 bpp (256 levels) 3 bpp (8 levels)

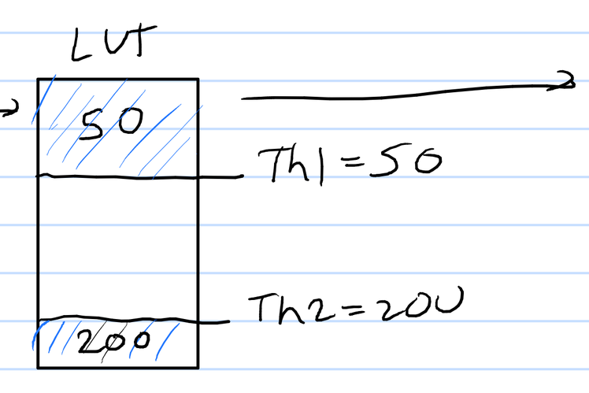


Clipping

th1 = 0
th2 = 255 — default
range of grayscale values

Say I made th1 = 50 and th2 = 200
∴ any grayscale < 50 becomes 50
∴ any grayscale > 200 becomes 200

if in[i] ≤ th1 → out[i] = th1
if in[i] ≥ th2 → out[i] = th2



clipping
gamma — pg 31
contrast — 21
histo stretch → If don't go 0 to 255 this will force it
(t1, t2 parameters)
all intensities < t1 → 0
> t2 → 255
∴ so take t1 and t2 stretch 0 to 255 between them

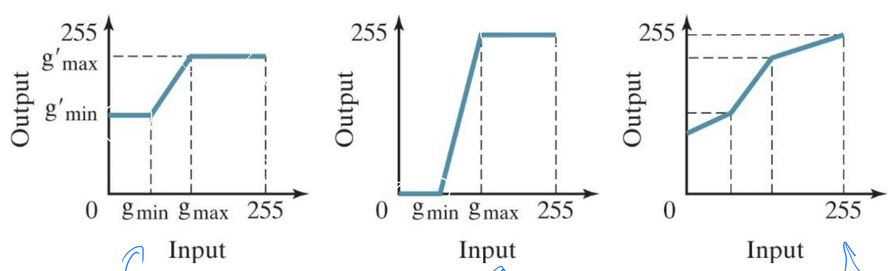
3 d
47 44

66 histmap

histo stretch — 47ish
histo matching — 66–75

Linear Contrast Stretching using Piecewise functions

- Advantage: The form of piecewise functions can be arbitrarily complex
- Disadvantage: Their specification requires considerably more user input



Linear contrast stretch map

stretches it a bit
maps gray lvls from g_{min} to g_{max} to g'_{min} to g'_{max}

crazy contrast stretch
below g_{min} gray lvl → output gray = 0
above g_{max} gray lvl → output gray = 255
from g_{min} to g_{max} → FULL RANGE 0 to 255
∴ if gray lvl not within g_{min} to g_{max} → its 0 or 255

piecewise linear constrast stretch

low contrast image

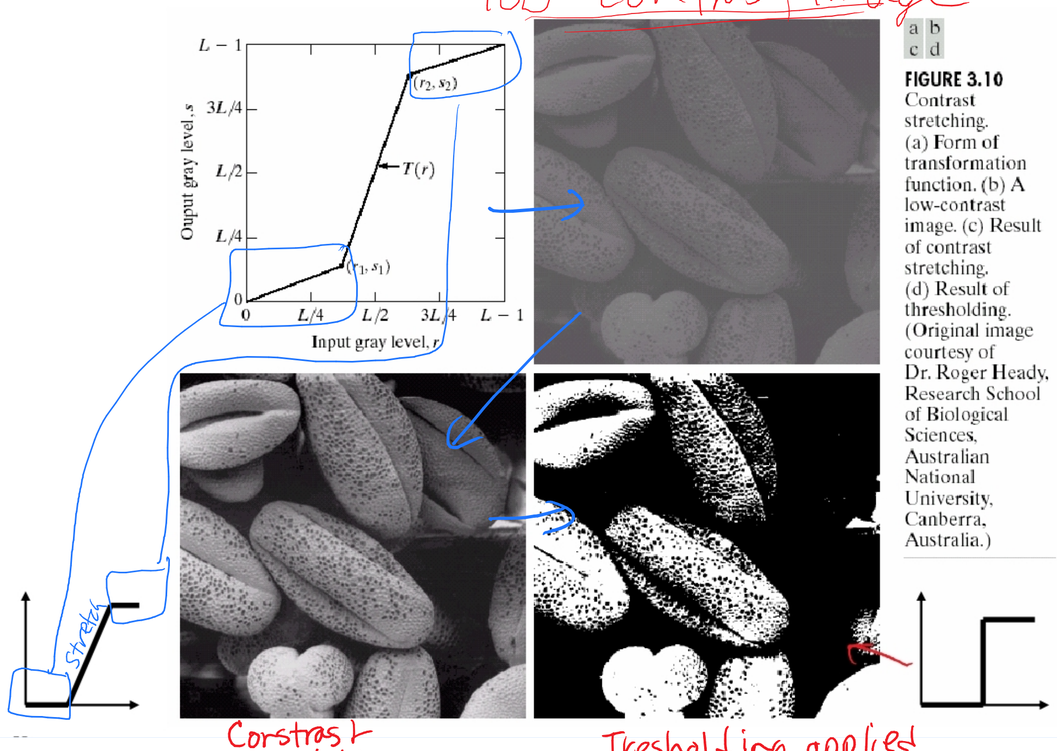


FIGURE 3.10 Contrast stretching. (a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

Contrasting Stretching

Thresholding applied

This is the brightness
You actually move the histogram to the left or right. add or subtract to each pixel

Linear Contrast Stretch:

$$I(x,y) = \frac{g'_{max} - g'_{min}}{g_{max} - g_{min}} \cdot (I(x,y) - g_{min}) + g'_{min}$$

(contrast value)

$g'_{max} = 255$ = white
$g'_{min} = 0$ = black

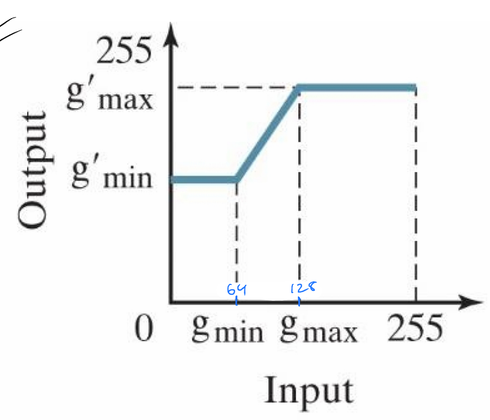
* g_{min} to g_{max} is input image's range

$I(x,y)$: input image
$-g_{min}$: take out g_{min}

This will basically map the Image's low values to BLACK = g'_{min}

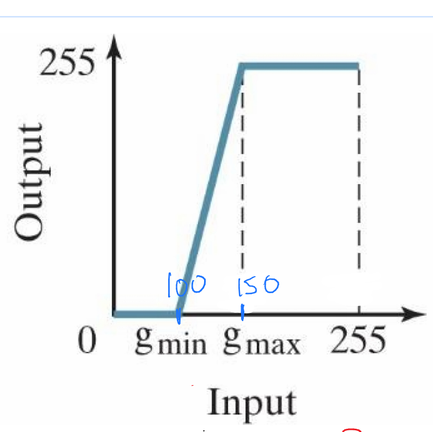
```
output_pixel = middle_of_range + (in[i] - middle_of_range)*contrast + brightness
               (128)                 (128)
```

ex1



- mapping 0 – 64 (g_{min}) to g'_{min}
- mapping 64 (g_{min}) – 128 (g_{max}) to STRETCH FUNC
- mapping 128 (g_{max}) – 255 to $g'_{max} = 255$

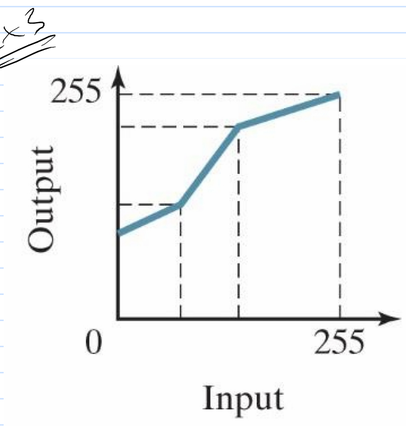
ex2



- mapping 0 – 100 (g_{min}) to $g'_{min} = 0$
- mapping 100 (g_{min}) – 150 (g_{max}) to STRETCH FUNC
- mapping 150 (g_{max}) – 255 to $g'_{max} = 255$

arbitrary

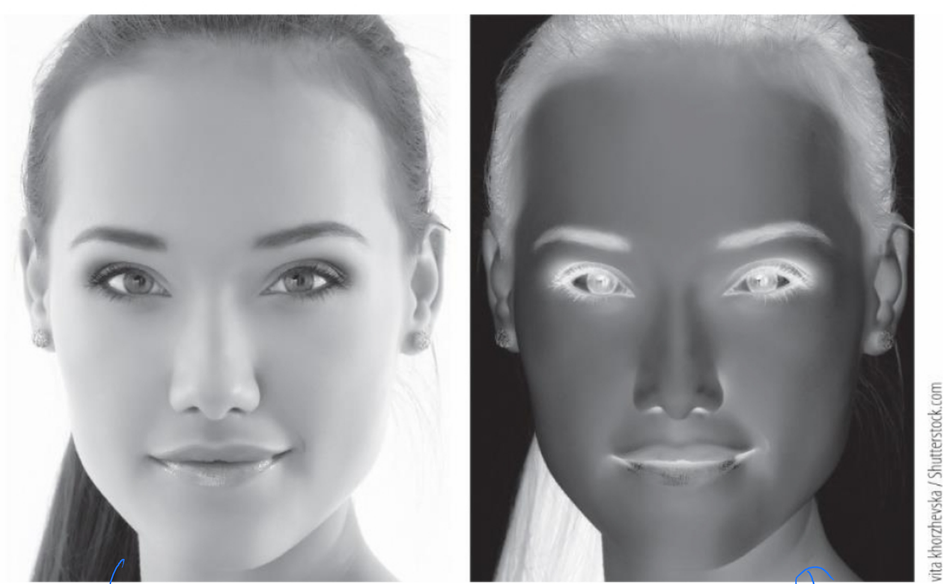
ex3



$$\frac{255 - 0}{150 - 100} = 5.1$$

Figure 3.9 An 8-bit grayscale image (left), and the inverted image obtained by subtracting each pixel from 255 (right).

8 bit grayscale pic



each output pixel = 255 − in[i]

Contrast Code

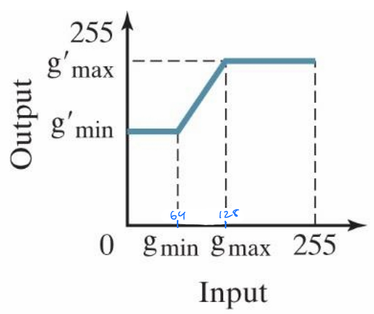
DONE

Linear Contrast Stretch:

$$I(x,y) = \frac{g'_{max} - g'_{min}}{g_{max} - g_{min}} \cdot \left(I(x,y) - g_{min} \right) + g'_{min}$$

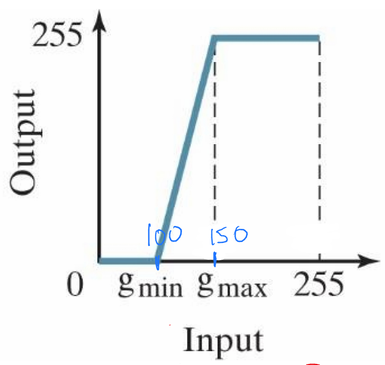
- contrast
- brightness
- $g'_{max} = 255 = \text{white}$
- $g'_{min} = 0 = \text{black}$
- $I(x,y)$ = input image
- g_{min} = take out g_{min}

This will basically map the Image's low values to BLACK $= g'_{min}$



- mapping $0 - \frac{64}{g_{min}}$ to g'_{min}
- mapping $\frac{64}{g_{min}} - \frac{128}{g_{max}}$ to STRETCH FUNC
- mapping $\frac{128}{g_{max}} - 255$ to $g'_{max} = 255$

ex2



- mapping $0 - \frac{100}{g_{min}}$ to $g'_{min} = 0$
- mapping $\frac{100}{g_{min}} - \frac{150}{g_{max}}$ to STRETCH FUNC
- mapping $\frac{150}{g_{max}} - 255$ to $g'_{max} = 255$

$\frac{255 - 0}{150 - 100} = 5.1$

Stretching the lowest and highest grayscale range to 0 to 255

$g'_{max} = 255$
$g'_{min} = 0$

$$\frac{255 - 0}{g_{max} - g_{min}} \cdot \left(in[i] - g_{min} \right) + g'_{min}$$

- say image only got grayscales from 100 to 150

$in[i] = 100 \rightarrow$ (should be 0) $\leftarrow \frac{255}{50}(100 - 100) + 0$ (brightness = 0)

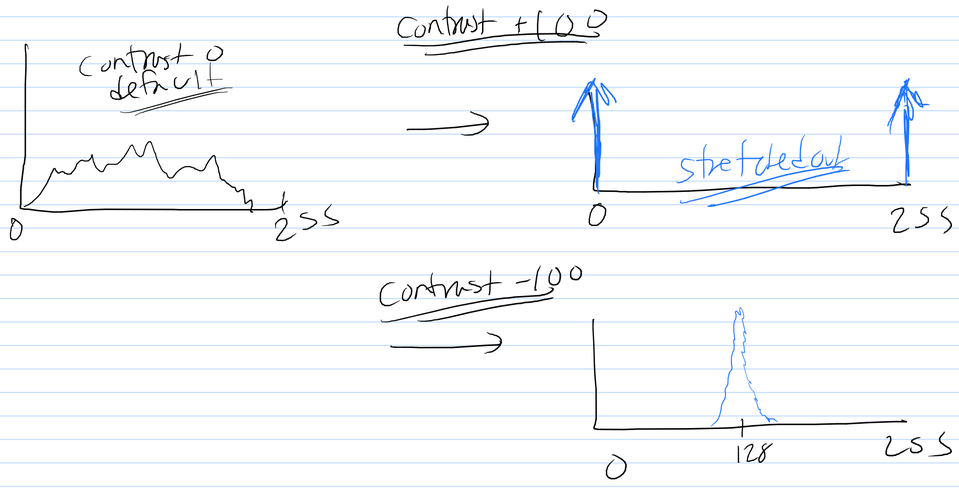
$in[i] = 130 \rightarrow \frac{255}{50}(130 - 100) + 0$

$in[i] = 150 \rightarrow$ (should be 255) $\leftarrow \frac{255}{50}(150 - 100) + 0$

$in[i] = 0 \rightarrow (0) \leftarrow \frac{255}{50}(0 - 100) + 0$

1) find smallest intensity in pic

1) $\left(\text{Contrast} \left(in[i] - g_{min} \right) + \text{brightness} \right) = \text{val}$ (0 to 255 output)
2) if val < 0 → set 0
3) if val > 0 → set 255



answer

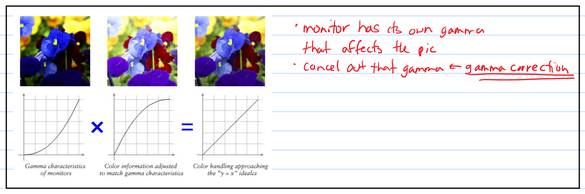
output_pixel = middle_of_range + (in[i] − middle_of_range)* contrast + brightness

(128 = middle_of_range)

← answer

Gamma Correction

- Controls brightness of image.
- If you want to accurately displayed image → need to do contrast correction.
- not properly corrected → image can look bleached out or too dark



- monitor has its own gamma that affects the pic
- cancel out that gamma ← gamma correction

$S = cr^{\gamma}$

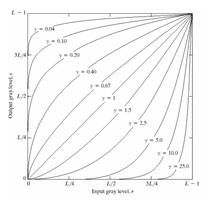
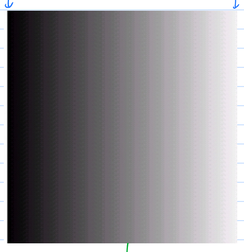


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

- Cand γ are (+) constants
- $c = \gamma = 1$ → identity function (input image = output image)
- This power-law curve
  - maps narrow range of dark vals to wider range
  - vice versa

Monitor Gamma Correction

input scale

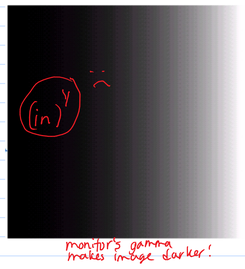


monitor adds its gamma

$(in)^{\gamma}$

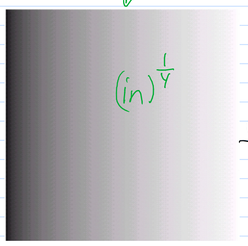
in → Monitor → $(in)^{\gamma}$ best

Image on monitor



monitor's gamma makes image darker!

Gamma correction: Before putting thru monitor change input image to cancel out monitor gamma



$(in)^{\frac{1}{\gamma}}$ → monitor → $(in)^{\gamma}$ → in good

Spreading out dark vals by $\gamma < 1$

$S = cr^{\gamma}$

$S = (1)r^{\gamma}$

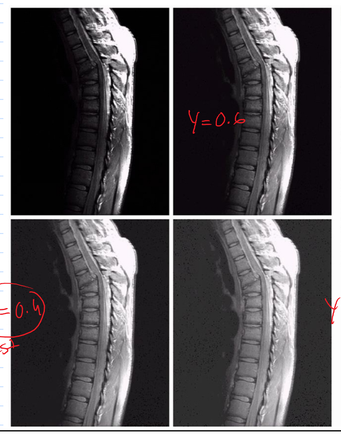


FIGURE 3.8 (a) Magnetic resonance (MR) image of a fractured human spine. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 0.6, 0.4$, and 0.3, respectively. (Original image for this example courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

Fixing washed out image $\gamma > 1$

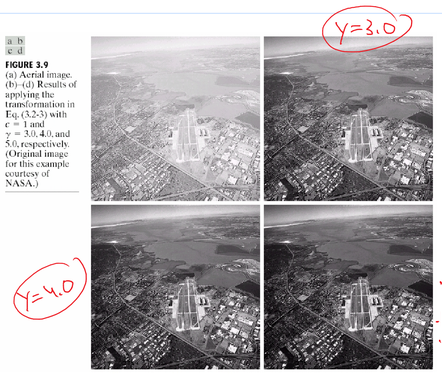


FIGURE 3.9 (a) Aerial image. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0, 4.0$, and 5.0, respectively. (Original image for this example courtesy of NASA.)

Washed-out image. Shrink graylevel range ⇨ $\gamma > 1$

(b) $\gamma = 3.0$ (suitable)

(c) $\gamma = 4.0$ (suitable)

(d) $\gamma = 5.0$ (High contrast; the image has areas that are too dark; some detail is lost)

Code

$$c \cdot \left(\frac{in[i,j]}{c}\right)^{1/\text{gamma}}$$

c = 255

float

Histogram

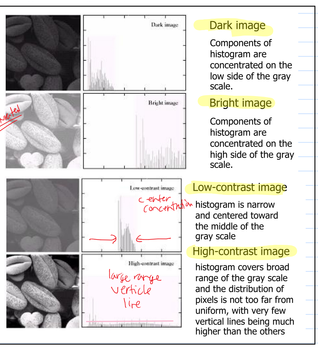
- a histogram with gray lvls from [0, L-1] is

r_k = k^{th} gray lvl
n_k = grayscale freq

$h(r_k) = n_k$

The k^{th} gray level — particular grayscale frequency: The # of pixels in the image with that grayscale

high contrast — dark image + low contrast — good exposure



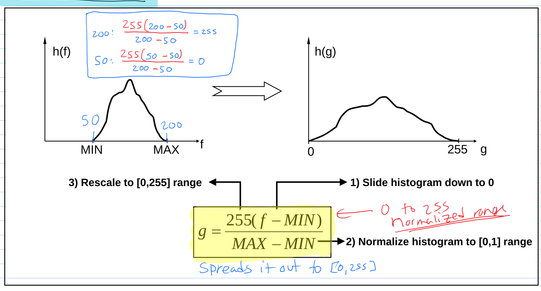
Dark image
Components of histogram are concentrated on the low side of the gray scale.

Bright image
Components of histogram are concentrated on the high side of the gray scale.

Low-contrast image
histogram is narrow and centered toward the middle of the gray scale

High-contrast image
histogram covers broad range of the gray scale and the distribution of pixels is not too far from uniform, with very few vertical lines being much higher than the others

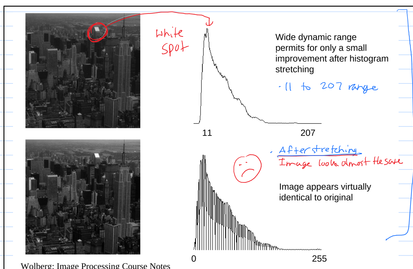
Histogram Stretching



normalization range: MIN to MAX
image range: min_z to max_z

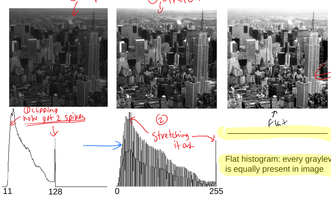
$$g = (in[] - min_z)\left(\frac{MAX - MIN}{max_z - min_z}\right) + MIN$$

normalized range MIN to MAX



Wolberg: Image Processing Course Notes

Improve effectiveness of histogram stretching by clipping intensities first



Histogram Stretching
1) clip the histogram first to get side spikes
if histo is [0 to 140]
[10 to 125] ← 2 side spikes
2) use the func to stretch it

Histogram equalization

1) find the frequency of each grayscale

```
for(i=0; i<MXGRAY; i++)
    H[i] = 0;
for(i=0; i<total; i++)
    H[ in[i] ] ++;
```

make a LUT where all is 0 (LUT: 0 … 255)

go thru each pixel vale and increment hist every time it come up (image → LUT)

2) Divide each grayscale entry @ gray level (r_k) by the total # of pixels in that image (n)

What is the probability that grayscale r_k will be @ the pixel?

$p(r_k) = \frac{n_k}{n}$

- r_k = gray level
- n_k = frequency of that gray lvl
- n = total # of pixels in image

normalized: Sum = 1

```
for(i=0; i<MXGRAY; i++)
    p[i] = LUT[i] / n
```

p[i]: probability or occurrence table; LUT[i]: LUT that has the frequency; n: total # of pixels

Didn't do histo matching