SYDE 575: Introduction to Image Processing

Image Compression
Part 1: Concepts and
Fixed-rate image compression

Why Compress Images?

- Popularity of digital imaging systems (e.g., digital still-image/video cameras, digital x-ray/CT/MR/ultrasound systems, scanners) has lead to a large volume of imaging data being created each day
- Digital images in an uncompressed form has considerable storage and transmission bandwidth requirements
- To help with these problems, image compression has become very popular

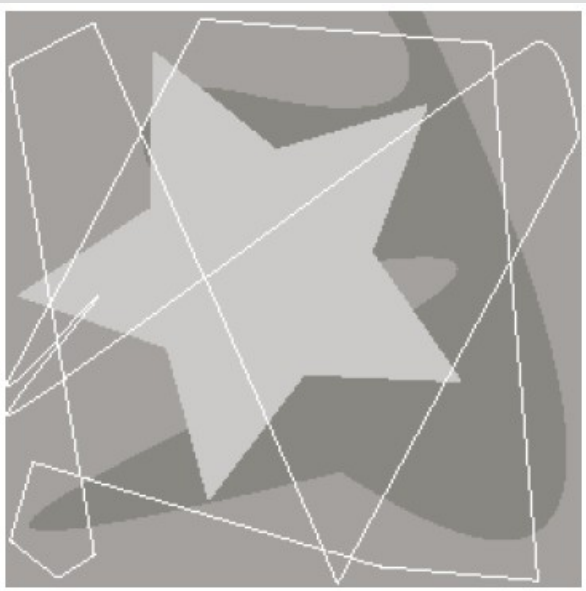
Image Compression

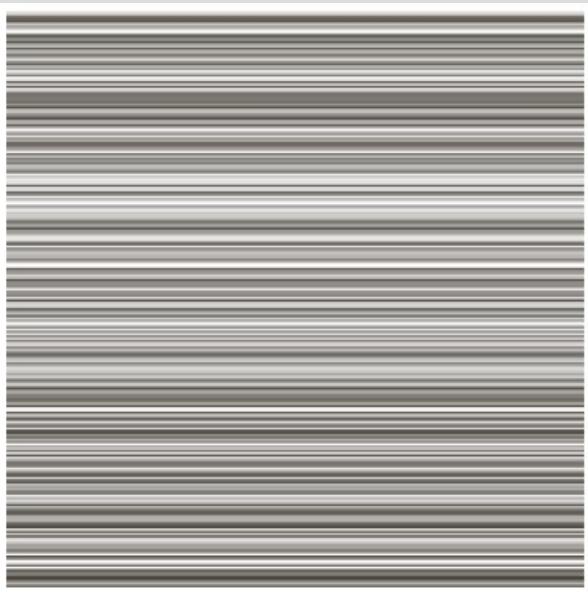
- Goal: reduce amount of data to needs to be stored to represent an image
- Idea: reduce the amount of redundant data that needs to be stored to reconstruct image
- Transform image data into a form that reduces statistical correlation and reduces information redundancy

Types of Redundancy

- Images can be generally characteristics by three types of redundancies
  - Coding redundancy
  - Spatial and temporal redundancy
  - Psycho-visual redundancy

Types of Redundancies





Source: Gonzalez and Woods

Coding Redundancy

- Gray levels in an image can be viewed as random variables
- Histogram shows the probability that each gray level appears in an image
- In most images, certain gray levels are more probable than others
- By adjusting code length based on probability of gray levels, one can reduce coding redundancy

Variable Length Coding

- Decrease code length as probability of occurrence increases
- Example

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_{87} = 87$	0.25	01010111	8	01	2
$r_{128} = 128$	0.47	10000000	8	1	1
$r_{186} = 186$	0.25	11000100	8	000	3
$r_{255} = 255$	0.03	11111111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0

Variable Length Coding

- For code 1, average number of bits required to code image is 8 bits
- For code 2,

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k)$$

$$= l(r_{87}) p_r(r_{87}) + l(r_{128}) p_r(r_{128}) + l(r_{186}) p_r(r_{186}) + l(r_{255}) p_r(r_{255})$$

$$= (2)(0.25) + (1)(0.47) + (3)(0.25) + (3)(0.03)$$

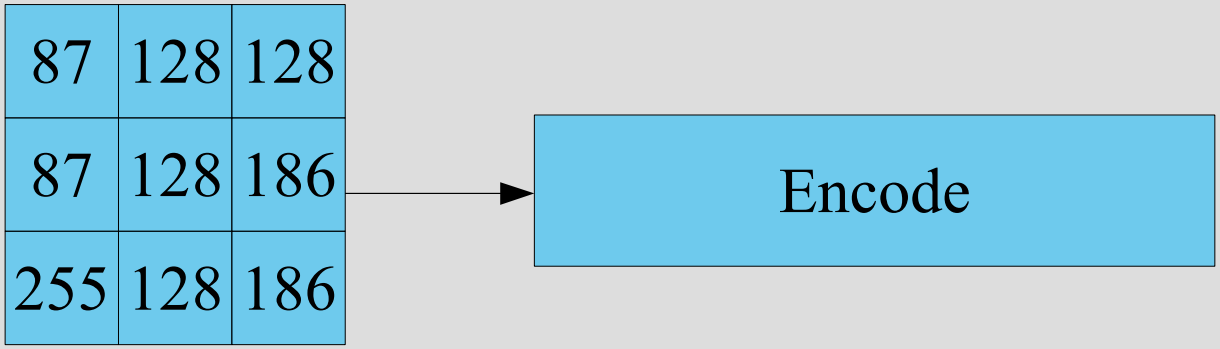
$$= 1.81$$

- Code 1 has high coding redundancy as it requires more bits than necessary to code

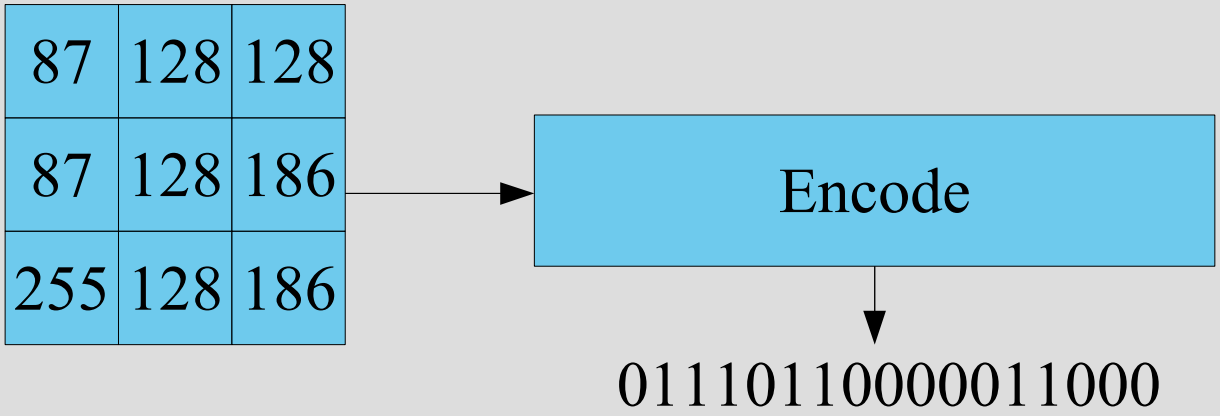
Example: Variable Length Coding

87	128	128
87	128	186
255	128	186

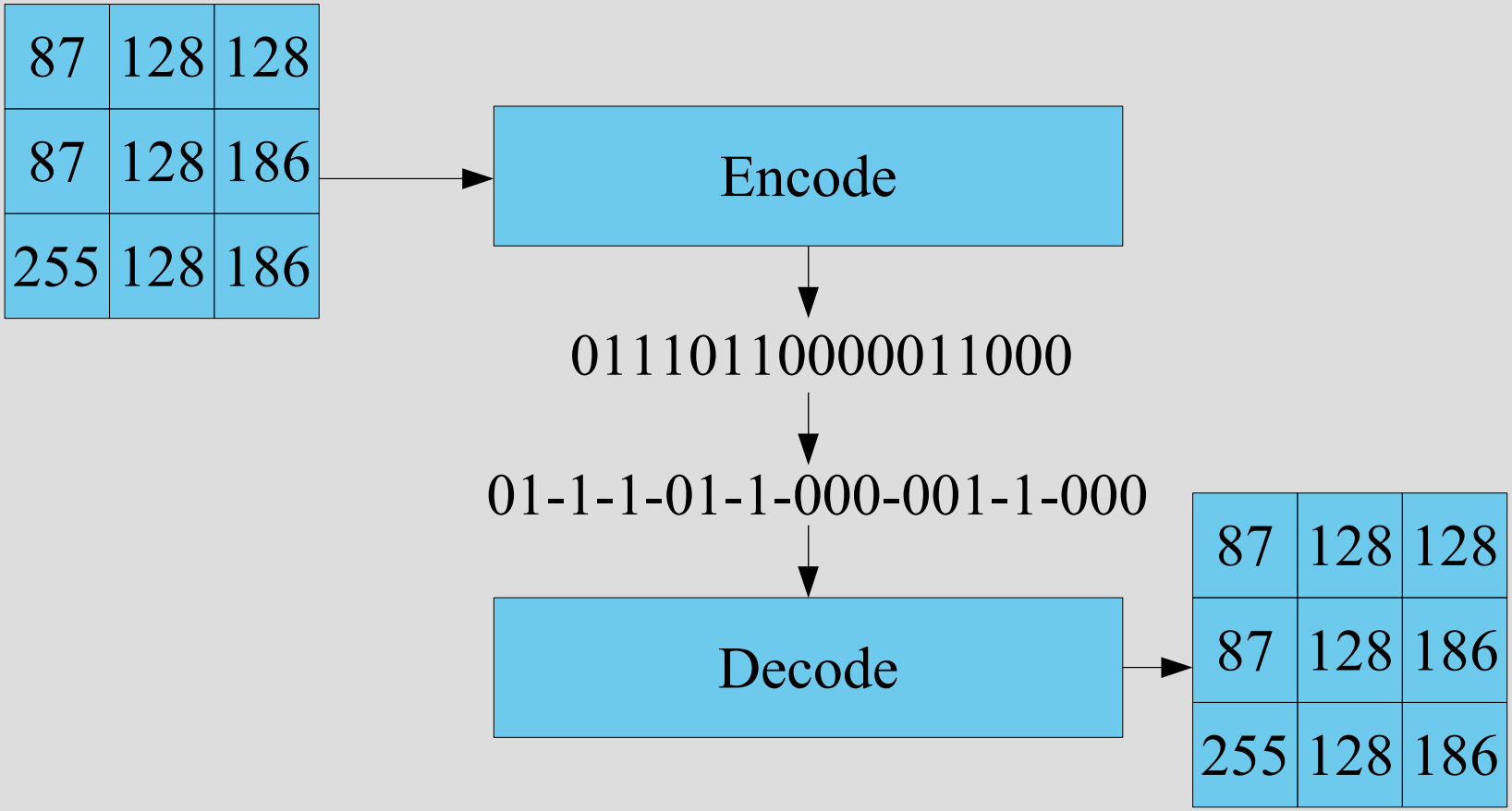
Example: Variable Length Coding



Example: Variable Length Coding



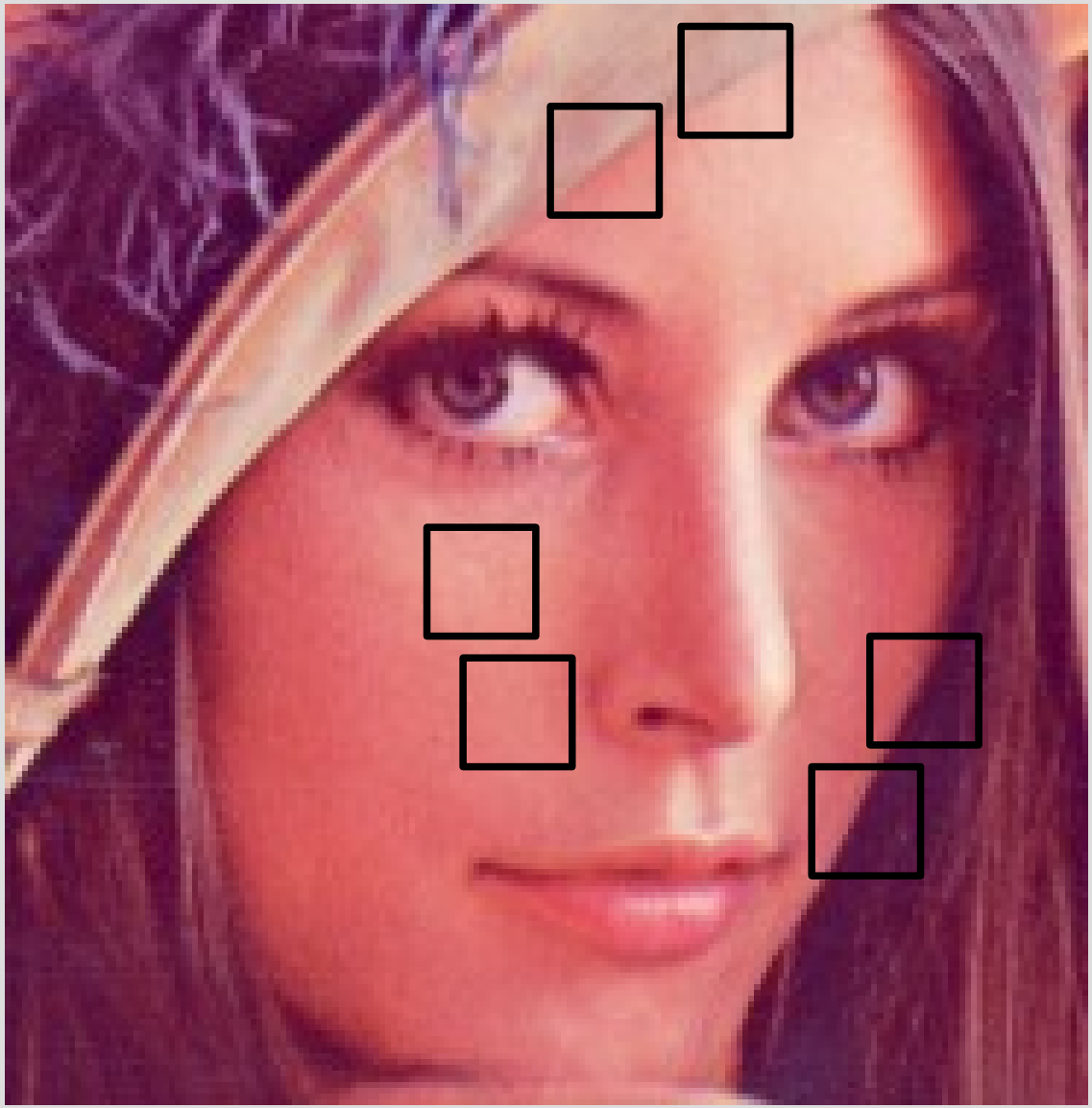
Example: Variable Length Coding



Spatial Redundancy

- Most images possess a lot of structural and intensity self-similarity
- Therefore, value of a given pixel or region can be reasonably predicted by neighboring pixels or regions
- One can take advantage of this self-similar nature of images to reduce the amount of information that needs to be stored.

Example of Spatial Redundancy



Temporal Redundancy

- In a video sequence, pixels and/or regions within each frame are similar to or dependent on pixels and/or regions from adjacent frames.
- Therefore, they can be reasonably predicted by temporally neighboring pixels or regions to reduce the amount of information that needs to be stored.

Example of Temporal Redundancy

Psycho-visual Redundancy

- Eye doesn't respond equally to all visual information
- Certain information less important than others to the human vision system
  - Examples
    - More sensitive to illumination changes than color changes
    - Less sensitive to distortions in textures than uniform regions
    - Less sensitive to distortions at high spatial frequencies

Psycho-visual Redundancy

- Can reduce the amount of information needed to represent an image by removing information that's non-essential for visual processing
  - Examples
    - Store less color information
    - Store less information for uniform regions than structures such as edges
    - Store less information about high frequency image characteristics

Example of Psycho-visual Redundancy

¼ color information

Image Compression Framework

Compress

Storage/
Transmission

Decompress

Types of Image Compression

- **Lossless compression**
  - Exploits coding redundancy
- Advantage: all of the image information is retained (allows for perfect image reconstruction)
- Disadvantage: poor compression performance
- Examples:
  - Run-length encoding
  - Dictionary algorithms (e.g., LZW)
  - Entropy encoding

Types of Image Compression

- **Lossy compression**
  - Decompressed image is different but close enough to original image
  - Exploits spatial and psycho-visual redundancies
- Advantages:
  - High compression performance
- Disadvantages:
  - Loss in original image information
- Examples:
  - JPEG and MPEG

Types of Lossy Compression Schemes

- **Fixed-rate compression**
  - The compression rate achieved is the same for all images
- Advantages
  - Simple and fast decoding
  - Allows for random access of information
- Disadvantages
  - Low compression performance
- Popular application: texture compression for real-time 3D applications (e.g., DXTC)

Types of Lossy Compression Schemes

- **Variable-rate compression**
  - The compression rate achieved varies depending the underlying image characteristics
- Advantages
  - High compression performance
- Disadvantages
  - More complex to encode and decode
  - Difficult to randomly access information
- Popular application: still image and video compression (e.g., JPEG/MPEG)

Fixed-rate Compression: Palette-based Compression

- Idea:
  - As mentioned earlier, values of pixels can be well predicted by neighboring pixels
  - Instead of storing all pixel values, store a few pixel values that are representative of the neighborhood (like colors in a painter's palette)
  - Encode other pixels in the neighborhood as the closest value in the palette

Block Truncation Coding (BTC)

- Block-based algorithm that preserves local mean and standard deviation of image
- Steps:
  1) Divide image into 4x4 pixel blocks
  2) For each block, compute the mean and standard deviation of pixel intensities
- e.g.,

48	48	45	45
49	49	46	45
50	49	44	45
49	50	45	45

$$\mu \approx 47$$

$$\sigma \approx 2$$

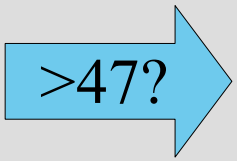
Block Truncation Coding (BTC)

- Steps:
  3) Classify each pixel in the block as follows

$$g(x,y) = \begin{cases} 1, & f(x,y) > \mu \\ 0, & f(x,y) \leq \mu \end{cases}$$

- e.g.,

48	48	45	45
49	49	46	45
50	49	44	45
49	50	45	45



1	1	0	0
1	1	0	0
1	1	0	0
1	1	0	0

Block Truncation Coding (BTC)

- Steps:
  4) Store binary block along with mean and standard deviation
  5) To decode, compute decoding values l (low) and h (high) based on mean and standard deviation

$$l = \mu - \sigma \sqrt{\frac{n_{x>\mu}}{n_{total} - n_{x>\mu}}}$$

$$h = \mu + \sigma \sqrt{\frac{n_{total} - n_{x>\mu}}{n_{x>\mu}}}$$

- e.g., $l = 47 - 2\sqrt{\frac{8}{16-8}} = 45$

$$h = 47 + 2\sqrt{\frac{16-8}{8}} = 49$$

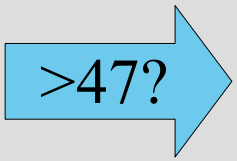
Block Truncation Coding (BTC)

- Steps:
  6) Decode binary block as follows

$$\hat{f} = \begin{cases} h, & g(x,y) = 1 \\ l, & g(x,y) = 0 \end{cases}$$

- e.g.,

1	1	0	0
1	1	0	0
1	1	0	0
1	1	0	0



49	49	45	45
49	49	45	45
49	49	45	45
49	49	45	45

BTC Compression Rate

- Suppose we are given an 4x4 grayscale image, with each pixel represented by a value from 0 to 255.
- The amount of bits required to store this image in an uncompressed format is 4x4x8bits=128 bits
- The bit rate of the image in an uncompressed format is 8 bpp (bits per pixel)

BTC Compression Rate

- Supposed we compress the image using BTC
- The mean and standard deviation each requires 8 bits
- The binary block requires 4x4x1bit=16 bits
- The amount of bits required to store this image in BTC compressed format is 16bits+2x8bits=32 bits
- The bit rate of the image in an BTC compressed format is 32bits/16pixels=2 bpp (bits per pixel) [Compression rate=4:1]

DXTC

- Used for texture compression in the Direct3D standard
- Well suited for 3D real-time applications as it allows for random texel access
- Very fast due to hardware acceleration on all current video cards
- Extends BTC for color images
- In addition to spatial redundancy, also takes advantage of psycho-visual redundancy (through quantization)

DXTC

- Steps:
  1) Divide image into 4x4 blocks
  2) For each block, store two 16-bit representative color values C0 (high) and C1 (low), where
     - 5 bits allocated for red
     - 6 bits allocated for green
     - 5 bits allocated for blue
  3) compute two additional color values

$$c_2 = \frac{2}{3}c_0 + \frac{1}{3}c_1,\ c_3 = \frac{1}{3}c_0 + \frac{2}{3}c_1$$

DXTC

- Steps:
  4) Assign a value from 0 to 3 to each pixel based on which of the four color values they are closest to
     - Creates a 4x4 two-bit lookup table for storage
  5) To decode, replace values from lookup table with one of the four color values

DXTC Compression Rate

- Suppose we are given an 4x4 color image, with each pixel represented by R, G, and B values ranging from 0 to 255 each.
- The amount of bits required to store this image in an uncompressed format is 4x4x(3x8bits)=384 bits
- The bit rate of the image in an uncompressed format is 24 bpp (bits per pixel)

DXTC Compression Rate

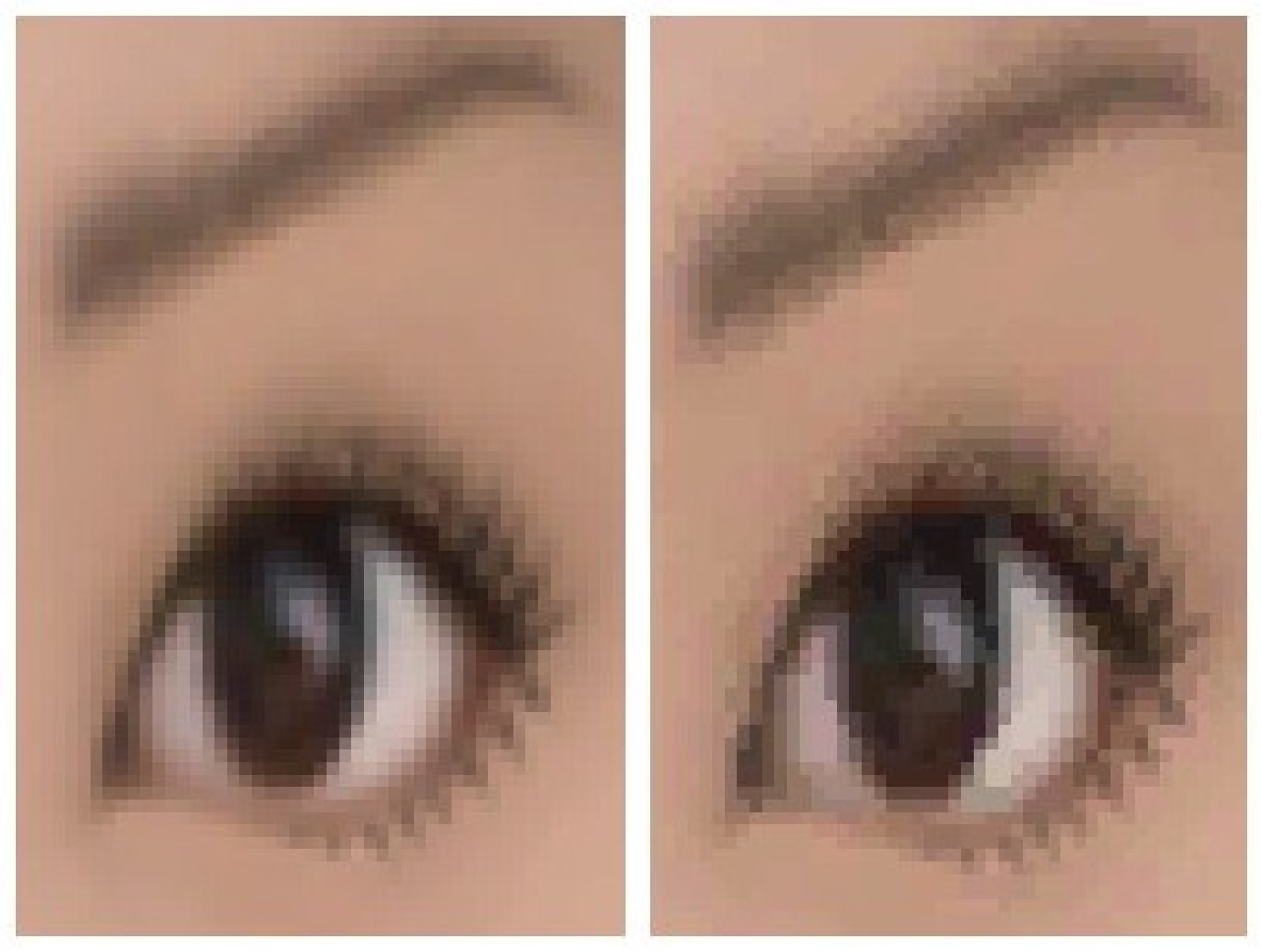
- Supposed we compress the color image using DXTC
- The high and low representative color values C0 and C1 each require 16-bits
- Each value in the 4x4 lookup table represents 4 possible values, thus requiring 4x4x2bit=32 bits
- The amount of bits required to store this color image in DXTC compressed format is 2x16bits+32bits=64 bits

DXTC Compression Rate

- The bit rate of the color image in a DXTC format is 64bits/16pixels=4 bpp (bits per pixel)
- The compression rate of DXTC for the color image can then be computed as $BPP_{uncompressed}:BPP_{DXTC} = 24:4 = 6:1$

Sample Results

- 6:1 compression using DXTC



Observations

- Image remains very sharp and clear
- Solid, uniform regions are well represented
  - Quantization does not perceptually affect image quality in this case
- Blocking artifacts can be seen at smooth transitions
- Reason: using a total of 4 colors does not sufficiently represent such regions, which require more color values to represent the smooth transition