جَامعة المَنارة
MANARA UNIVERSITY

Digital Image Processing

المحاضرة الخامسة

Image Enhancement

العمليات على مستوى البكسلات
العمليات على السويات الرمادية (تقليل، تعتيب، تقطيع، توابع التحويل النقطية التحويل)

د. عيسى الغنام

2025

العمليات على مستوى البكسلات

- **تعديل قيم البكسل(قيم الشدة اللونية للبكسل) دون إحداث تغيير في حجمه أو موقعه أو البنى المحلية في الصورة**

– تحسين الصورة باستخدام العمليات الحسابية و المنطقية Enhancement Using Arithmetic/Logic Operations: العمليات الحسابية والمنطقية هي عمليات تجري على العناصر المتقابلة في صورتين عنصراً لعنصرٍ. باستثناء عملية NOT

العمليات الحسابية (/ , * ,-,+)

العمليات المنطقية (NOT, AND, OR, XOR)

–العمليات على السويات الرمادية (تقليل، تعتيب، تقطيع، توابع التحويل النقطية التحويل)

Definition: is the conversion of the intensity of the original image into the intensity of the resulting image using the function:

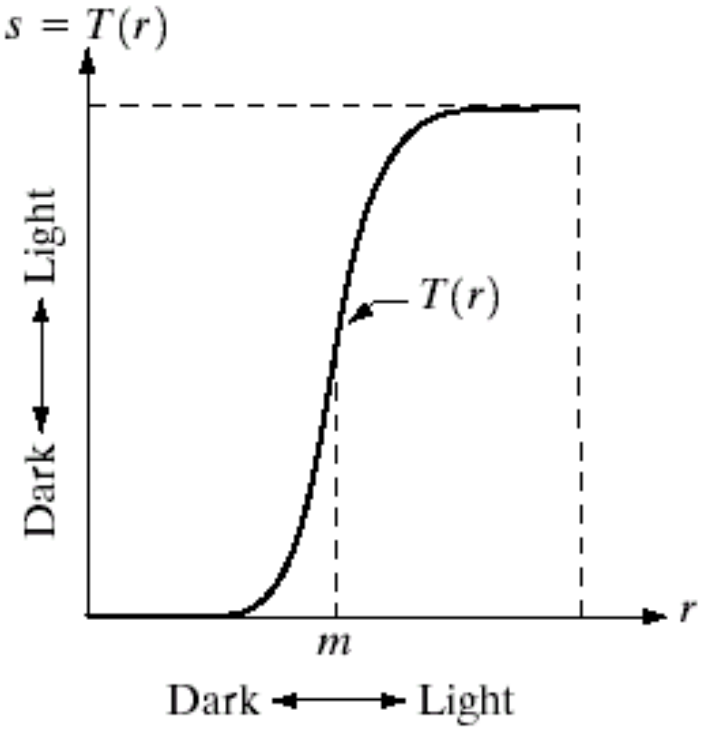
$$g(x,y)=T(f(x,y))$$

f هو الصورة الدخل
g الصورة المعالجة صورة الخرج
T هو تابع المعبر عن العلمية المطبقة

$$s = T(r)$$

Where r is the input intensity and s is the output intensity

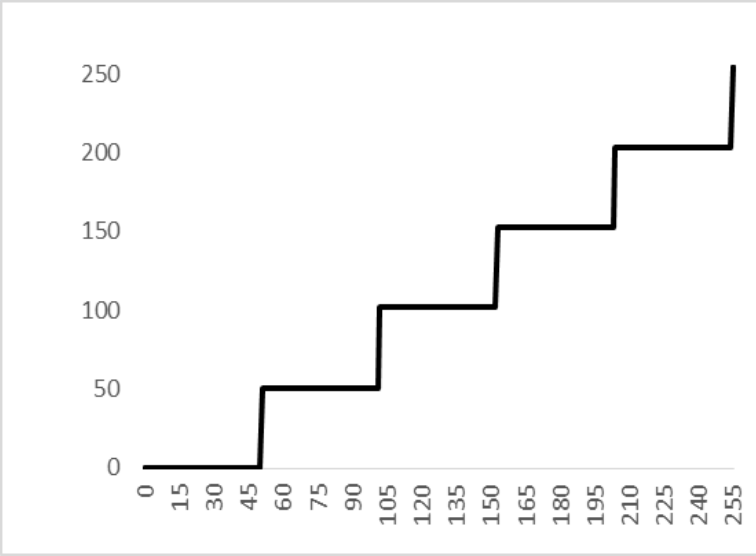
r سوية اللونية لصورة الدخل
S السوية اللونية في الخرج



Example: Contrast enhancement

العمليات على السويات الرمادية
التقليل من السويات الرمادية

- تتطلب بعض عمليات التصنيف والتعرف أن تكون الصورة ممثلة بعدد محدود من السويات اللونية
- يتم تقسيم مجال السويات اللونية إلى مجموعة من المجالات المتساوية M
- يكون عدد السويات اللونية التي سيتم جمعها في سوية لونية واحدة مساوٍ لـ

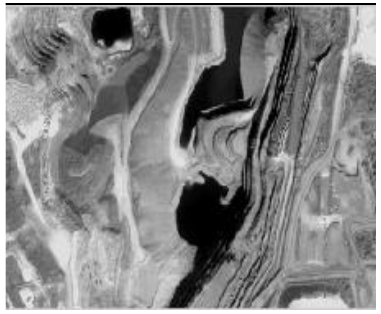


تقسيم السويات الرمادية إلى مجالات متساوية عددها
M=5

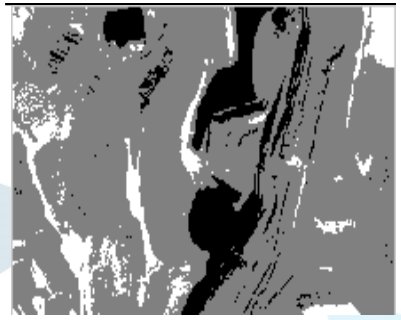
$$N = \frac{L-1}{M}$$

N=(256-1)/5=51

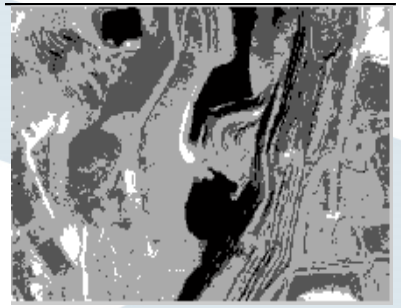
تأثير تقليل عدد السويات الرمادية في الصورة



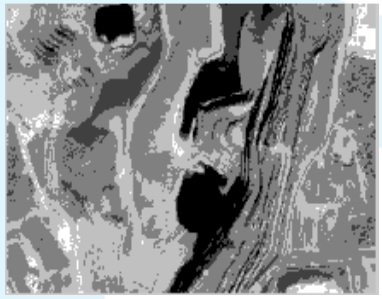
الصورة الأصلية



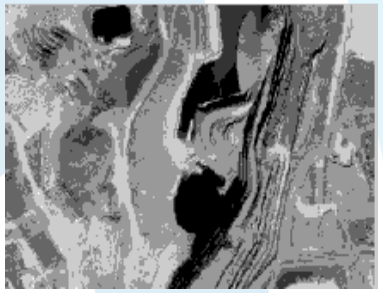
M = 3



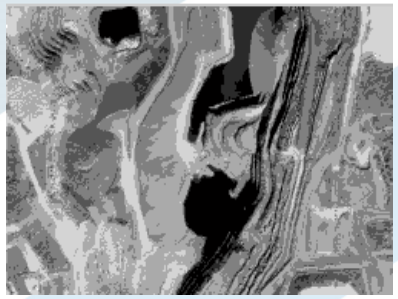
M = 4



M = 5



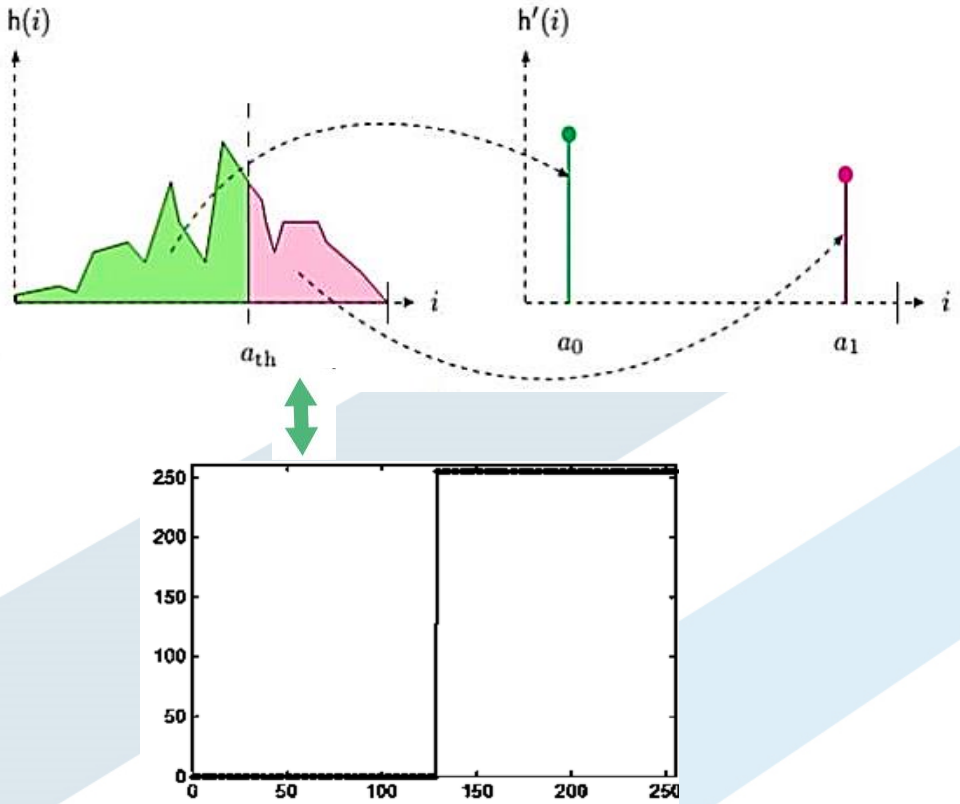
M = 6



M = 7

العمليات على السويات الرمادية
التعتيب

حالة خاصة من تقليل السويات الرمادية (M=2 وليس بالضرورة بعدد سويات متساوفي كل مجال)



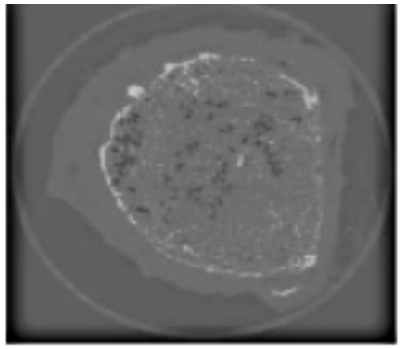
تستخدم لعزل الكائنات المرغوبة في الصورة اعتماداً على لونها

تستخدم عتبة $0 < a_{th} \leq amax$

$$B = \begin{cases} a_0 & a < a_{th} \\ a_1 & a \geq a_{th} \end{cases}$$

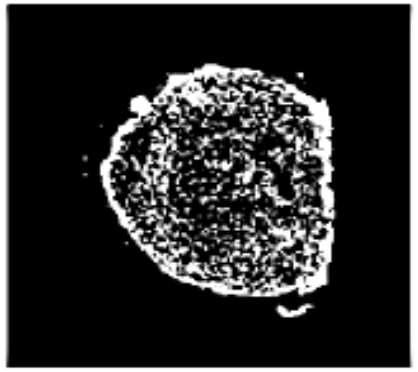
نحصل على صورة ثنائية عند $a_0=0, a_1=1$

تأثير تغيير قيمة العتبة في الصور الثنائية الناتجة

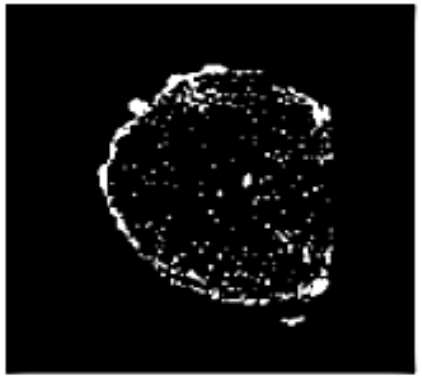


الصورة الأصلية

الصورة الثنائية عند عتبة ath=100



الصورة الثنائية عند عتبة ath=115



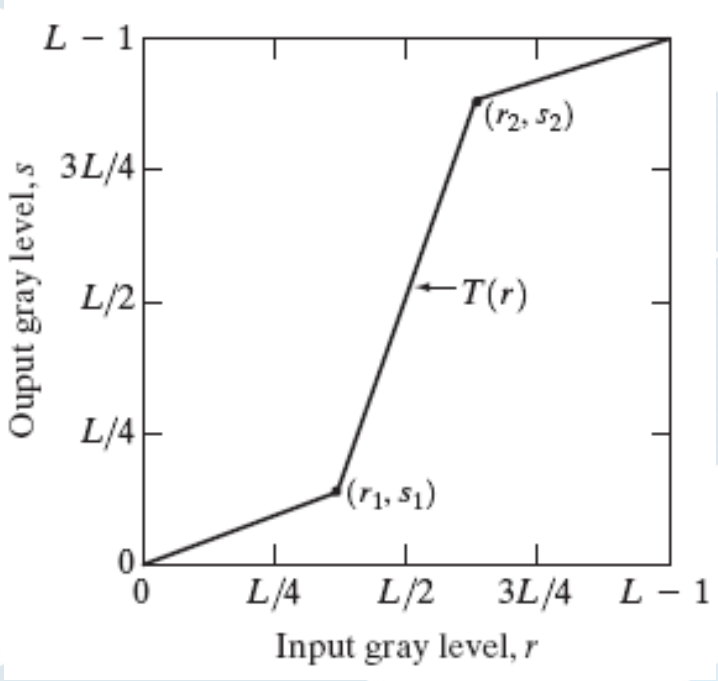
الصورة الثنائية عند عتبة ath=128

توابع التحويل الخطي المجزأ

piecewise – linear transformation functions

- التباين ذو المجال الموسع.contrast stretching
- تجزئة السويات الرمادية Gray-level slicing.
- تجزئة مستويات الخانة Bit-plane slicing

التباين ذو المجال الموسع
contrast stretching



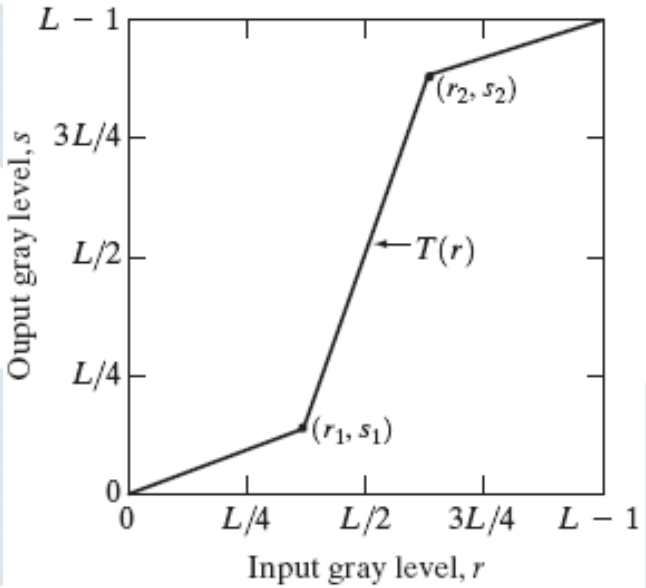
- و هو من أبسط توابع التحويل الخطي المجزأ التابع المستخدم في تحويل التباين ذو المجال الموسع يكون بالشكل:
- تنتج الإضاءة الضعيفة صوراً ضعيفة التباين من الممكن حل المشكلة باستخدام contrast stretching
- الهدف من هذه الطريقة زيادة المدى الديناميكي لقيم السويات الرمادية في الصورة المعالجة.

التباين ذو المجال الموسع
contrast stretching

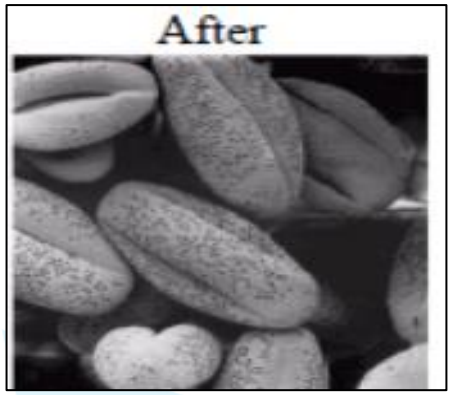
Contrast means the difference between
The darkest color in Picture with the
brightest color in the image

المثال التالي يوضح تأثير تحويل التباين ذو المجال الموسع.

الصورة الأولى تعبر عن صورة ضعيفة التباين.



To increase the dynamic range of the gray levels in the image being processed.



الثانية تعبر عن ناتج التحويل باستخدام القيم التالية

$$(r_1, s_1) = (r_{\min}, 0)$$

$$(r_2, s_2) = (r_{\max}, L-1)$$

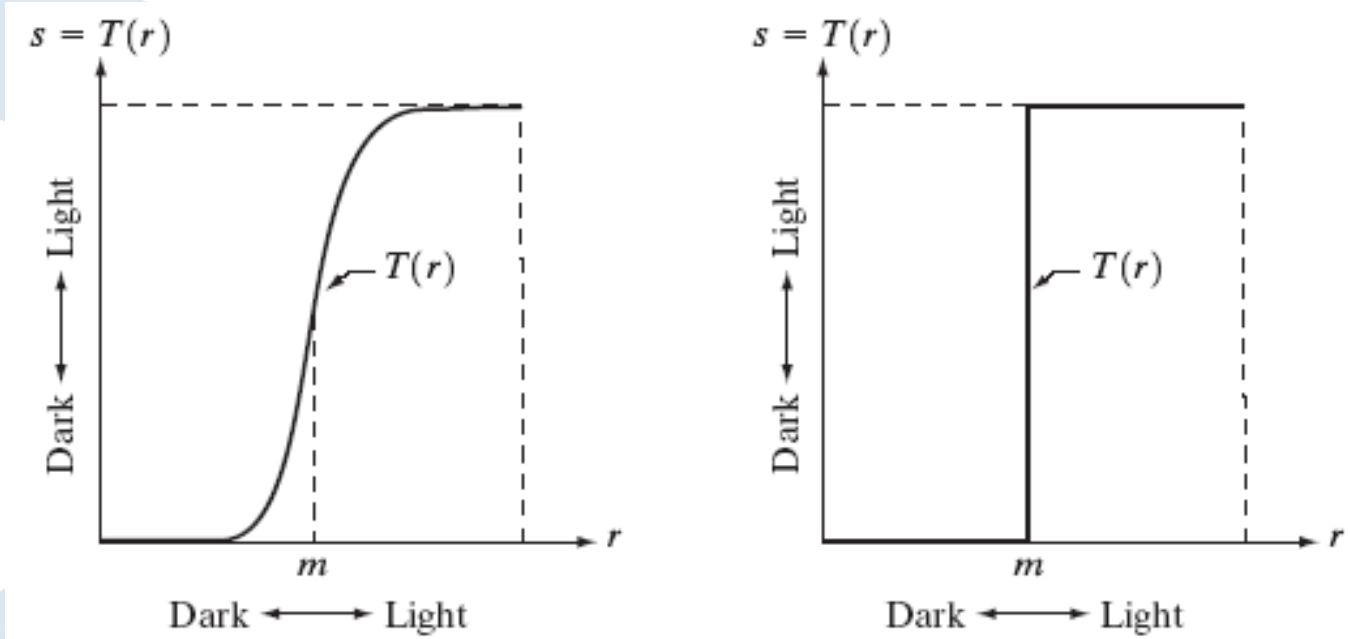
حيث قيم r_{max} و r_{min} تحدد أعلى و أدنى قيمة سوية رمادية في الصورة

contrast stretching

الثالثة تعبر عن ناتج التحويل على الصورة السابق باستخدام القيم التالية r1=r2=m حيث m هي المتوسط لقيم السويات الرمادية في الصورة. الأصلية

thresholding

التباين ذو المجال الموسع
contrast stretching



a b

FIGURE 3.2 Gray-level transformation functions for contrast enhancement.

Contrast Stretching

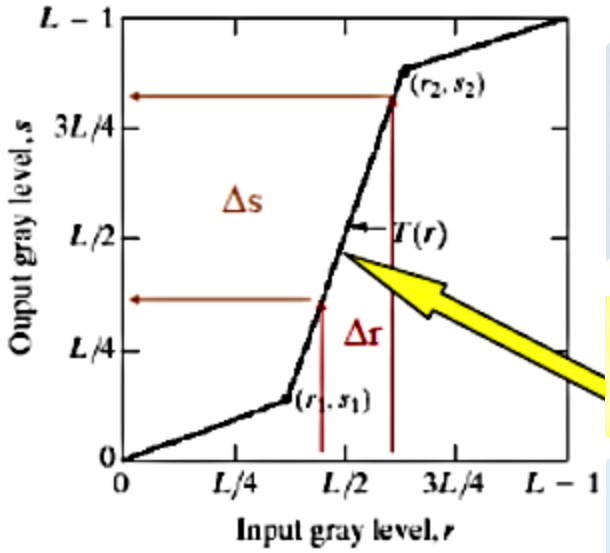
THRESHOLDING

التباين ذو المجال الموسع
contrast stretching

How to know where the contrast is enhanced ?

Look at T (r) Slope.

- if Slope > 1 → Contrast increase
- if Slope < 1 → Contrast decrease
- if Slope = 1 → Contrast same



Δr is narrow compared to Δs and leads to Contrast increase

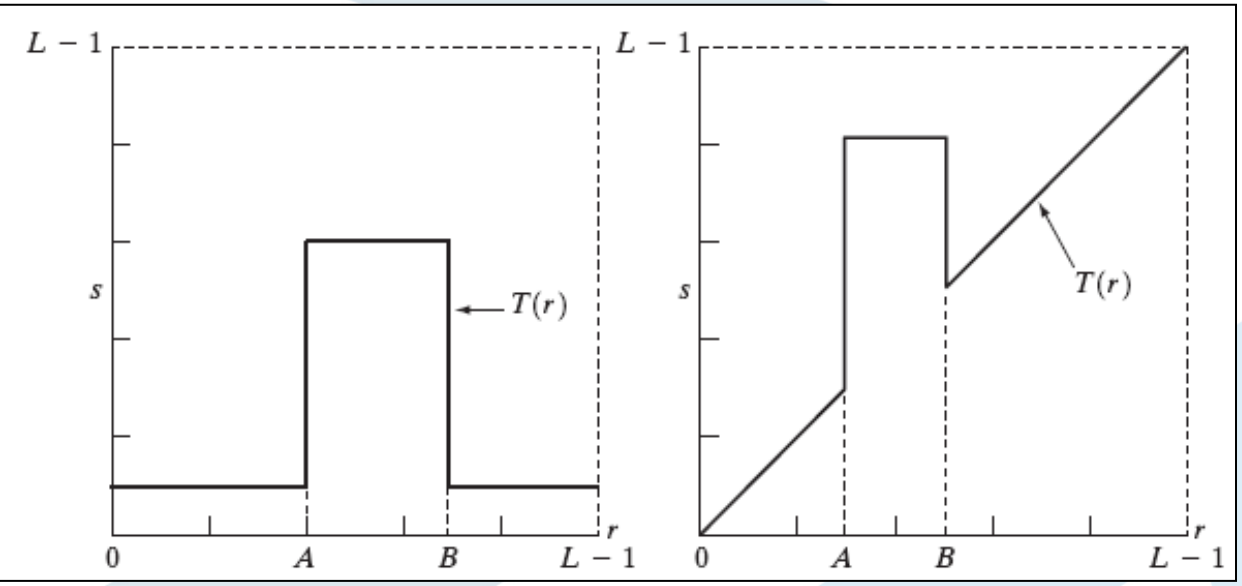
ناقش الحالات:

r1=s1 و r2=s2

s1=0 و r1=r2 و s2=L-1

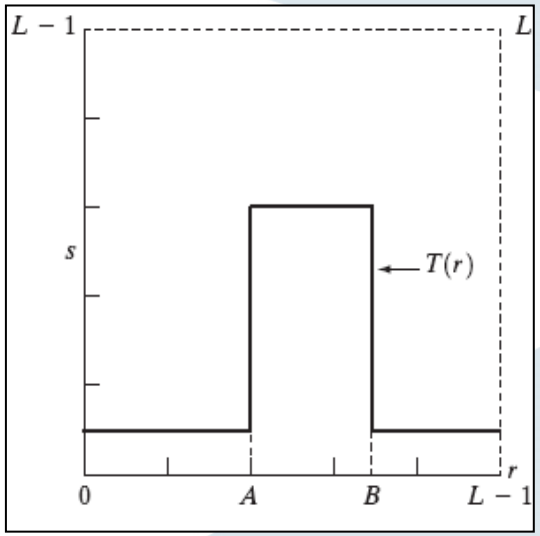
- The locations of (r_1,s_1) and (r_2,s_2) control the shape of the transformation function.
  - If $r_1 = s_1$ and $r_2 = s_2$ the transformation is a linear function and produces no changes.
  - If $r_1=r_2$, $s_1=0$ and $s_2=L-1$, the transformation becomes a thresholding function that creates a binary image.
- More on function shapes:
  - Intermediate values of (r_1,s_1) and (r_2,s_2) produce various degrees of spread in the gray levels of the output image, thus affecting its contrast.
  - Generally, $r_1 \leq r_2$ and $s_1 \leq s_2$ is assumed.

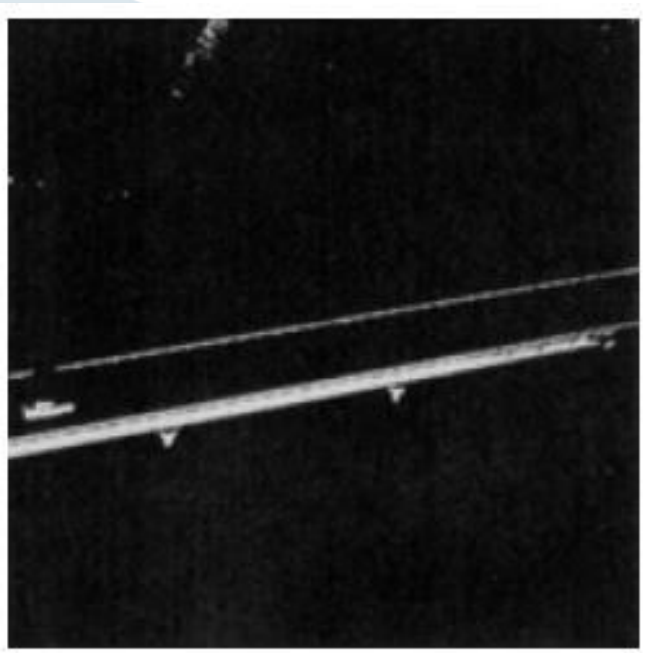
تجزئة السويات الرمادية
Gray-level slicing



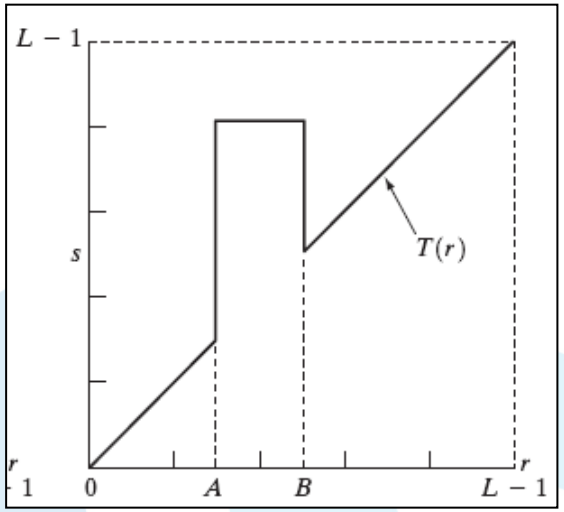
- تعتمد على توضيح مجال معين
- يستخدم هذا التحويل في العديد من التطبيقات كاستخدامه في صور الأقمار الصناعية للكشف عن المياه الجوفية.
- هناك العديد من الطرق المتبعة لانجاز هذا التحويل منها:
  - إظهار قيم كبيرة لجميع مستويات الرماديات ضمن المجال المرغوب و مستويات منخفضة للنقاط الأخرى غير الهامة.
  - إضافة سطوع أعلى للمجال المرغوب من مستويات الرماديات ولكنه يبقي الخلفية وسويات الرمادية الأخرى ضمن الصورة كما هي.

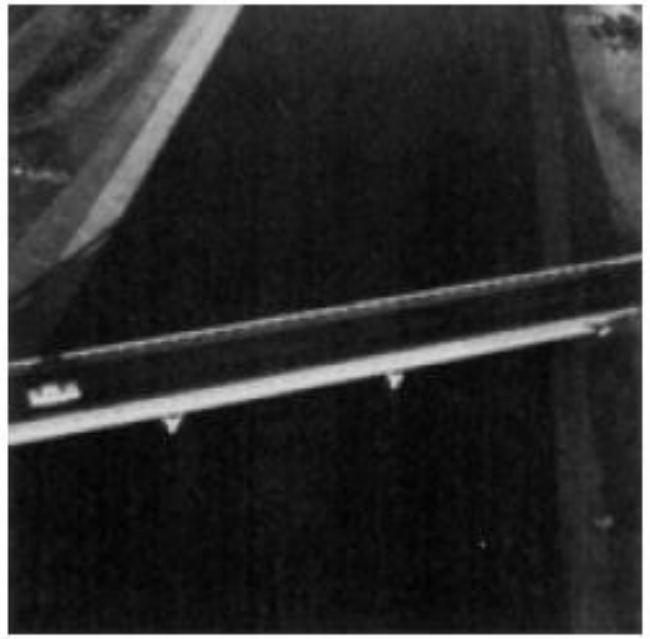
تجزئة السويات الرمادية
Gray-level slicing





السويات أقل من a و أكثر من b أصبحت سوداء اللون



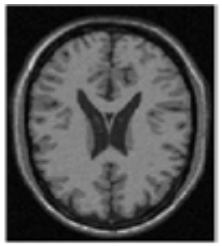


السويات أقل من a و أكثر من b حافظت على قيمها

العمليات على السويات الرمادية
تقطيع السويات الرمادية

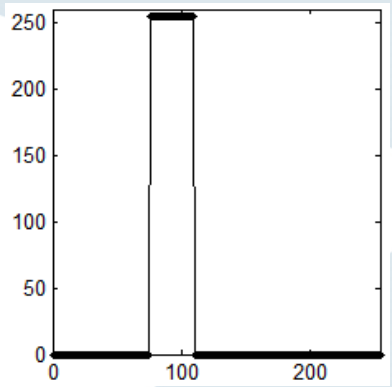
- إظهار مجال معيّن من السويات الرمادية دون غيرها في التطبيقات التي تهدف لتحسين عرض ميّزات معيّنة في الصورة

- هناك طريقتان أساسيتان:
  - إعطاء قيمة لونية عالية لكلّ السويات الرمادية التي تقع ضمن المجال المرغوب في حين تعطى قيمة لونية منخفضة لبقية السويات
  - زيادة سطوع مجال السويات الرمادية المرغوب في حين تظلّ السويات الرمادية الأخرى بلا أي تغيير

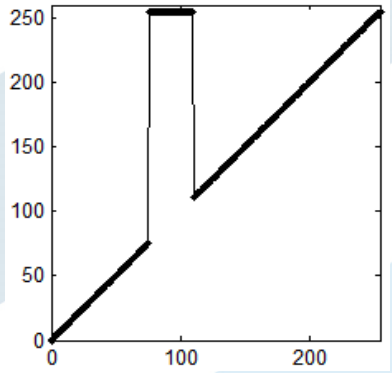


الصورة الأصلية

Approach 1



التقطيع وفق الطريقة الأولى

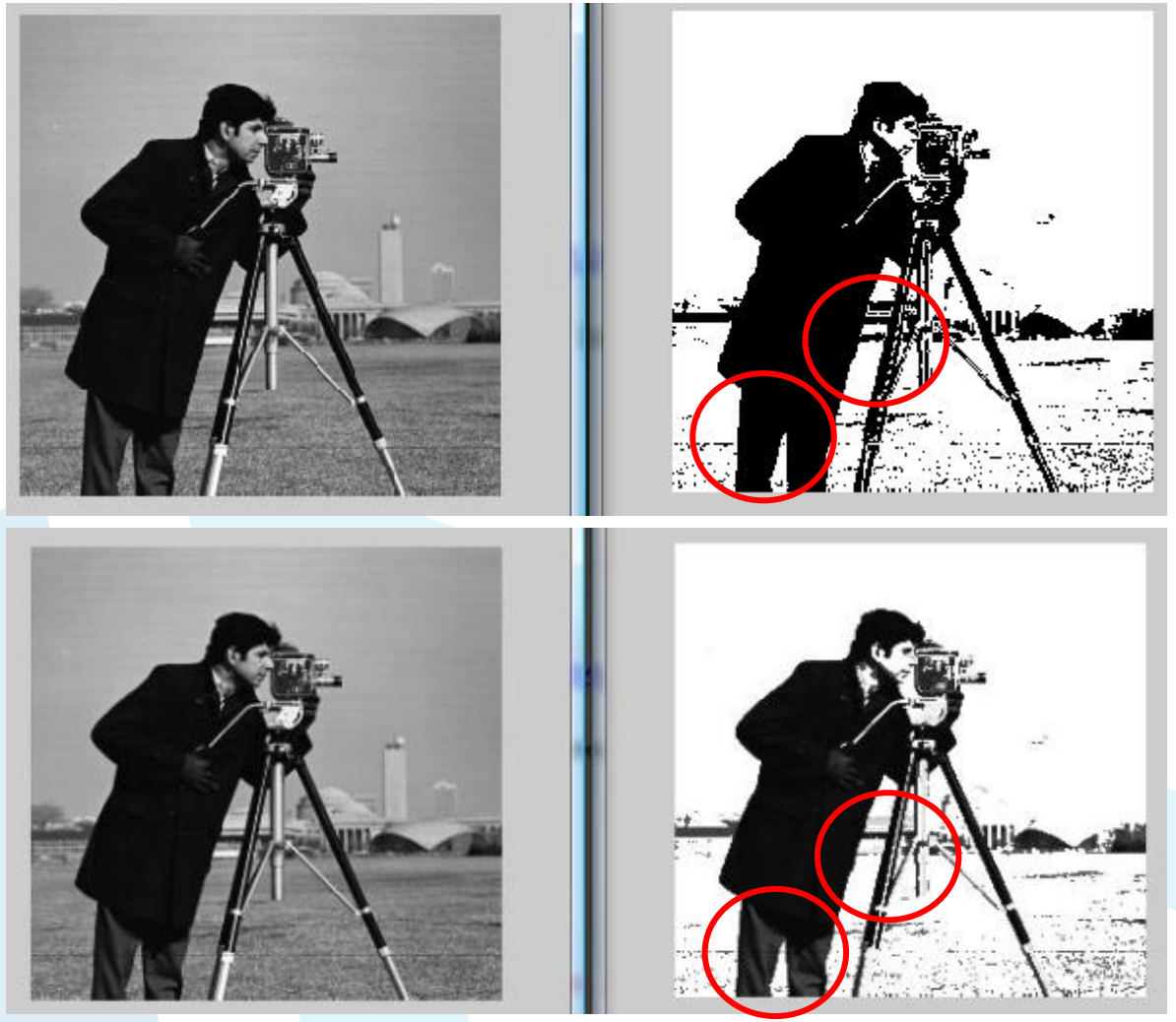


Approach 2

التقطيع وفق الطريقة الثانية

العمليات على السويات الرمادية
تقطيع السويات الرمادية

- example1: apply intensity level slicing in Matlab to read cameraman image, then If the pixel intensity in the old image is between (100 and 200) convert it in the new image into 255(white). Otherwise convert it to 0 (black).
- example2: apply intensity level slicing in Matlab to read cameraman image, then If the pixel intensity in the old image is between (100 and 200) convert it in the new image into 255(white). Otherwise it leaves it the same



Solution1:

```
x=imread('cameraman.tif');
y=x;
[w h]=size(x);
for i=1:w
   for j=1:h
       if x(i,j)>=100 && x(i,j)<=200
            y(i,j)=255;
       else
            y(i,j)=0;
       end
   end
end
figure, imshow(x);
figure, imshow(y);
```

Solution2:

```
x=imread('cameraman.tif');
 y=x;
 [w h]=size(x);
 for i=1:w
     for j=1:h
         if x(i,j)>=100 && x(i,j)<=200
              y(i,j)=255;
         else
              y(i,j)=x(i,j);
         end
     end
end
figure, imshow(x);
figure, imshow(y);
```

Solution1:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Read the image
x = cv2.imread('cameraman.tif', cv2.IMREAD_GRAYSCALE)

# Create a copy of the image
y = np.zeros_like(x)  # Create an empty image with the same shape as x

# Apply the condition to create the binary image
y[(x >= 100) & (x <= 200)] = 255  # Set pixels within range to 255
y[(x < 100) | (x > 200)] = 0      # Set pixels outside the range to 0

# Display the original and processed images
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(x, cmap='gray')
plt.title('Original Image')
plt.axis('off')  # Hide axis
plt.subplot(1, 2, 2)
plt.imshow(y, cmap='gray')
plt.title('Processed Image')
plt.axis('off')  # Hide axis
plt.show()
```

Figure 1

Original Image

Processed Image

Solution2:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Read the image
x = cv2.imread('cameraman.tif', cv2.IMREAD_GRAYSCALE)

# Create a copy of the image
y = x.copy()

# Apply the condition to modify the image
y[(x >= 100) & (x <= 200)] = 255  # Set pixels within range to 255
# Pixels not within the range will retain their original value due to the copy

# Display the original and processed images
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(x, cmap='gray')
plt.title('Original Image')
plt.axis('off')  # Hide axis

plt.subplot(1, 2, 2)
plt.imshow(y, cmap='gray')
plt.title('Processed Image')
plt.axis('off')  # Hide axis

plt.show()
```

Figure 1

Original Image

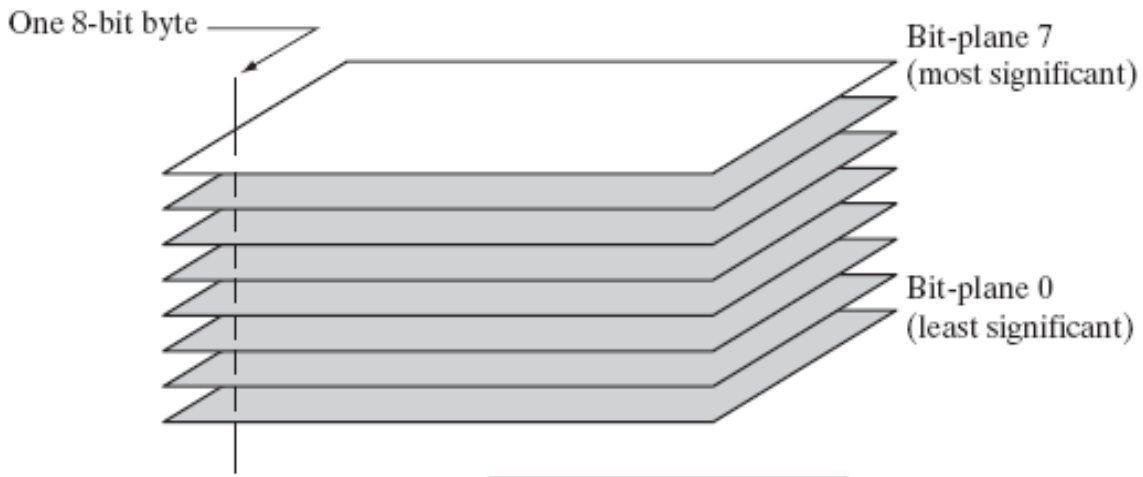
Processed Image

Homework

Example: Apply Intensity Level Slicing (Approch2) in Python or Matlab To Read Moon Image, Then If The Pixel Intensity In The Old Image Is Between (0 And 20) Convert It In The New Image Into 130.

التقطيع حسب سويات البت
Bit- plane slicing

- بدلاً من توضيح مجال معين من السويات الرمادية قد يكون التركيز على العناصر المساهمة في وضوح الصورة عن طريق اعتبار خانات معينة أكثر فعاليةً.
- إنتاج صور تعبّر عن القيم الممثلة لسويات البت المختلفة والتي تدعى بالتقطيع بحسب سويات البت (تقطيع الصورة إلى سويات عدّة تمثل كلّ سوية مستوى بت معيّن في الصورة)



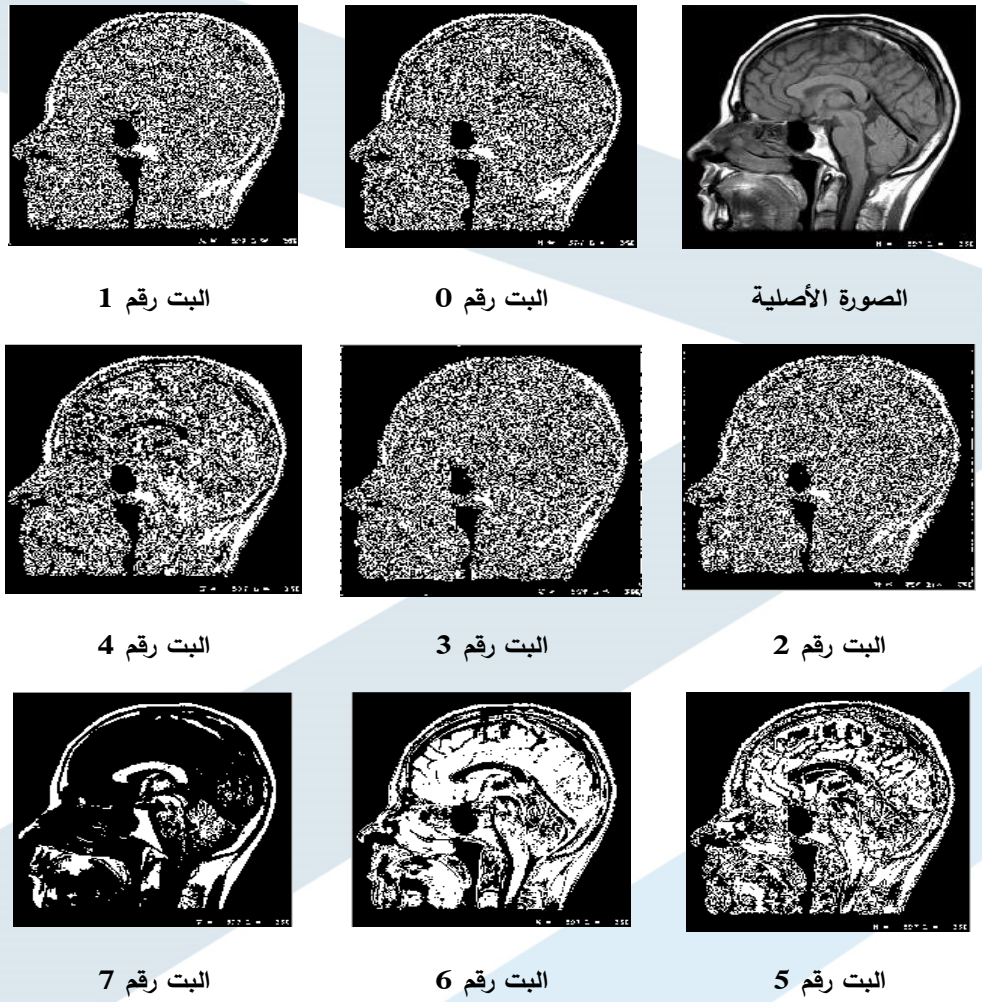
- 8-bit Image composed of 8 1-bit planes.
- Higher-order bits usually contain most of the significant visual information. Most significant bits contain the majority of visually significant data.
- Lower-order bits contain subtle details.

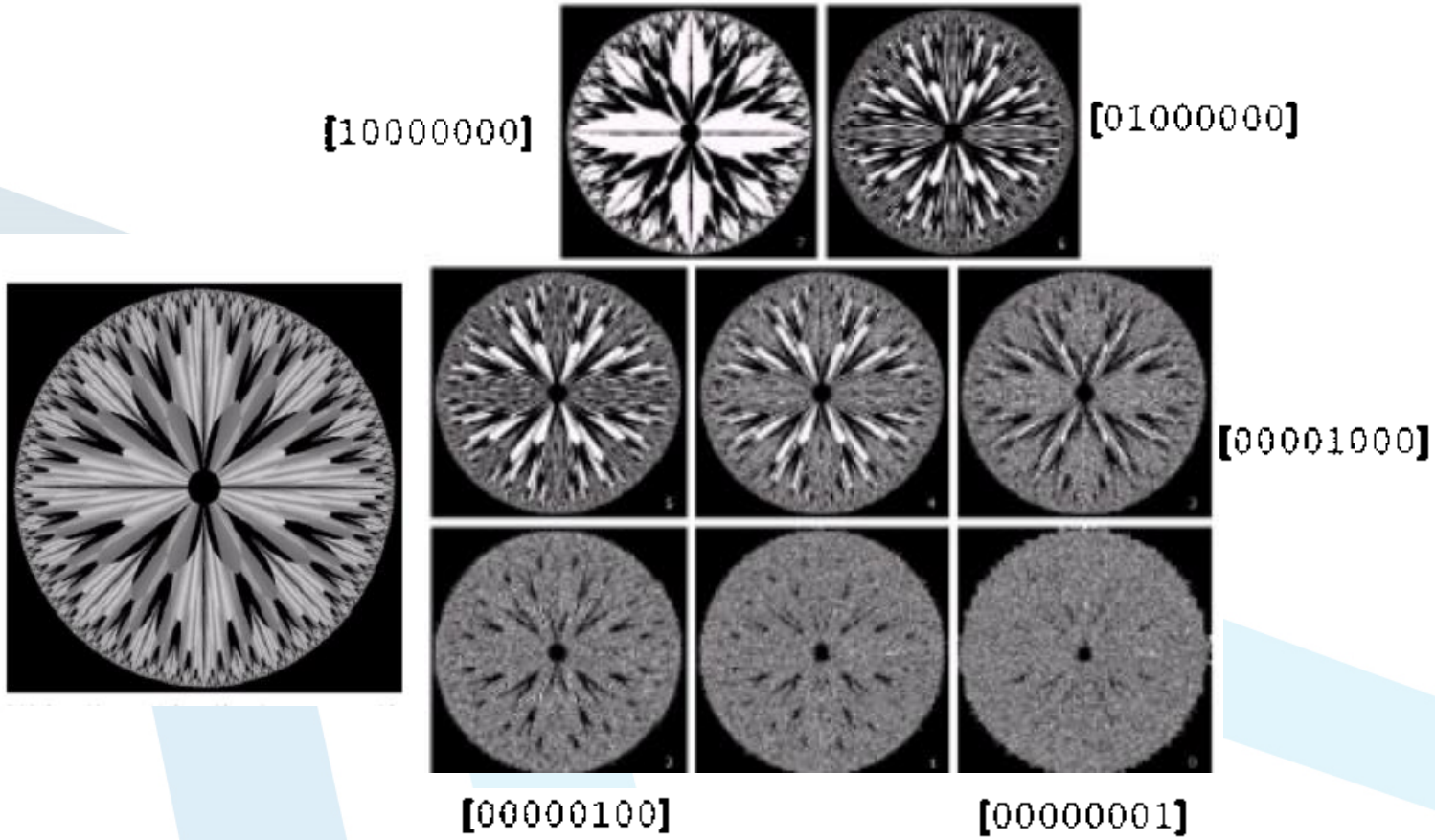
- Remember that pixels are digital numbers composed of bits.
- Pixels are digital numbers, each one composed of bits. Instead of highlighting gray-level range, we could highlight the contribution made by each bit.
- This method is useful and used in image compression.

التقطيع حسب سويات البت

Bit- plane slicing

مثال: الصور الثمانية الناتجة عن التقطيع بحسب سويات البت





التقطيع حسب سويات البت
Bit- plane slicing

a b c
d e f
g h i

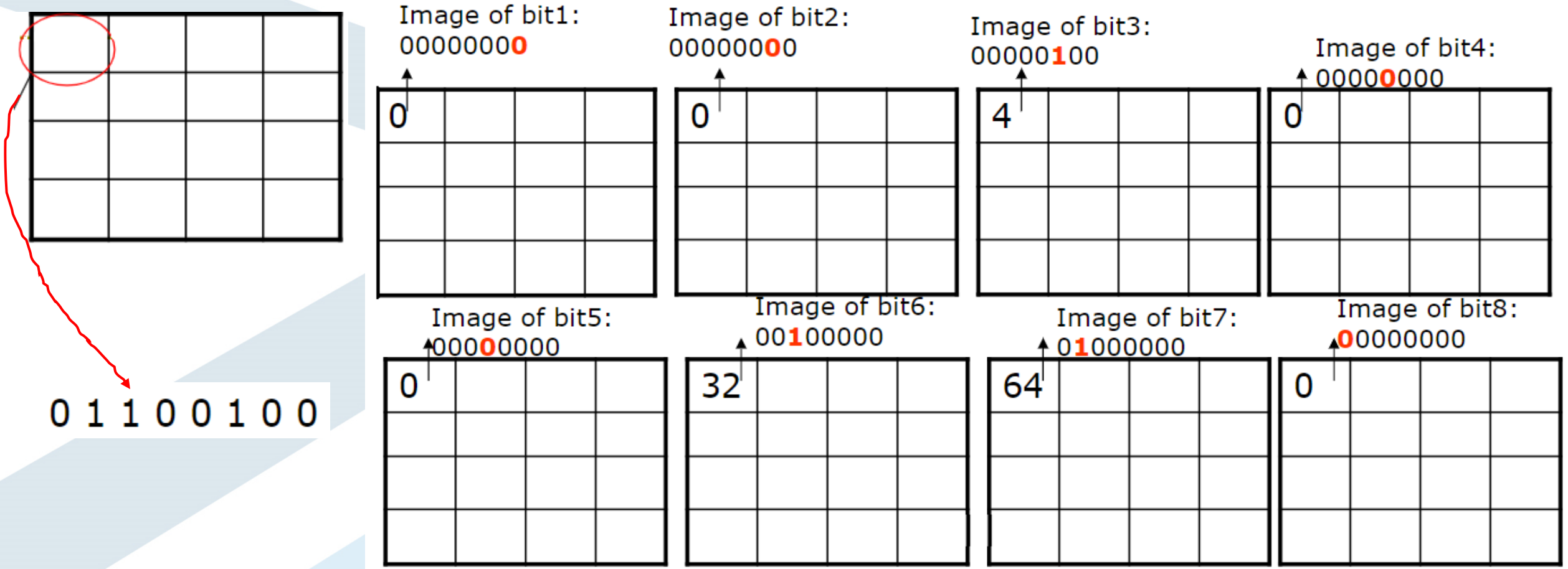
FIGURE 3.14 (a) An 8-bit gray-scale image of size 500 × 1192 pixels. (b) through (i) Bit planes 1 through 8, with bit plane 1 corresponding to the least significant bit. Each bit plane is a binary image.

Reconstructed image using only bit planes 8 and 7

Reconstructed image using only bit planes 8, 7 and 6

Reconstructed image using only bit planes 7, 6 and 5

التقطيع حسب سويات البت
Bit- plane slicing



التقطيع حسب سويات البت
Bit- plane slicing

Apply bit-plane slicing in Matlab to read cameramanimage , then extract the image of bit 6.

Solution:

```
x=imread('cameraman.tif');
y=x*0;
[w h]=size(x);
for i=1:w
   for j=1:h
      b=bitget(x(i,j),6);
      y(i,j)=bitset(y(i,j),6,b);
  end
end
figure, imshow(x);
figure, imshow(y);
```

We have to use bitget and bitset to extract 8 images;

Apply bit-plane slicing in python to read cameramanimage , then extract the image of bit 6.

جَـامعة المَـنارة
MANARA UNIVERSITY

التقطيع حسب سويات البت
Bit- plane slicing

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
# Read the image
x = cv2.imread('cameraman.tif', cv2.IMREAD_GRAYSCALE)

# Create an image y initialized to zero with the same shape as x
y = np.zeros_like(x)
# Get the dimensions of the image
w, h = x.shape

# Iterate over each pixel
for i in range(w):
    for j in range(h):
        # Get the 6th bit (note that bit positions start at 0)
        b = (x[i, j] >> 5) & 1  # right shift by 5 and mask with 1
        # Set the 6th bit of y
        y[i, j] = (y[i, j] & ~(1 << 5)) | (b << 5)  # clear the 6th bit and then set it
```

```
# Display the original and processed images
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(x, cmap='gray')
plt.title('Original Image')
plt.axis('off')  # Hide axis

plt.subplot(1, 2, 2)
plt.imshow(y, cmap='gray')
plt.title('Processed Image')
plt.axis('off')  # Hide axis

plt.show()
```


Figure 1
Original Image
Processed Image

Iterate over each pixel

- A nested loop iterates through each pixel in the image.
- The sixth bit is obtained by right-shifting the pixel value by 5 positions and applying a bitwise AND with 1.
- To set this bit in y, we clear the sixth bit in y (using bitwise AND with a mask that has all bits set to 1 except the sixth bit) and then set the sixth bit based on the value of b.

التقطيع حسب سويات البت
Bit- plane slicing

- D156=B10011100

1. 156%2=0
2. FLOOR(156/2) %2= 78%2=0
3. FLOOR(156/4) %2= 39%4 =1
4. FLOOR(156/8) %2=19%2=1
5. FLOOR(156/16) %2=9%2=1
6. FLOOR(156/32) %2=4%2=0
7. FLOOR(156/64) %2=2%2=0
8. FLOOR(156/128) %2=1%2=1

التقطيع حسب سويات البت
Bit- plane slicing

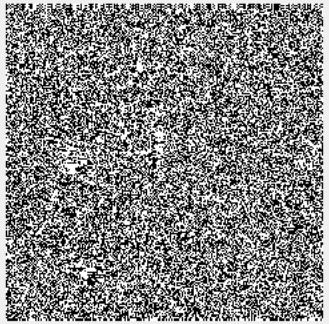
```
c = imread('cameraman.tif');

cd = double(c);

c1 = mod(cd, 2);
c2 = mod(floor(cd/2), 2);
c3 = mod(floor(cd/4), 2);
c4 = mod(floor(cd/8), 2);
c5 = mod(floor(cd/16), 2);
c6 = mod(floor(cd/32), 2);
c7 = mod(floor(cd/64), 2);
c8 = mod(floor(cd/128), 2);

cc = (2 * (2 * (2 * (2 * (2 *
(2 * (2 * c8 + c7) + c6) + c5)
+ c4) + c3) + c2) + c1);
```

```
subplot(2, 5, 1);
imshow(c);
title('Original Image');

subplot(2, 5, 2);
imshow(c1);
title('Bit Plane 1');
subplot(2, 5, 3);
imshow(c2);
title('Bit Plane 2');
subplot(2, 5, 4);
imshow(c3);
title('Bit Plane 3');
subplot(2, 5, 5);
imshow(c4);
title('Bit Plane 4');
subplot(2, 5, 6);
imshow(c5);
title('Bit Plane 5');
```

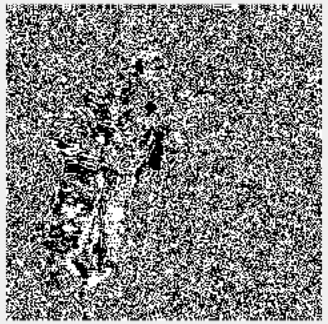
```
subplot(2, 5, 7);
imshow(c6);
title('Bit Plane 6');
subplot(2, 5, 8);
imshow(c7);
title('Bit Plane 7');
subplot(2, 5, 9);
imshow(c8);
title('Bit Plane 8');

subplot(2, 5, 10);
imshow(uint8(cc));
title('Recombined Image');
```

Original Image

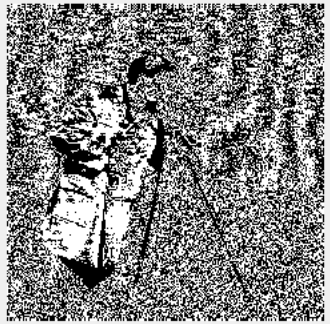
Bit Plane 1



Bit Plane 2



Bit Plane 3



Bit Plane 4

Bit Plane 5

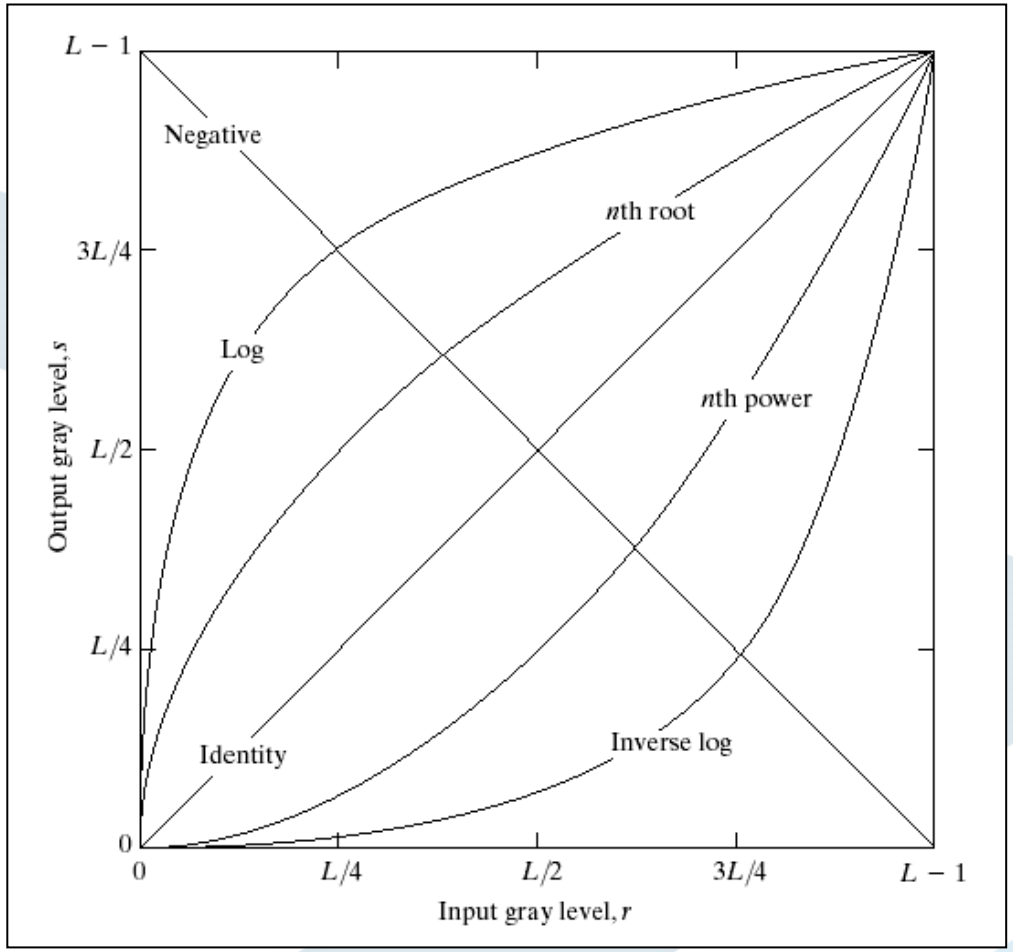
Bit Plane 6

Bit Plane 7

Bit Plane 8

Recombined Image

العمليات على السويات الرمادية
توابع التحويل النقطية



Some basic gray-level transformation functions used for image enhancement

Linear: Negative, Identity

Logarithmic: Log, Inverse Log

Power-Law: nth power, nth root

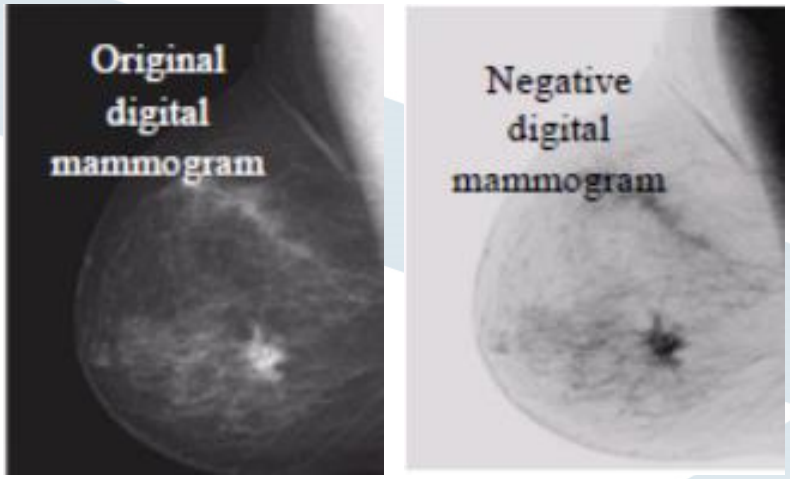
- **تغير توابع التحويل المجال الديناميكي للصورة ليصبح أكثر فاعلية فيتحسن التباين فيها**
- التباين هو الفرق بين أصغر وأكبر قيمة بكسل في الصورة
- **تغير هذه التوابع العلاقة بين المجال الديناميكي ومجال السويات الرمادية الممكنة لتمثيل القيم اللونية في الصورة**
  - التابع اللوغاريتمي Logarithm Transform
  - التابع الأسي Power-Law Transform
  - تابع القوة "جاما"
  - تصحيح غاما

توابع التحويل النقطية
Image negative
معكوس الصورة NOT

- و هي صورة ذات قيم مستويات رمادية محصورة ضمن المجال
- و تستخدم بشكل رئيسي العلاقة:

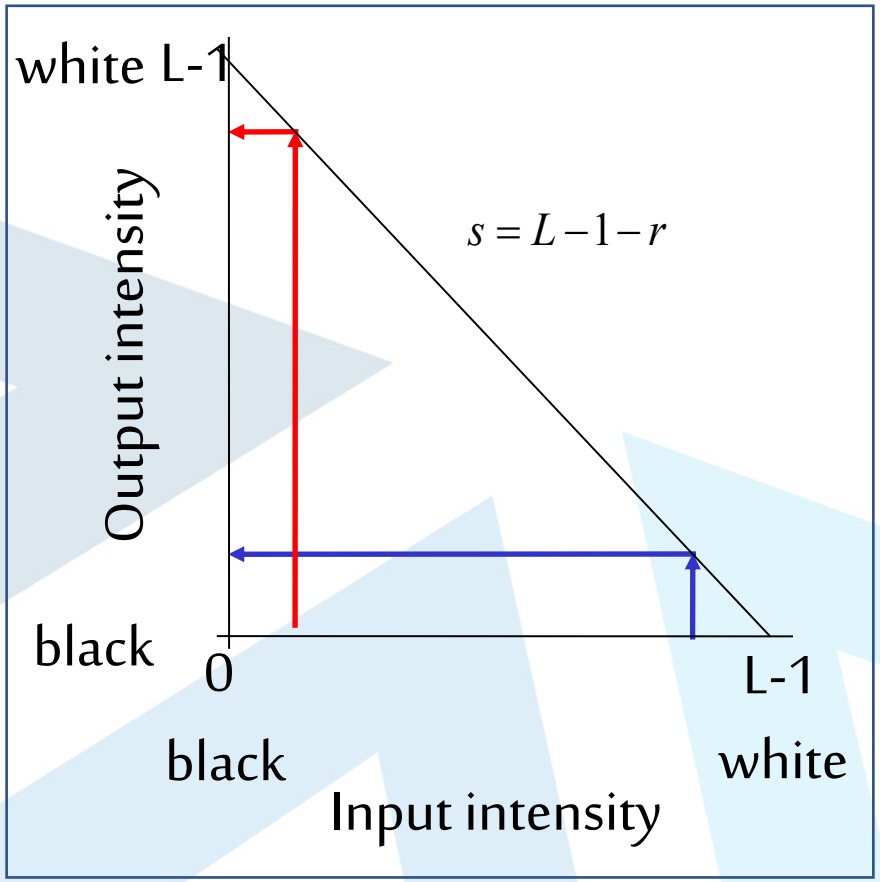
$$s = L - 1 - r.$$

- تستخدم عادة بالصور الطبية لإظهار أكبر قدر ممكن من التمايز و يعتمد بشكل رئيسي على تبديل درجات اللون الأبيض بالأسود و تمايزات الألوان الأخرى بمثيلاتها.



في التطبيقات الطبية

ملاحظة: نظرًا لأن العين تستجيب لوغاريتميًا لتغيرات السطوع، فقد يتعذر اكتشاف التغيرات الطفيفة في السطوع في المناطق الساطعة من الصورة. مع العملية not، تتحول تغييرات السطوع الدقيقة هذه إلى مناطق مظلمة بحيث تصبح مرئية بوضوح



L = number of gray levels

Log Transformations
التابع اللوغاريتمي

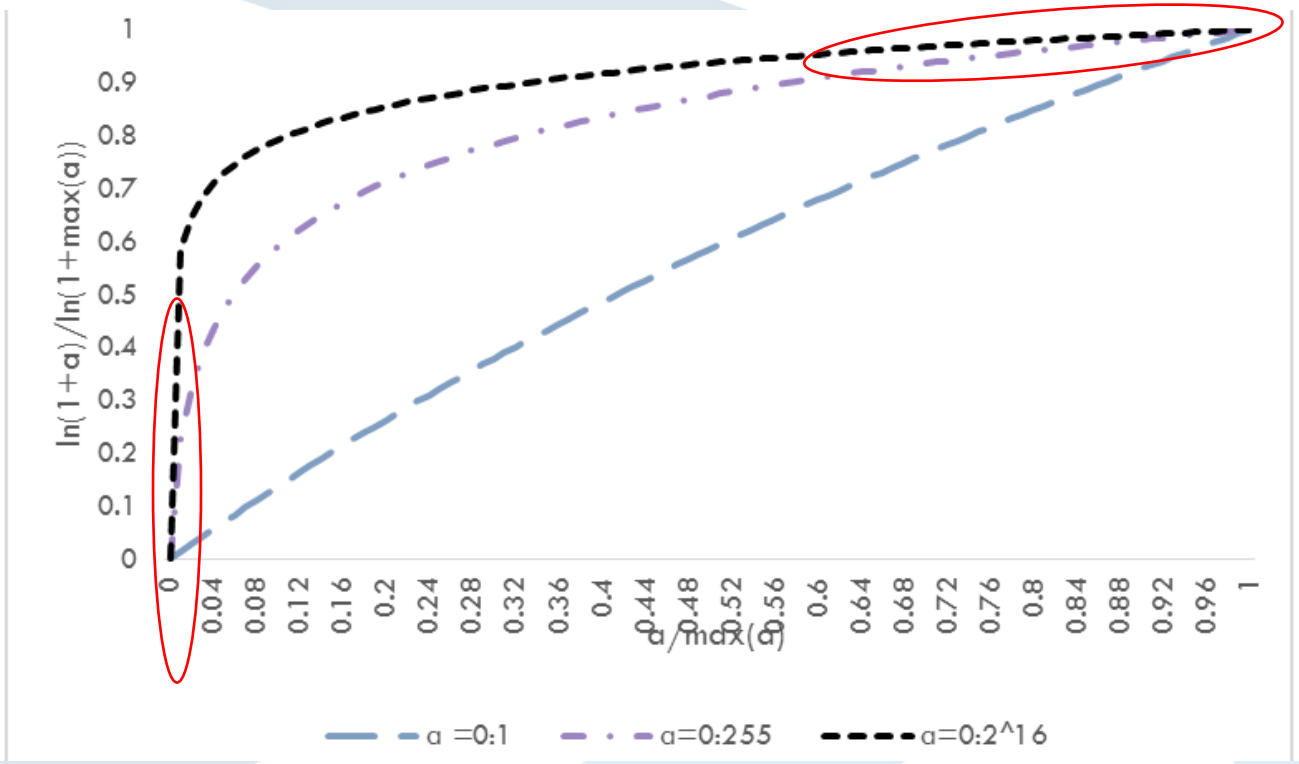
يمكن ضغط المجال الديناميكي للصورة باستبدال قيمة كلّ بكسل باللوغاريتمّ الطبيعي لهذه القيمة

- يطبق وفقا للمعادلة التالية:

حيث: $s = c\log(r+1)$

  - المقدار c عبارة عن ثابت
  - القيمة r أكبر أو تساوي الصفر.
- يعطي هذا التحويل حالات تمايز لونية أكثر دقة.

- Compresses the dynamic range of images with large variations in pixel values

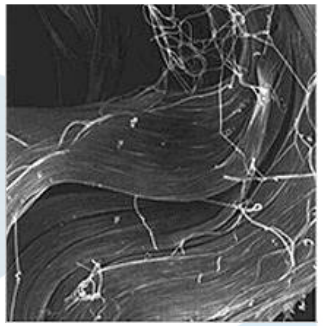


تغيّرشكل التابع اللوغاريتمي تبعاً لتغيّرنوع معطيات الدخل

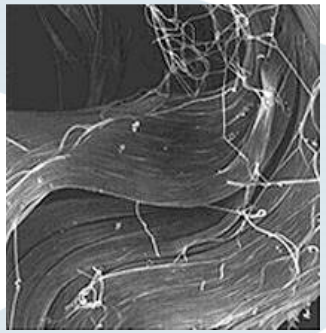
- يقوم بزيادة تباين السويات الرمادية المنخفضة في صورة الدخل فيخصص لها مجال واسع من السويات الرمادية في صورة الخرج بينما يقوم بضغط مجال السويات الرمادية العالية

تطبيق التابع اللوغاريتمي على صور ذات مجالات لونية مختلفة

- يستخدم تابع اللوغاريتم في زيادة سطوع السويات الرمادية المنخفضة وإظهار تفاصيل أكبر في الأماكن الداكنة في الصورة

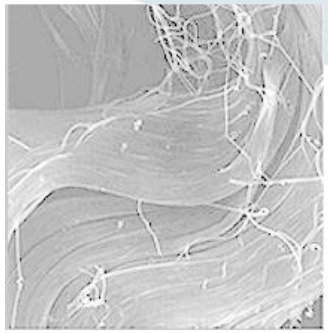


الصورة الأصلية



تابع اللوغاريتمّ في المجال

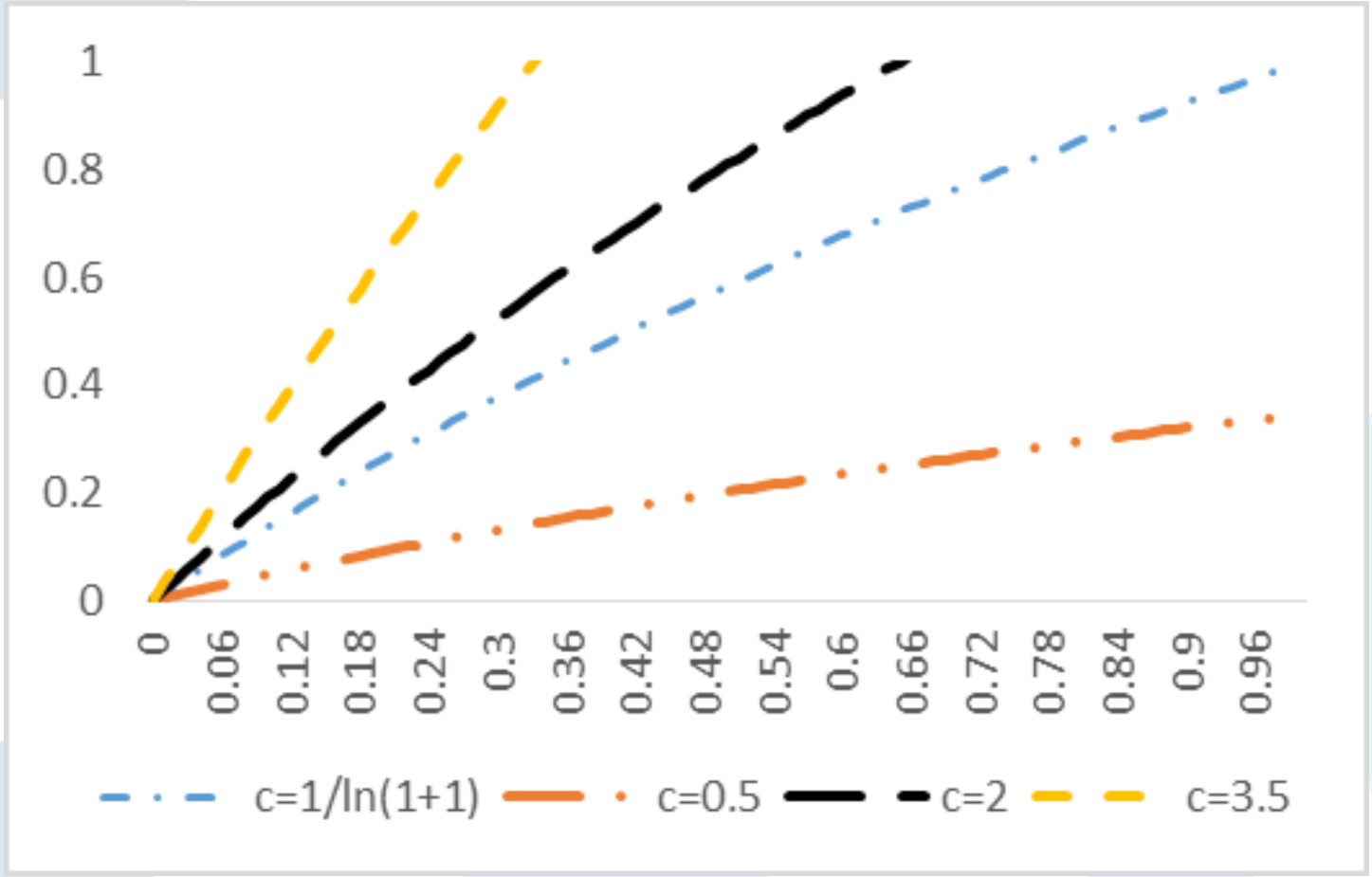
[1 0]

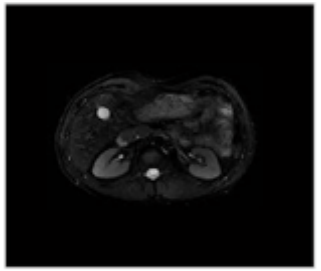


تابع اللوغاريتمّ في المجال

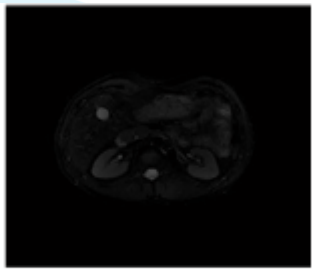
[255 0]

تأثير تغيير قيمة الثابت c في خواص التابع اللوغاريتمي

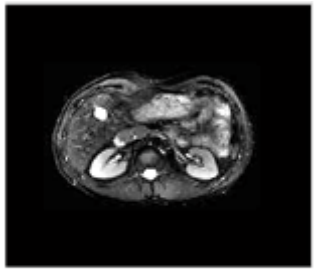




الصورة الأصلية



c=0.5



c=3.5

```
import cv2
import numpy as np
import math


def apply_log_transform(image):
    # Normalize pixel values to 0-1 range for log calculation
    # Then apply log and scale back to 0-255
    # The constant 'c' is chosen such that the max output value is 255
    c = 255 / math.log(1 + np.max(image)) if np.max(image) > 0 else 1.0

    # Build a lookup table
    # Convert i to float before log to avoid integer division issues
    table = np.array([c * math.log(1 + i) for i in np.arange(0, 256)]).astype("uint8")

    # Apply the transformation using the lookup table
    return cv2.LUT(image, table)
```

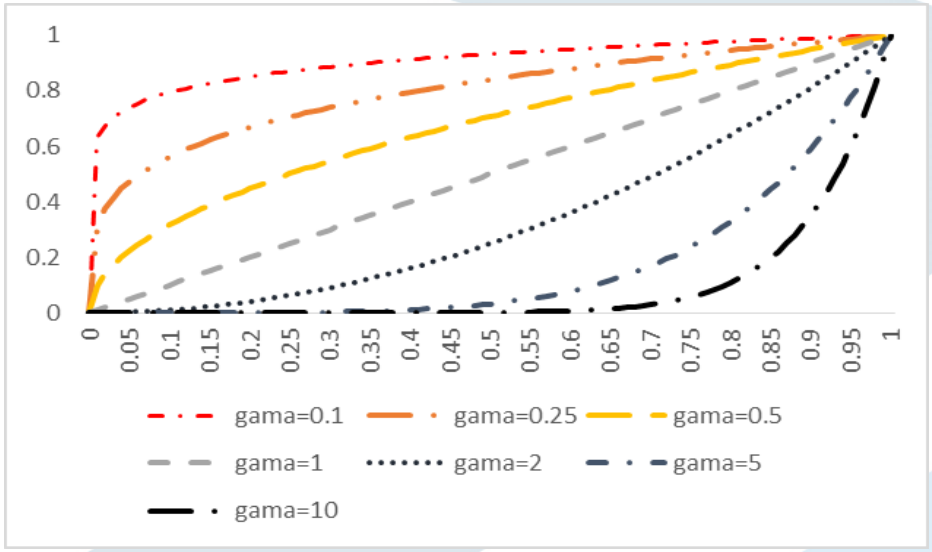
s = c * log(1 + r)

```
if __name__ == "__main__":
    image_path = 'example.jpg'
    original_image = cv2.imread(image_path)
    if original_image is None:
        print(f"Error: Could not load image from {image_path}")
        print("Please make sure 'example.jpg' exists or provide a correct path.")
    else:
        cv2.imshow("Original Image", original_image)
        log_transformed_image = apply_log_transform(original_image)
        cv2.imshow("Logarithmic Transformed Image", log_transformed_image)

        print("\nPress any key to close the windows...")
        cv2.waitKey(0)
        cv2.destroyAllWindows()
```

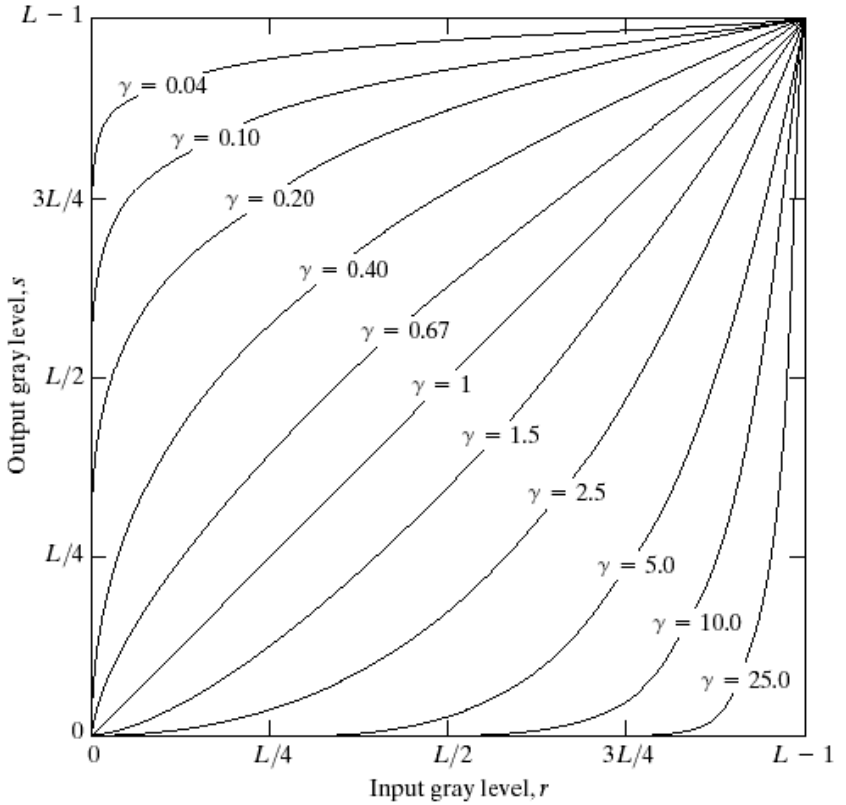

Original Image
Logarithmic Transformed Image

Power-Law Transform

- يقوم برفع قيمة كل بكسل في الصورة إلى قوة ثابتة $s = cr^{\gamma}$ حيث c و γ قيماً موجبة أو بالشكل التالي: $s = c(r + \varepsilon)^{\gamma}$
- يعطي خصائص مماثلة لكل من التابع اللوغاريتمي والتابع الأسي تبعاً لـ γ ويعتبر بديلا عنهما



يمثل المنحني التالي مخططات هذه القيمة:



- $\gamma > 1$ تحسين المناطق المشرقة في الصورة على حساب المناطق الداكنة
- $\gamma < 1$ عكس ذلك

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

$\gamma = c = 1$: identity

Power-Law Transform

a b
c d

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.)

$s = r^3$

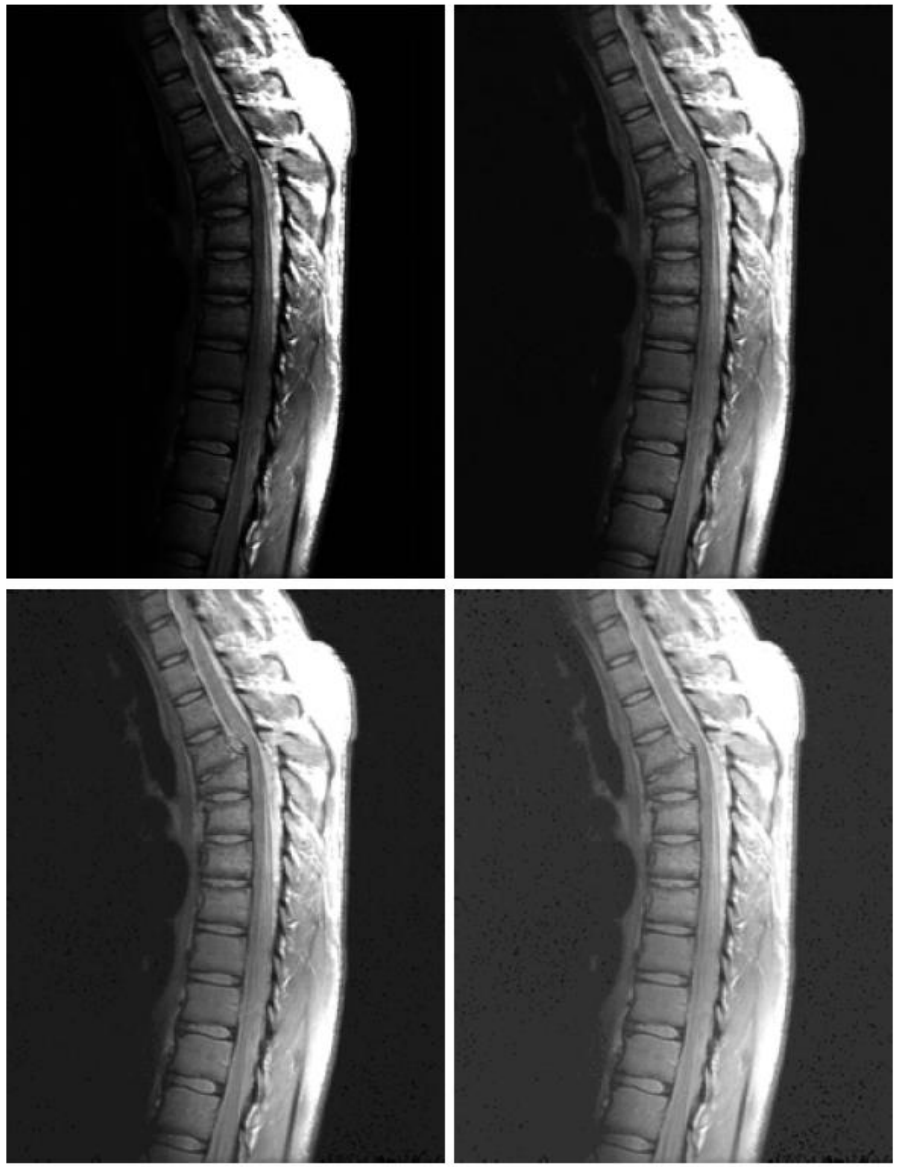
$s = r^4$

$s = r^5$

الشكل التالي يوضح صورة شديدة السطوع و تحتاج إلى درجة من التعتيم لزيادة وضوحها

Power-Law Transformations :

γ=0.6
$s = r^{0.6}$



a b
c d

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.)

γ=0.4
$s = r^{0.4}$

γ=0.3
$s = r^{0.3}$

•إن تغيير العامل غاما لا يغير فقط من السطوع و لكن من إشباع الألوان الثلاثة الأساسية, وبعض النظم تمتلك عوامل غاما مختلفة و موزعة جدا في نفس الجهاز, والمثال التالي يوضح عملية التعديل باستخدام المرنان المغناطيسي.

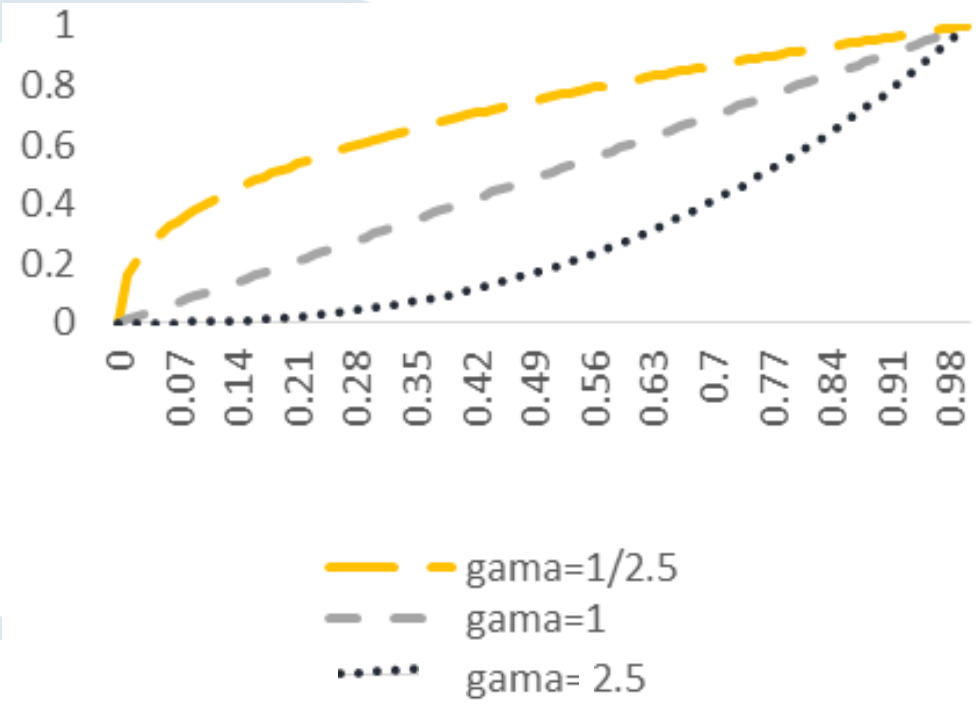
The MRI images using the Gamma Correction settings.

- The images to the right show a magnetic resonance (MR) image of a fractured human spine
- Different curves highlight different detail

Gamma Correction

- إن معظم أجهزة العرض تملك علاقة غير خطية بين جهد الدخل والشدة الضوئية، إذ تختلف الشدة الضوئية بمقدار يعادل جهد الدخل مرفوعاً للقوة γ

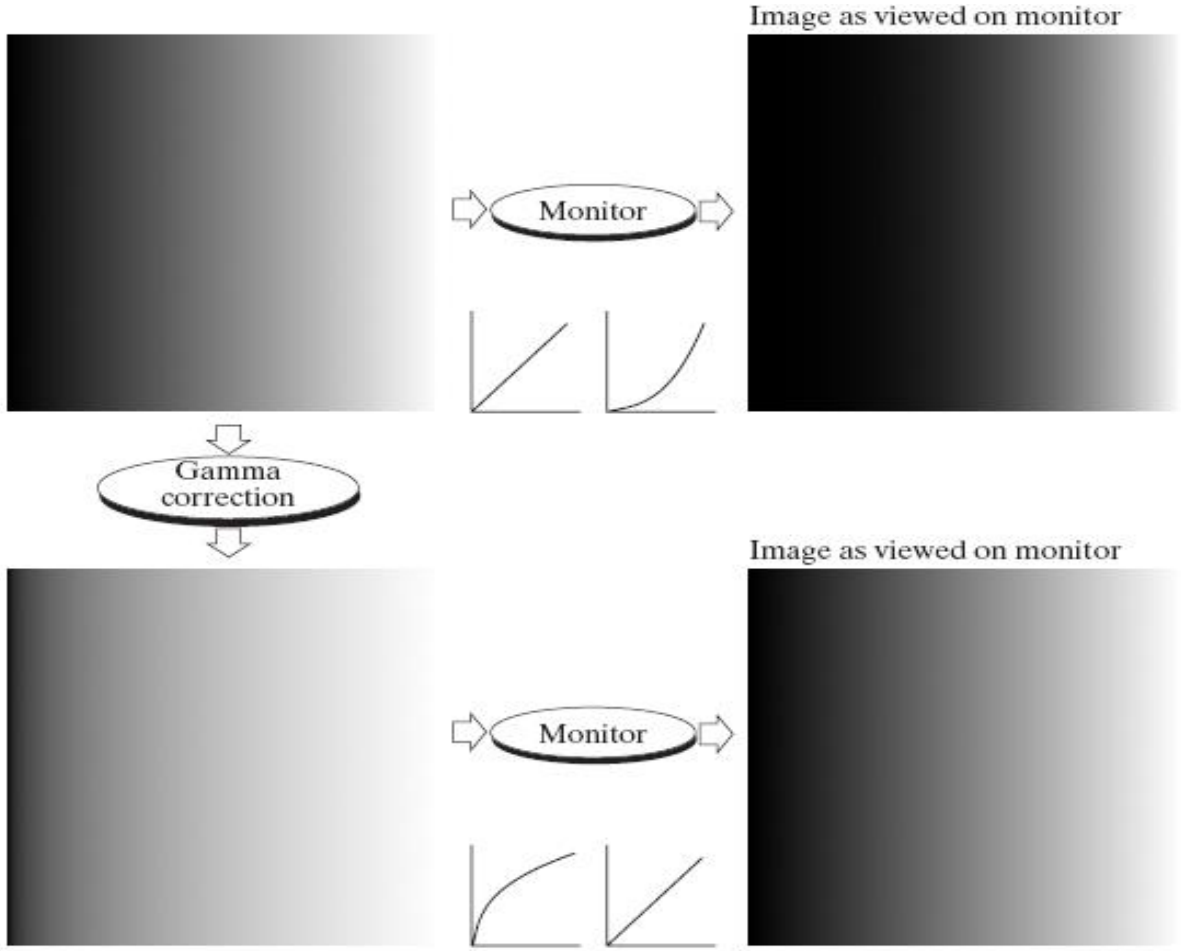
- تتم معالجة قيم الشدة الضوئية باستخدام تابع القوة بقيمة 1/γ



$$B = \left((A)^{\frac{1}{\gamma}}\right)^{\gamma} = A$$

Gamma Correction

Desired Image



a b
c d

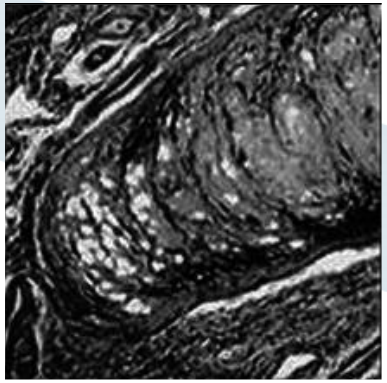
FIGURE 3.7
(a) Linear-wedge gray-scale image. (b) Response of monitor to linear wedge. (c) Gamma-corrected wedge. (d) Output of monitor.

Image shown directly on the monitor

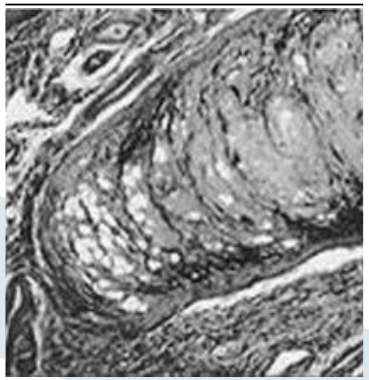
adjusting by Gamma correction

Image shown at Monitor later

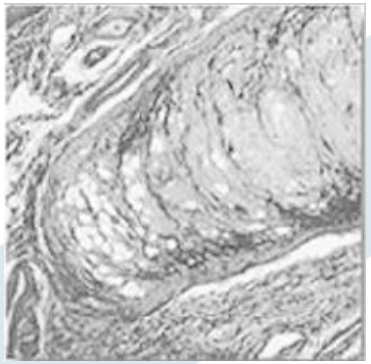
Gamma Correction



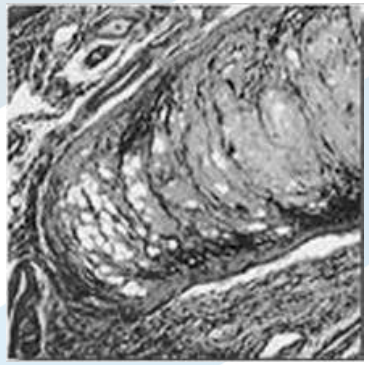
الصورة التي تظهر على أجهزة العرض عند
$\gamma = 2.5$



الصورة الأصلية



الصورة الناتجة عن تطبيق تابع القوة عند
$\gamma = 1/2.5$



الصورة الناتجة عن تصحيح جاما

Matlab code

```
a=imread('elastic.jpg');
b=imadjust(a,[],[],2.5);

c=imadjust(a,[],[],1/2.5);

gamma=imadjust(b,[],[],1/2.5);
```

J = imadjust(I,[low_in high_in],[low_out high_out],gamma)

```
import cv2
import numpy as np
def apply_gamma_correction(image, gamma=1.0):
    inv_gamma = 1.0 / gamma
   table = np.array([((i / 255.0) ** inv_gamma) * 255 for i in np.arange(0, 256)]).astype("uint8")
   return cv2.LUT(image, table)
if __name__ == "__main__":
     image_path = 'example.jpg'
    original_image = cv2.imread(image_path)
    if original_image is None:
        print(f"Error: Could not load image from {image_path}")
        print("Please make sure 'example.jpg' exists or provide a correct path.")
else:
    # Display the original image
    cv2.imshow("Original Image", original_image)
    # 2. Apply Gamma Correction with different gamma values
    # Make the image brighter (gamma < 1.0)
    gamma_bright = 0.6
    bright_image = apply_gamma_correction(original_image, gamma=gamma_bright)
    cv2.imshow(f"Gamma Corrected (Gamma={gamma_bright}) - Brighter", bright_image)

    # Make the image darker (gamma > 1.0)
    gamma_dark = 2.0
    dark_image = apply_gamma_correction(original_image, gamma=gamma_dark)
    cv2.imshow(f"Gamma Corrected (Gamma={gamma_dark}) - Darker", dark_image)

    # 3. Wait for a key press and then close all windows
    print("\nPress any key to close the windows...")
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```


Original Image
Gamma Corrected (Gamma=0.6) - Brighter
Gamma Corrected (Gamma=2.0) - Darker

def apply_gamma_correction(image, gamma=1.0) :

inv_gamma = 1.0 / gamma #We calculate the inverse of the gamma value because the formula for gamma correction usually applies (pixel_value / 255.0) ** (1 / gamma).

table = np.array([...]).astype("uint8") # This is the core of the operation. We create a lookup table (LUT).

- For each possible pixel intensity value from 0 to 255 (np.arange(0, 256)):
- We normalize the value to the range [0, 1] by dividing by 255.0.
- We raise this normalized value to the power of inv_gamma.
- We then multiply by 255 again to scale it back to the [0, 255] range.
- Finally, we cast it to uint8 (unsigned 8-bit integer), which is the standard data type for pixel values in images.

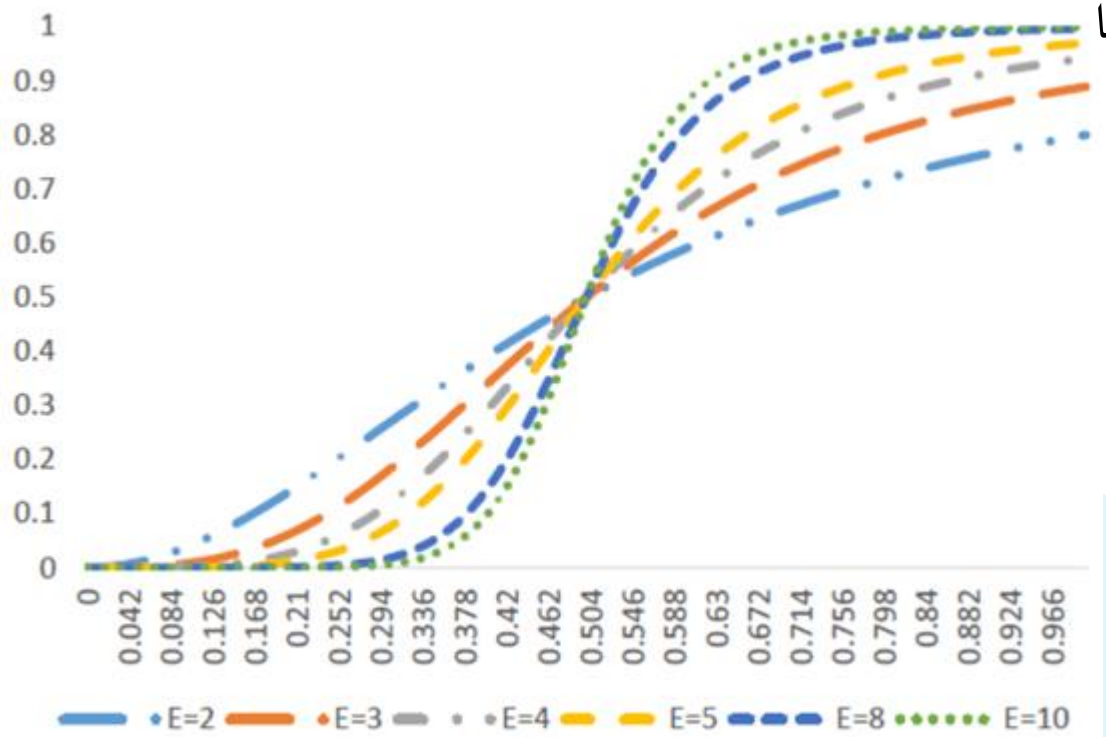
return cv2.LUT(image, table) # OpenCV's cv2.LUT() function efficiently applies this lookup table to every pixel in the input image. For each pixel in the image, its intensity value is used as an index into the table, and the corresponding value from the table replaces the original pixel value.

تعديل التباين وتابع سيغمويد sigmoid

- يعد اكثر تطبيقات توابع التحويل النقطية شيوعا
- يوسع السويات الرمادية ويعدل التباين
- يسمى هذا التابع بتابع سيغمويد

Matlab code

```
I=imread('Nebula.gif');
a=im2double(I);
m=mean2(a);
contrast1=1./(1+(m./(a
+eps)).^2);
contrast2=1./(1+(m./(a
+eps)).^5);
contrast3=1./(1+(m./(a
+eps)).^10);
```



تأثير تغيير قيمة E في ميل تابع سيغمويد.

$$B = \frac{1}{1 + \frac{m}{A^E}}$$

تعبّر m عن القيمة المتوسطة للتابع. تكون الصورة الناتجة أكثر اشراقا كلما كانت قيمة m أكبر والعكس صحيح.

تحدد E مقدار ميل منحني التابع وتعدّ الميزة الأساس والأكثر فائدة في توسعة التباين لكن زيادة قيمة E بشكل كبير يحول تابع توسعة التباين إلى تعتيب.

تعديل التباين وتابع سيغمويد sigmoid

الصورة الأصلية

E=2

E=5

E=10

The general form of a sigmoid function is:

$$S(x) = 1 / (1 + e^{(-k * (x - x0))})$$

Where:

- x: The input pixel intensity (typically 0-255).
- k: The gain or slope parameter. A larger k value makes the curve steeper, leading to higher contrast.
- x0: The midpoint or threshold parameter. This is the pixel intensity value at which the output of the sigmoid function will be 0.5 (or 127.5 if scaled to 0-255). Values around x0 will be stretched, while values far from x0 will be compressed.
- For image processing, we'll map the input pixel intensities (0-255) to the sigmoid function and then scale the output back to the 0-255 range.

$$S(x) = 1 / (1 + e^{(-k * (x - x0))})$$

```
import cv2
import numpy as np
import math

def apply_sigmoid_transform(image, gain=5.0, midpoint=127.5):
    # Create a lookup table for all possible 256 pixel values
    # Each value 'i' is normalized to [0, 1] for calculation if needed,
    # then passed through the sigmoid and scaled back to [0, 255].
    # We use (i - midpoint) to center the sigmoid around the midpoint
    # and then scale the output from [0, 1] to [0, 255].
    table = np.array([255 * (1 / (1 + math.exp(-gain * (i - midpoint) / 255)))
                      for i in np.arange(0, 256)]).astype("uint8")

    # Apply the transformation using the lookup table
    return cv2.LUT(image, table)
```

```
# --- Main Part of the Script ---
if __name__ == "__main__":
    image_path = 'example.jpg'
    original_image = cv2.imread(image_path)

    if original_image is None:
        print(f"Error: Could not load image from {image_path}")
        print("Please make sure 'example.jpg' exists or provide a correct path.")
    else:
        # Display the original image
        cv2.imshow("Original Image", original_image)
        # 2. Apply Sigmoid Transformation with different parameters
        # Example 1: Default (midpoint=127.5, gain=5.0) - Enhances overall contrast, especially mid-tones
        sigmoid_default_image = apply_sigmoid_transform(original_image)
        cv2.imshow("Sigmoid Transformed (Default)", sigmoid_default_image)

        # Example 2: Higher gain (steeper curve, more aggressive contrast)
        sigmoid_high_gain_image = apply_sigmoid_transform(original_image, gain=10.0)
        cv2.imshow("Sigmoid Transformed (High Gain=10.0)", sigmoid_high_gain_image)

        # Example 3: Shift midpoint to darker values (emphasizes darker areas more)
        sigmoid_dark_midpoint_image = apply_sigmoid_transform(original_image, gain=7.0, midpoint=80.0)
        cv2.imshow("Sigmoid Transformed (Midpoint=80.0)", sigmoid_dark_midpoint_image)

        # Example 4: Shift midpoint to brighter values (emphasizes brighter areas more)
        sigmoid_bright_midpoint_image = apply_sigmoid_transform(original_image, gain=7.0, midpoint=180.0)
        cv2.imshow("Sigmoid Transformed (Midpoint=180.0)", sigmoid_bright_midpoint_image)

        # 3. Wait for a key press and then close all windows
        print("\nPress any key to close the windows...")
        cv2.waitKey(0)
        cv2.destroyAllWindows()
```


Original Image
Sigmoid Transformed (Default)
Sigmoid Transformed (High Gain=10.0)


Sigmoid Transformed (Midpoint=80.0)
Sigmoid Transformed (Midpoint=180.0)

جَامعة المَنارة
MANARA UNIVERSITY

نهاية المحاضرة