Lecture slides of the course

Information hiding & secret sharing

Image Steganography (P2)

Phạm Trọng Nghĩa

ptnghia@fit.hcmus.edu.vn

This section

- More on the LSB method from the previous session (for nonpalette-based images)
- Palette-based image introduction
- Stagenography on palette-based image

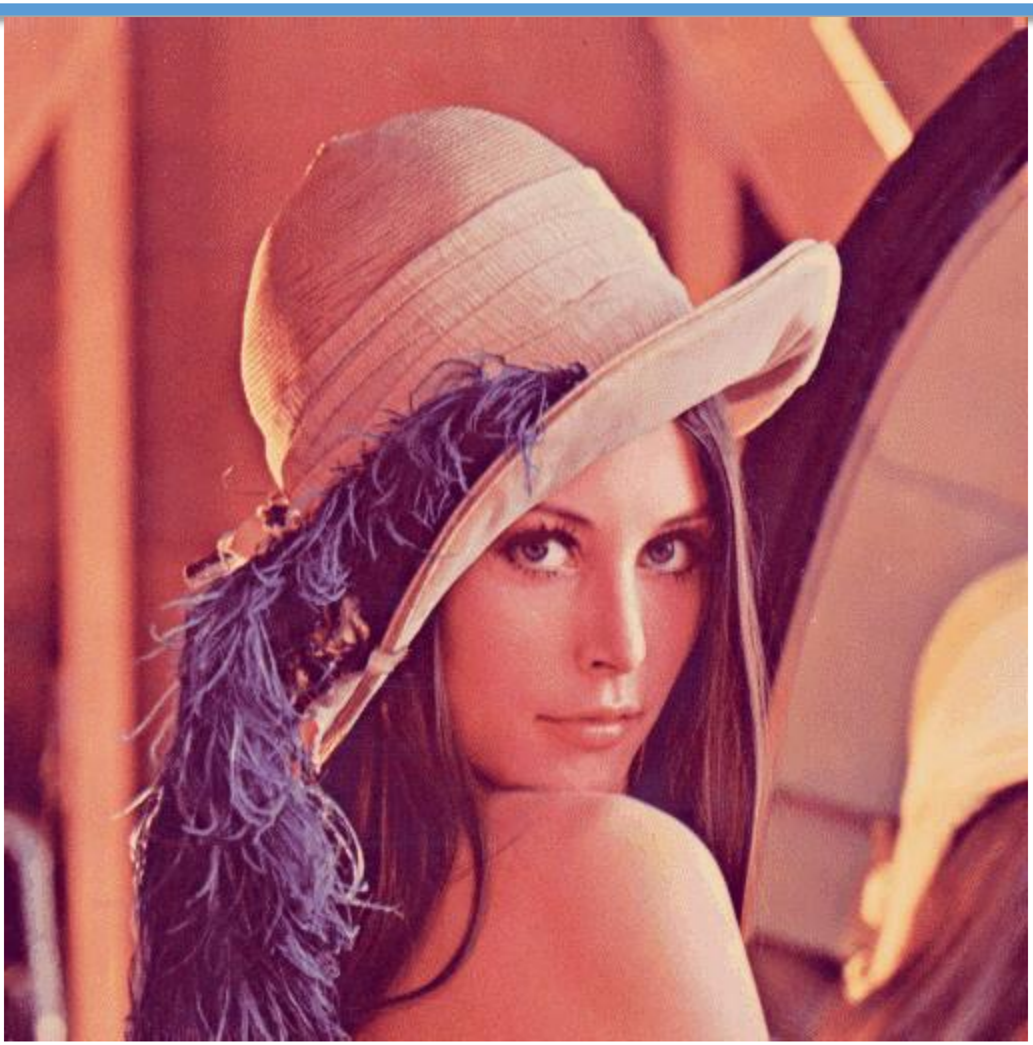
This section

- **More on the LSB method from the previous session (for nonpalette-based images)**
- Palette-based image introduction
- Stagenography on palette-based image

Measure the difference between stego image and cover image

- In addition to the qualitative way of seeing with the eye, we can quantify the difference between two images using the measure of MSE (Mean Squared Error).
- MSE between image ***I*₁** and image ***I*₂** = mean of squared difference between ***I*₁** and ***I*₂**. Example:
    - Given I_1 is a grayscale image consisting of 4 pixels $[1, 3, 5, 7]$
    - Given I_2 is a grayscale image consisting of 4 pixels $[2, 4, 6, 8]$
    - $MSE(I_1, I_2) = \frac{1}{4}\left((1-2)^2 + (3-4)^2 + (5-6)^2 + (7-8)^2\right)$

Stagenography on images using LSB method



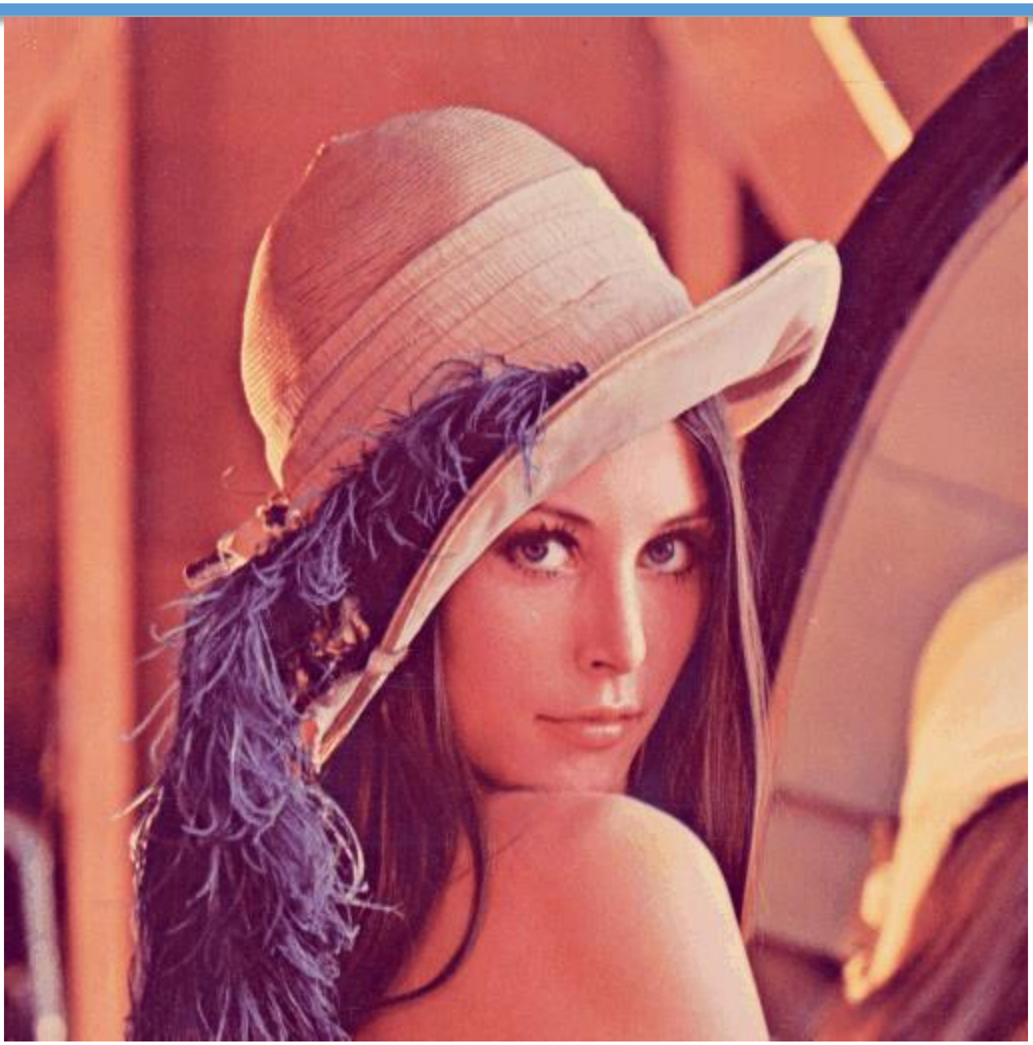
Original Image

Stagenography on images using LSB method

Number of bit LSB (in total 8 bit) embed

k = 1

MSE = 0.5



Stego Image

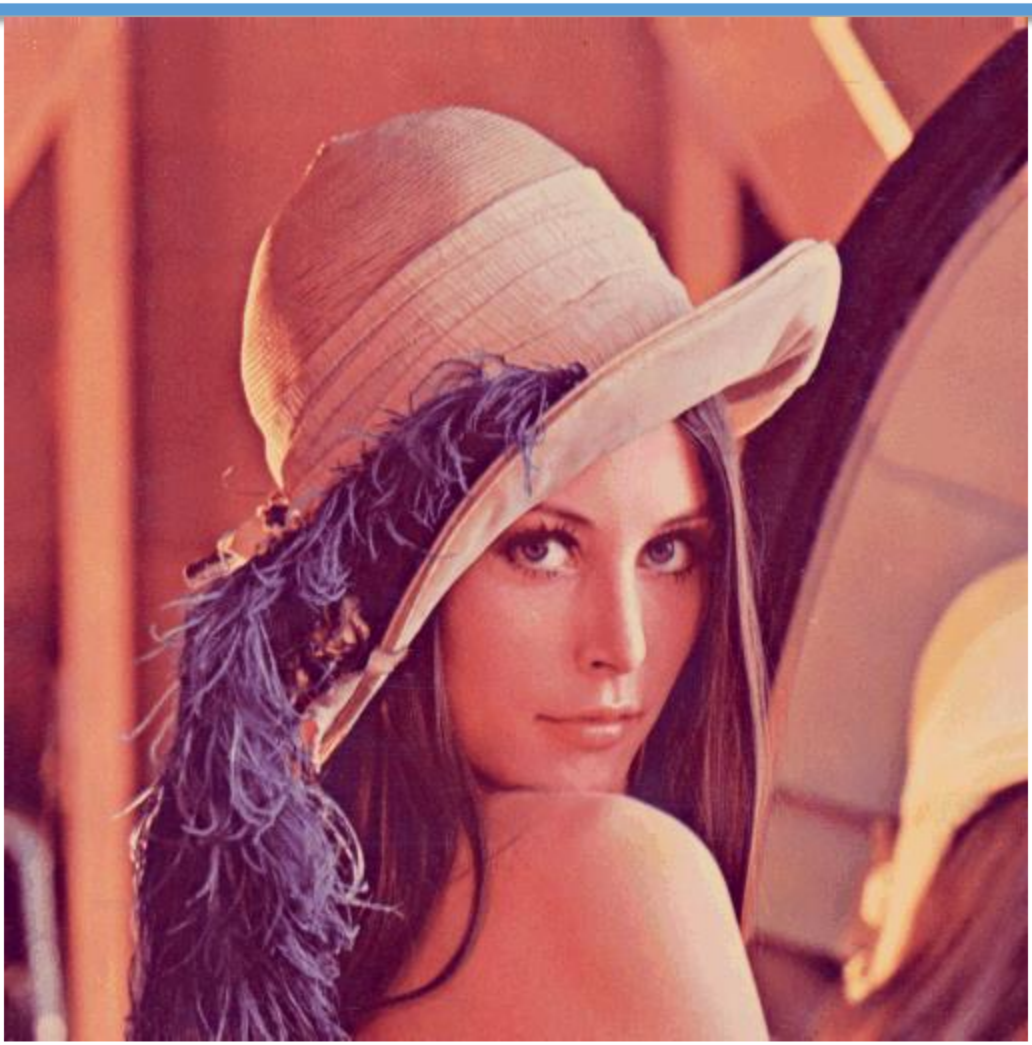
Stagenography on images using LSB method

k = 2
MSE = 1.6

Stego Image

Stagenography on images using LSB method

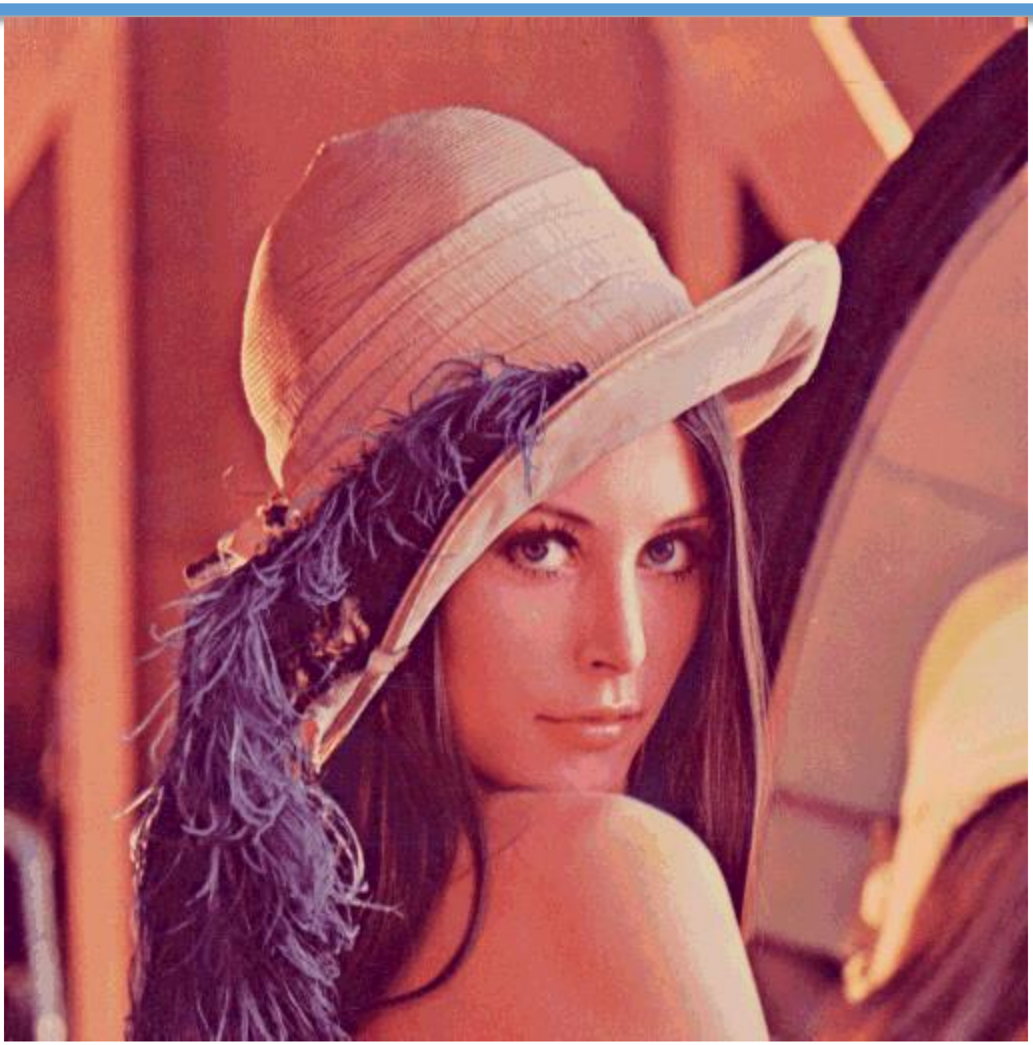
k = 3
MSE = 13.3


Stego Image

Stagenography on images using LSB method

k = 4

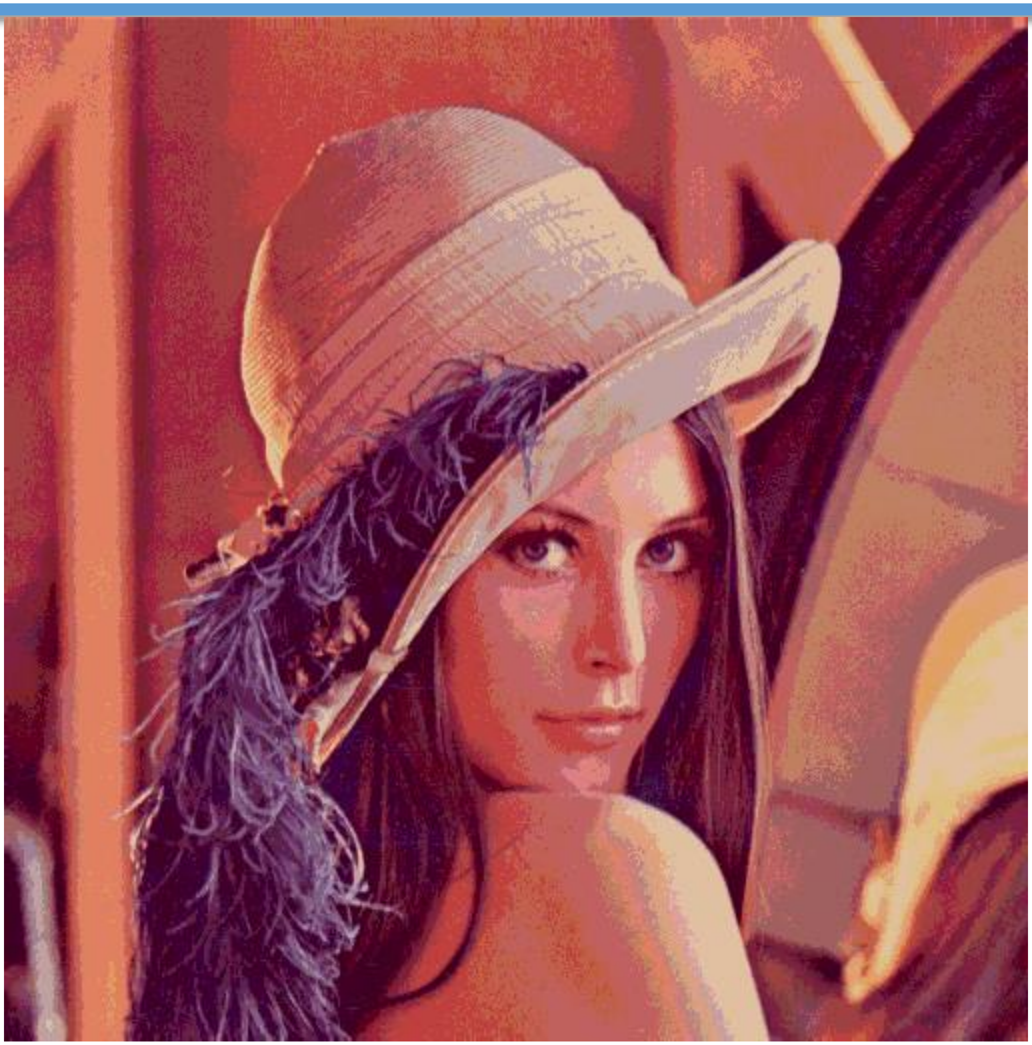
MSE = 70.6



Stego Image

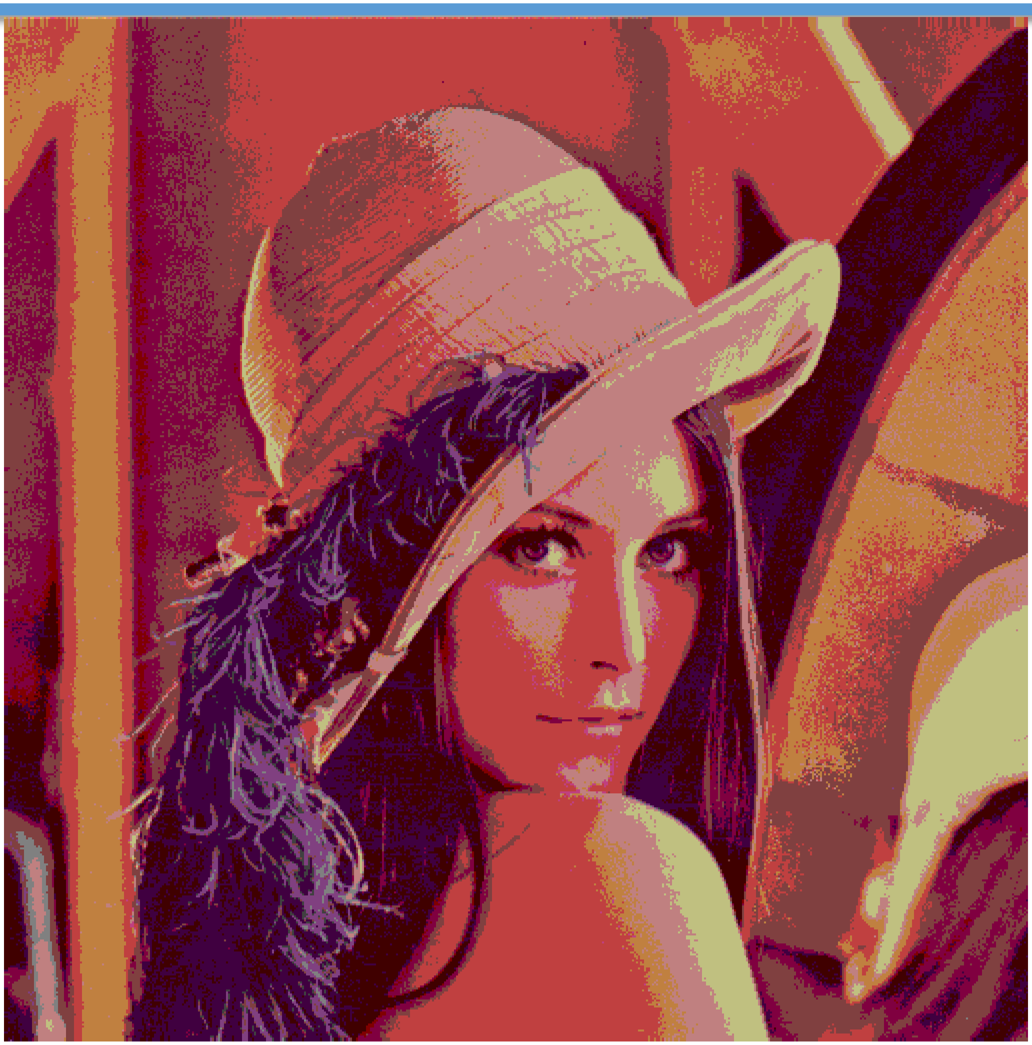
Stagenography on images using LSB method

k = 5
MSE = 320.8


Stego Image

Stagenography on images using LSB method

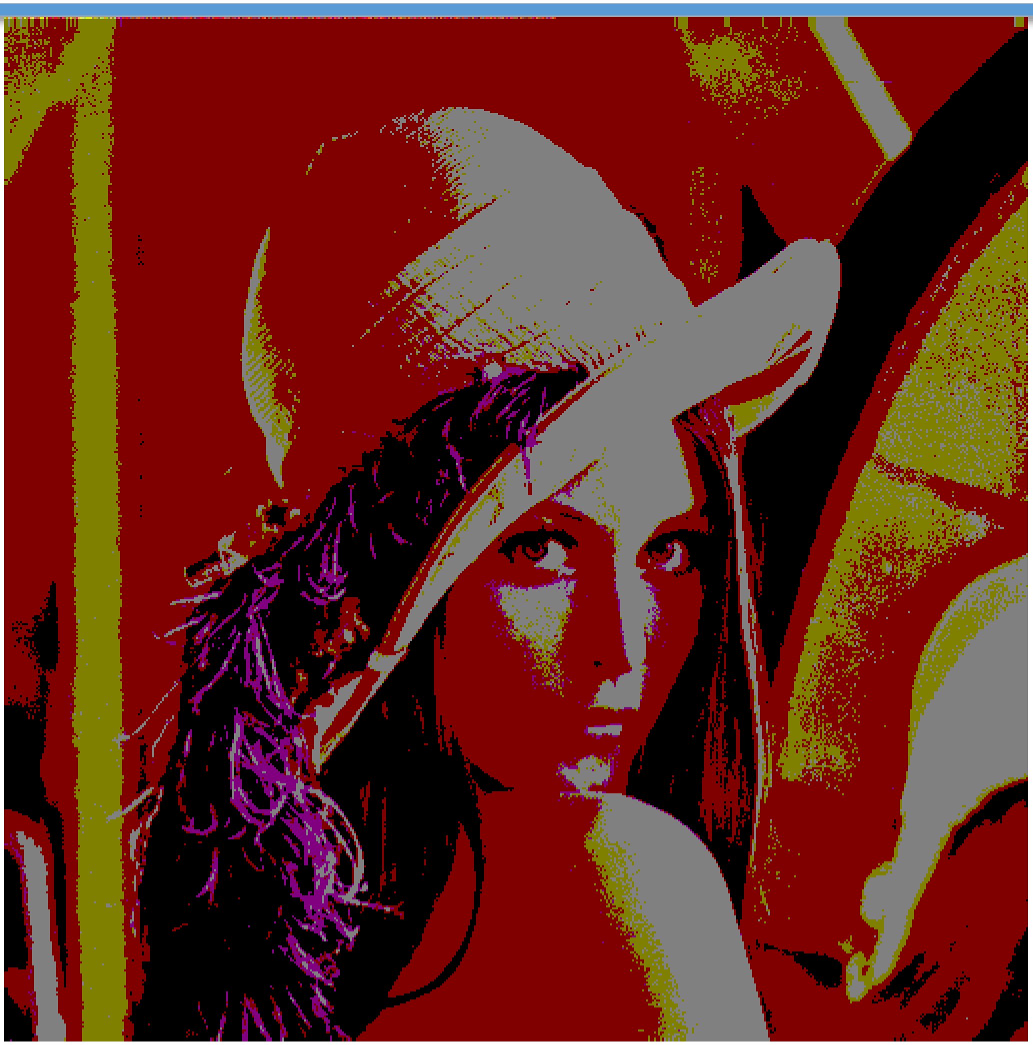
k = 6
MSE = 1255.1


Stego Image

Stagenography on images using LSB method

k = 7

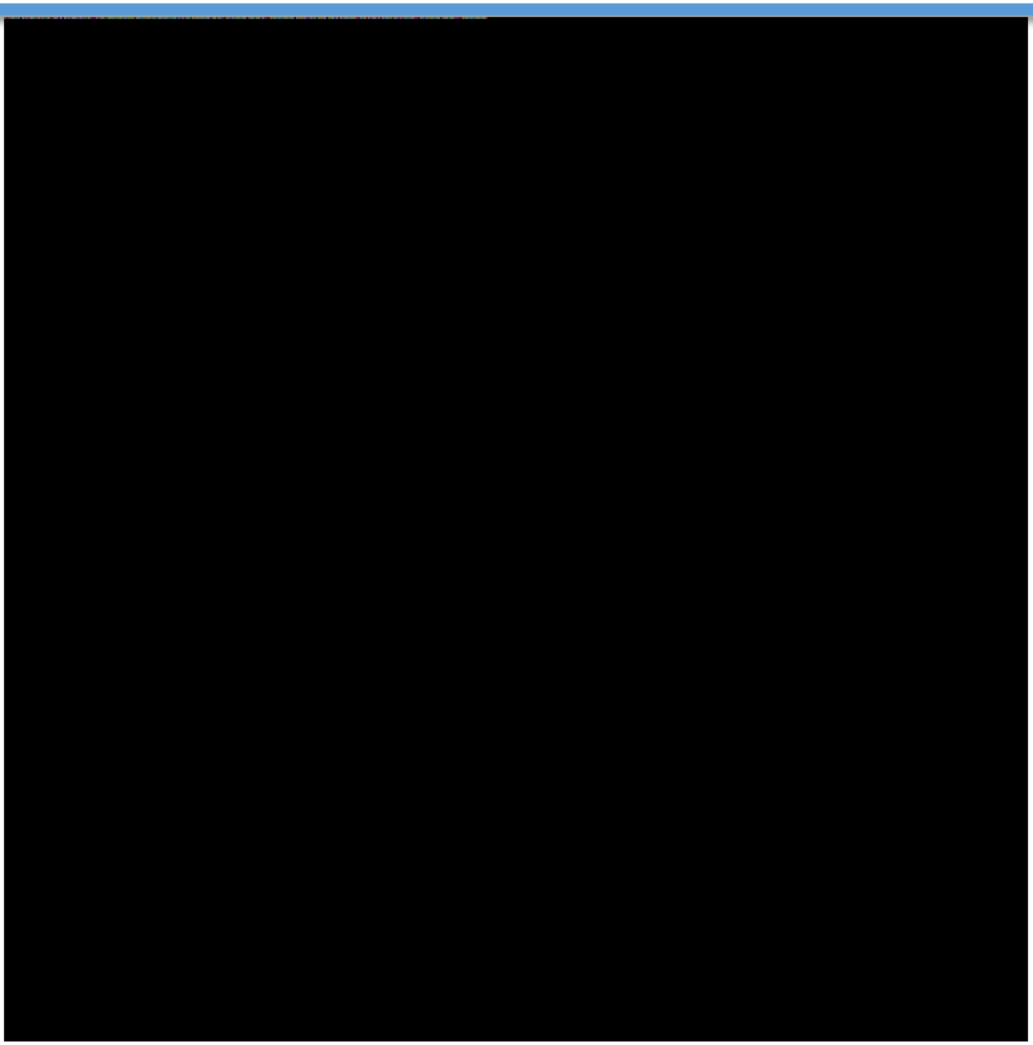
MSE = 6164.3



Stego Image

Stagenography on images using LSB method

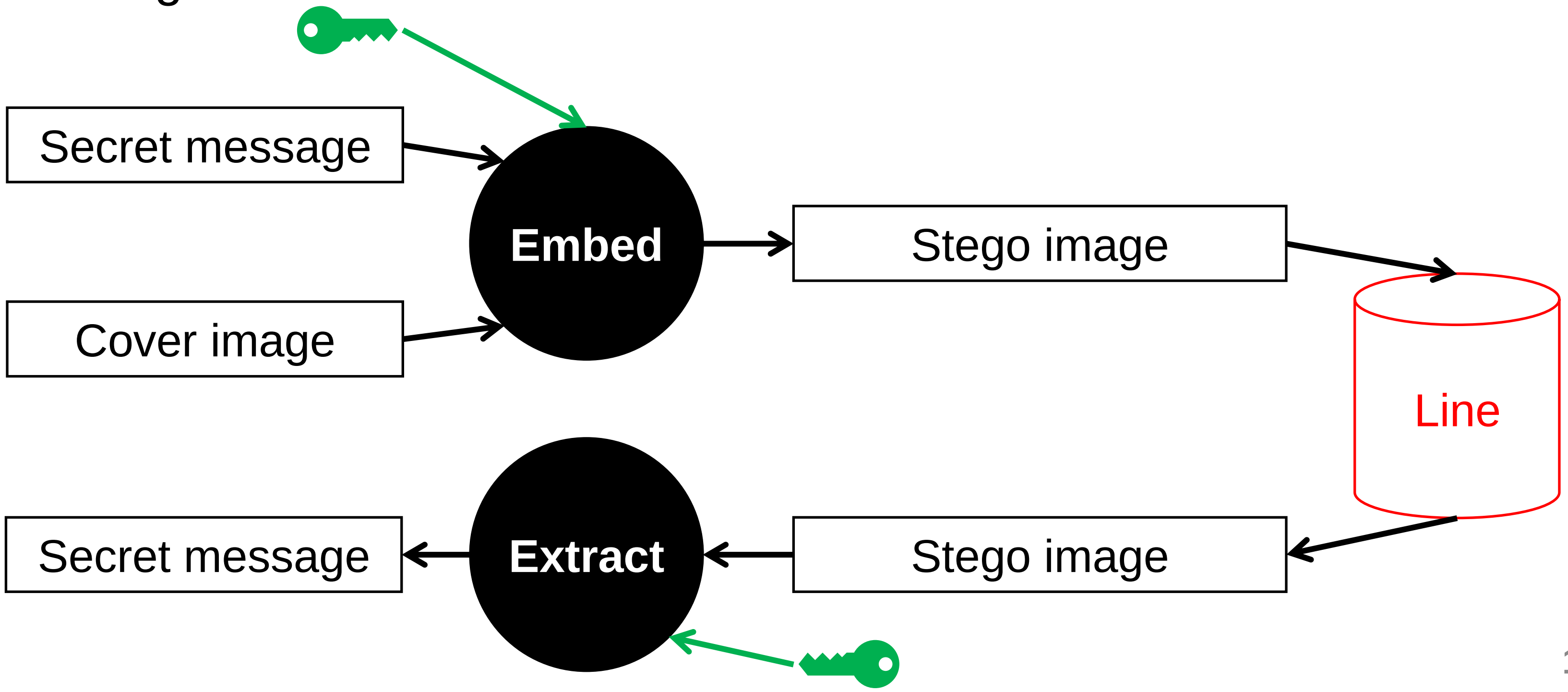
k = 8
MSE = 19024.0


Stego Image

Q: The 3rd party may know about the LSB method; how can it be difficult for 3rd parties to extract secret information even if they know about the LSB method?

A: Use extra keys (only Alice and Bob know), for example:

- Use encryption key: encrypt the secret then embed
- Use random seed key: embed secret bits into elements on cover image in random order of random seed



Review

Q: The cover photo is a grayscale image of size 100×100. To ensure invisibility we just want to use k = 1 bit LSB. If the cipher bit string is longer than 100×100, how can it still be embedded?

A: Compress the cipher bit string

This section

- More on the LSB method from the previous session (for nonpalette-based images)
- **Palette-based image introduction**
- Stagenography on palette-based image

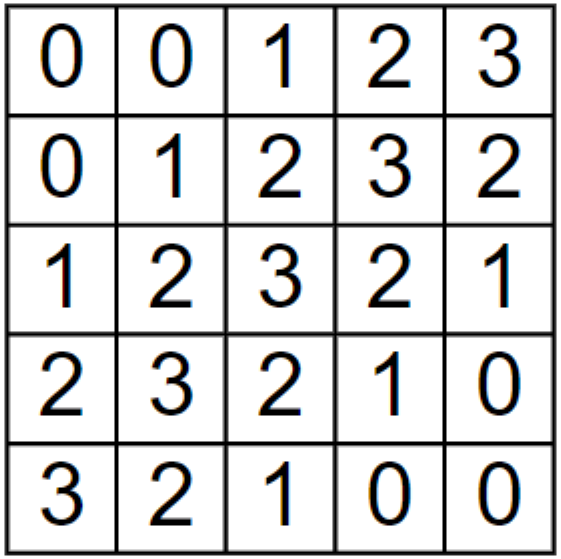
Palette-based image introduction

File **.bmp** (nonpalette), 768K

File **.gif** (palette), 276K

Palette-based image introduction

- A technique to manage digital images' colors in a limited fashion



0	0	1	2	3
0	1	2	3	2
1	2	3	2	1
2	3	2	1	0
3	2	1	0	0

Image Data

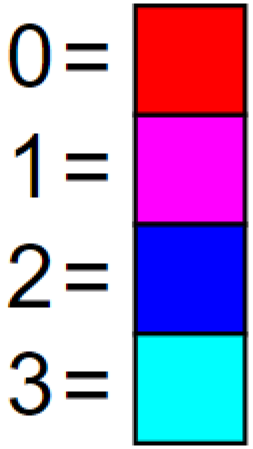
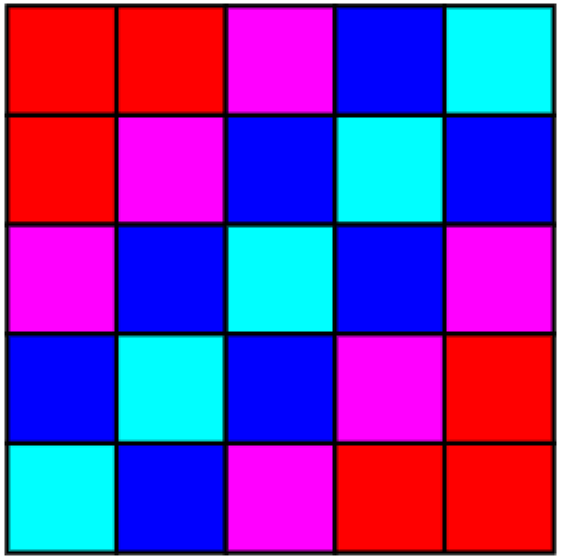


Image Palette



Display Image

Palette-based image introduction

- Palette images ***reduce the storage space*** required for images by:
  - Reducing the number of colors used in the image to (256 colors)
  - Using one number per pixel (one channel) to specify the color for each pixel.
  - Each color number corresponds to a color in a palette of 256 colors.
  - Each color in the palette is an RGB color out of a possible range of millions of colors.

Palette-based image introduction

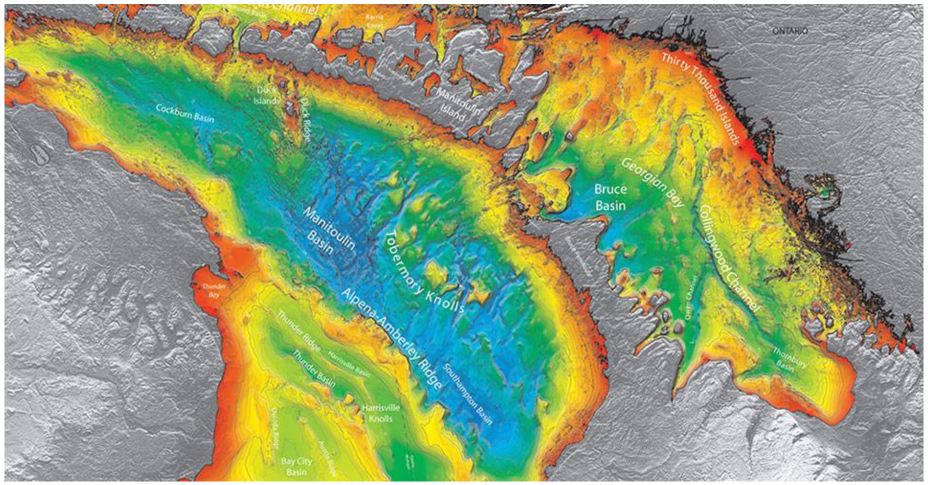
- **Palette images** are known as **Indexed Color** mode images in Adobe PhotoShop and *palettes* are called *color tables* or *color maps* in some applications.
- Using palette images to save storage space has become less important for images in modern times:
  - Cost of disk drive space has plummeted.
  - Preserving high visual quality is usually now more important with most images than saving space.

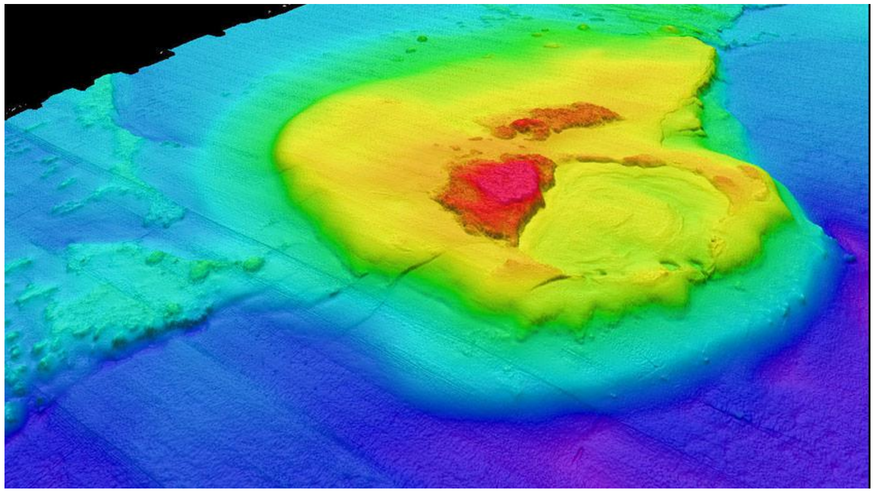
Palette-based image usage

- Single Channel Rasters: such rasters often represent terrain elevation or bathymetry.   Coloring pixels using a palette can provide more realistic terrain relief or a contouring effect.
- Classification - An important raster data usage of palettes is to force pixels into a pre-defined set of color values. Such values can represent various classification schemes for the physical regions or data values represented by the pixels.
- Web Images - Reducing the size of images is important when publishing images on Internet. Even with fast connections, Internet is so slow that it is still important to reduce the size of images used on web sites. Reducing the number of colors in an image allows web graphics formats the best possibilities of compressing the image to a small size, which in turn allows faster loading.

Palette-based image usage

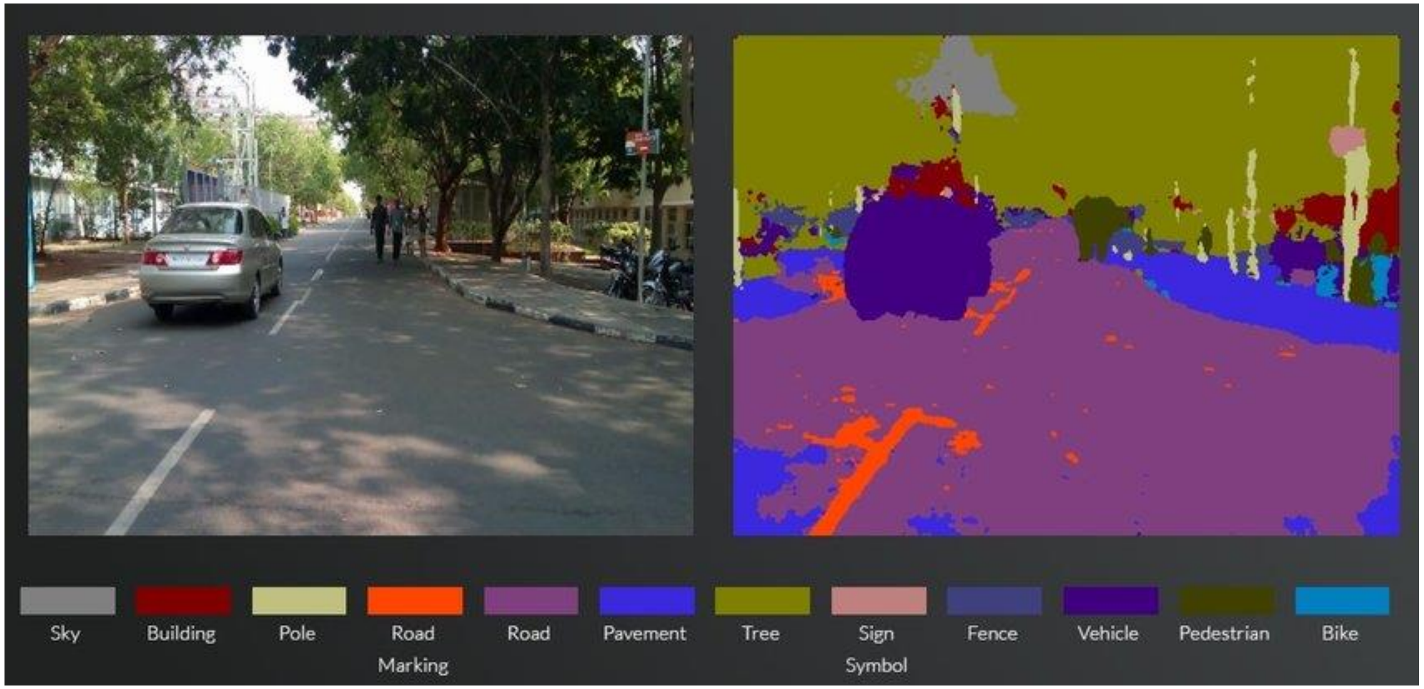
- **Single Channel Rasters**: such rasters often represent terrain elevation or bathymetry. Coloring pixels using a palette can provide more realistic terrain relief or a contouring effect.





Palette-based image usage

- **Classification:** an important raster data usage of palettes is to force pixels into a pre-defined set of color values. Such values can represent various classification schemes for the physical regions or data values represented by the pixels.

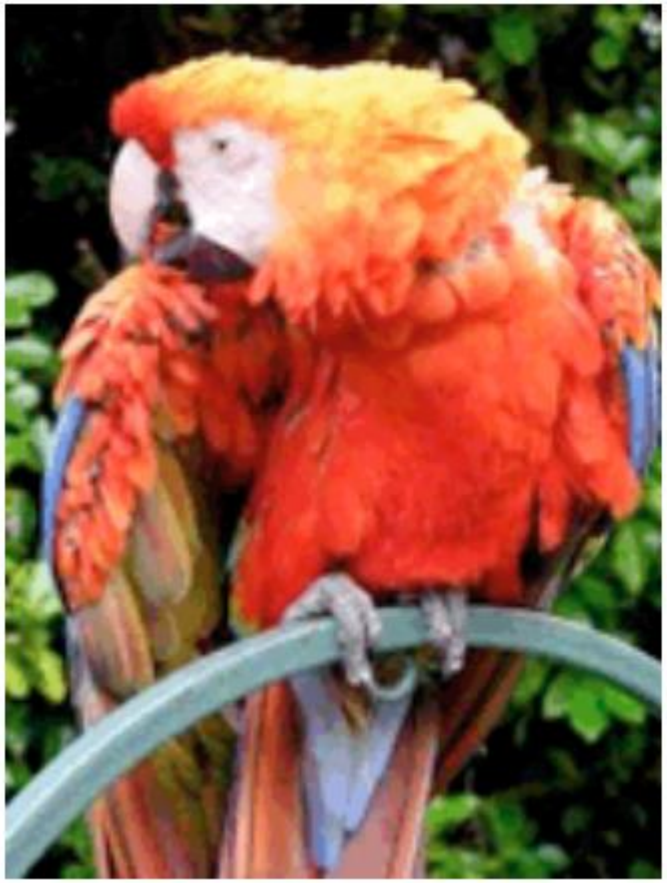
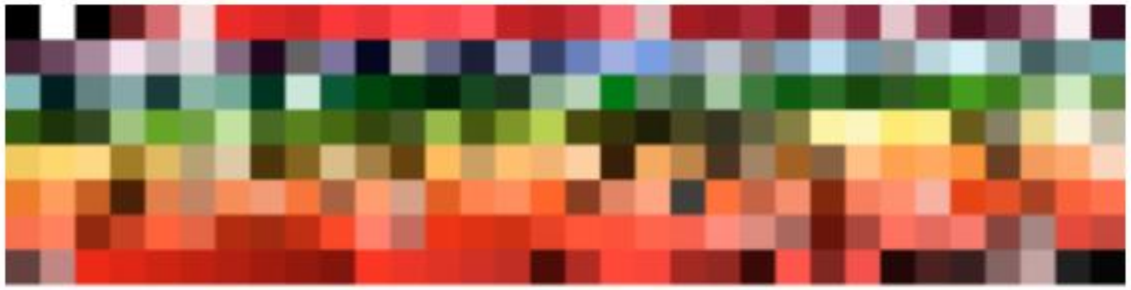


Palette-based image usage

- **Web Images**: reducing the size of images is important when publishing images on Internet. Even with fast connections, Internet is so slow that it is still important to reduce the size of images used on web sites.

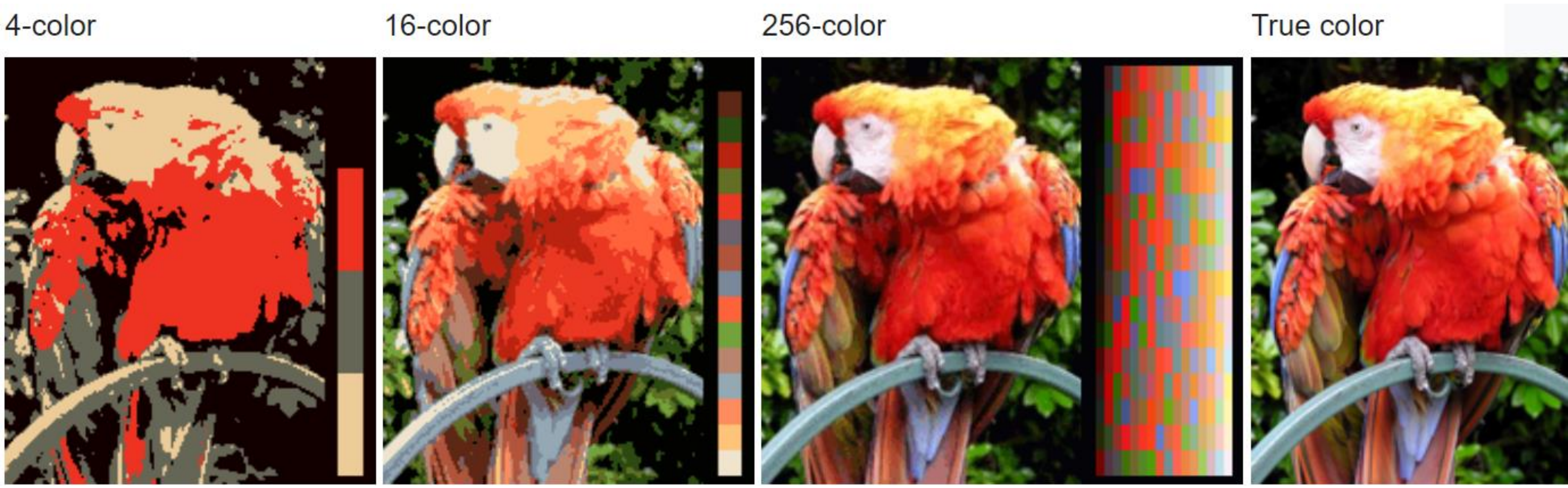
Example

- Indexed 256-color image

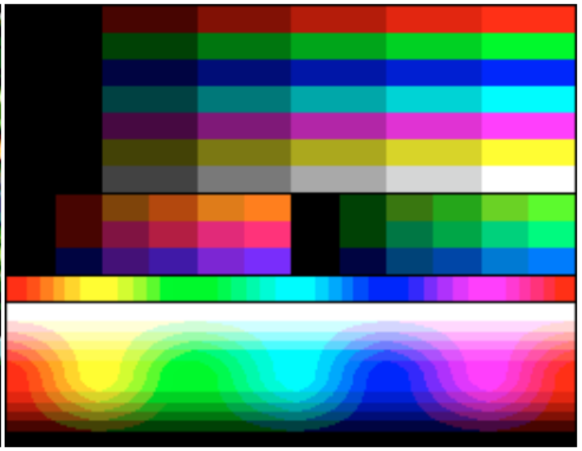
Palette-based image

- Advantages
  - Indexed color saves a lot of memory, storage space, and transmission time
- Disadvantages
  - The limited set of simultaneous colors per image



Master palette

- A unique, common master palette, which can be used to display with reasonable accuracy any kind of image.
- Do not need to store the palette in the file.
- More detail here.



Red	#000000	#330000	#660000	#990000	#CC0000	#FF0000
Green	#000000	#003300	#006600	#009900	#00CC00	#00FF00
Blue	#000000	#000033	#000066	#000099	#0000CC	#0000FF

Adaptive palette

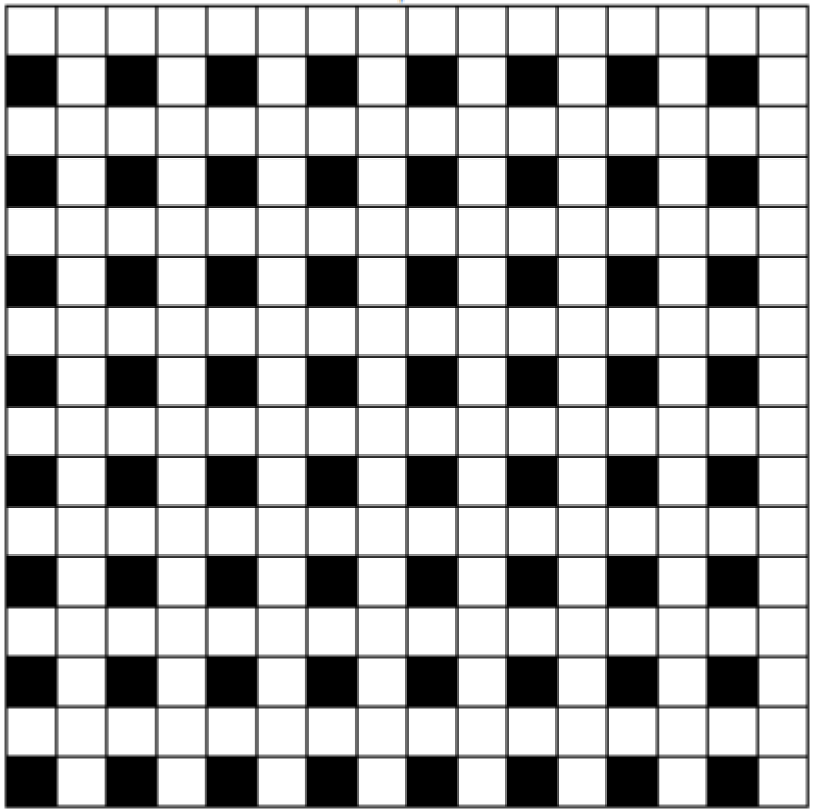
- Adaptive palettes: the colors are selected or quantized through some algorithm directly from the original image (by picking the most frequent colors).
- Combine further dithering, the indexed color image can nearly match the original.
- Need to store the palette in the file and load when display the image.

Dithering

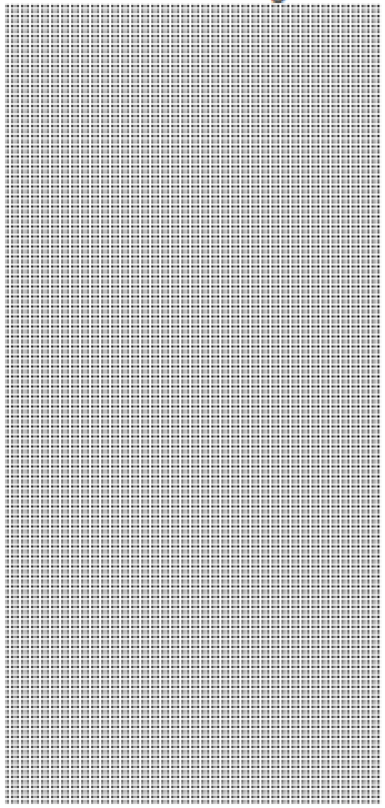
- Representing a color by combinations of dots of other colors.
- If the one exact shade of pink is not in the palette, then dots of other palette colors are mixed to simulate the color.
- Dithering often causes a visible and objectionable dotted or speckled image. But that simulated color may be much closer than a nearest color approximation.

Dithering

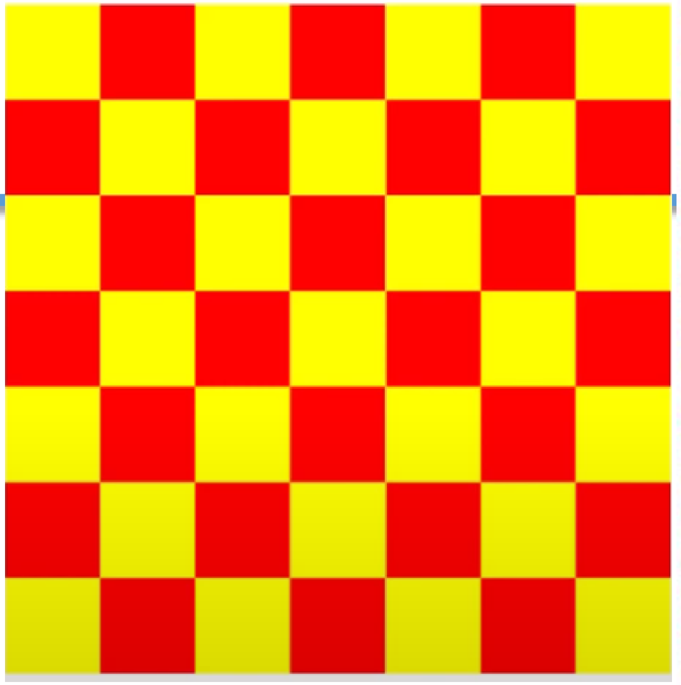
Zoomed In
25% Black, 75% White

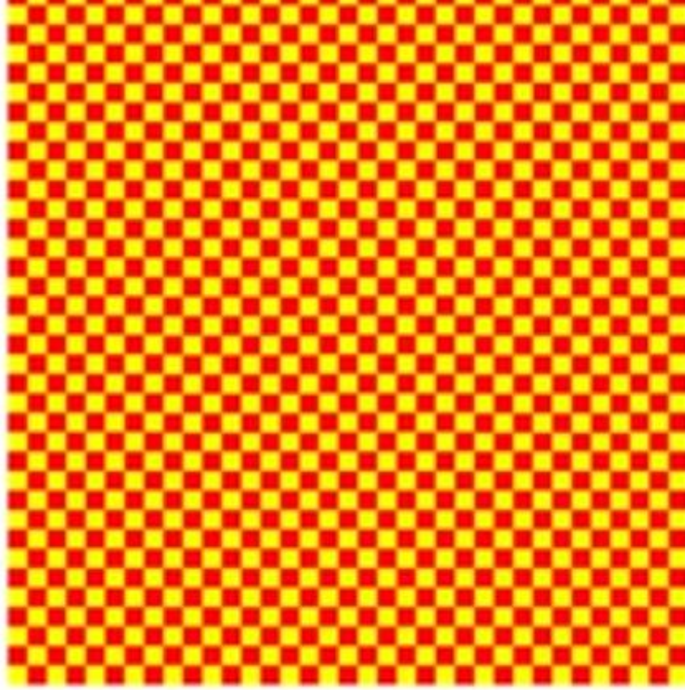


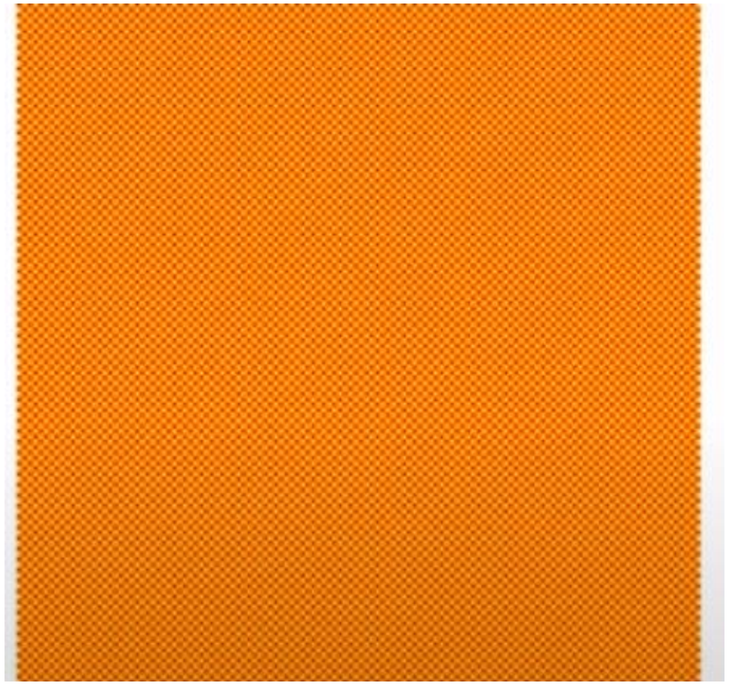
Actual Size
Dithered Image



Dithering

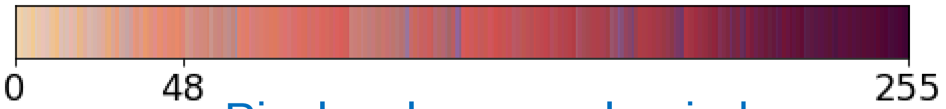






Palette-based Image

Color palette of 256 RGB colors selected from 256^3 RGB colors



Pixel value = color index (0-255) in palette

```
[[ 48  48  43 ...,  39  72 119]
 [ 48  48  43 ...,  39  72 119]
 [ 48  48  43 ...,  39  72 119]
 ...,
 [241 241 232 ..., 167 166 181]
 [241 241 227 ..., 157 157 155]
 [241 241 227 ..., 157 157 155]]
```

File **.gif** (palette, lossless), 276K

More about image properties

- Image properties :
  - Grayscale or RGB
  - Nonpalette-based or palette-based
- Others: lossless (no compression or lossless compression) or lossy (lossy compression)
- Several image formats :
  - BMP: lossless, palette-based or nonpalette-based
  - GIF: lossless, palette-based
  - PNG: lossless, palette-based or nonpalette-based
  - JPEG: lossy, nonpalette-based
  - ...
- Hide secret information on lossy image?

Hiding secret by LSB method on lossy images

File stego.**bmp** (lossless),
size: 768K

File stego.**jpg** (lossy),
size : 39K

Chuỗi rút trích được:
"I love u"

Chuỗi rút trích được:
"\xc4\xed\xb6\xdb\x13..."

More about image properties

- Image properties :
  - Grayscale or RGB
  - Nonpalette-based or palette-based
- Others: lossless (no compression or lossless compression) or lossy (lossy compression)
- Several image formats :
  - BMP: lossless, palette-based or nonpalette-based
  - GIF: lossless, palette-based
  - PNG: lossless, palette-based or nonpalette-based
  - JPEG: lossy, nonpalette-based
  - ...
- Hide secret information on lossy image?
  - Difficult , temporarily leave it for later, for now, only do it with lossless images

This section

- More on the LSB method from the previous session (for nonpalette-based images)
- Palette-based image introduction
- **Stagenography on palette-based image**

Steganography in palette images

- Palette images, such as GIF or the indexed form of PNG, represent the image data using pointers to a palette of colors stored in the header.
- It is possible to hide messages both in the **palette** and in the **image data**

Embedding in palette

- **How to embed**: reorder image pallete based on secret bit string.
    - Hide short messages as permutations of the palette
- ***Reordering the image palette*** and ***reindexing*** the image data correspondingly
- This method is implemented in the steganographic program Gifshuffle

Embedding in palette

- **How to embed**: reorder image pallete based on secret bit string.
    - Hide short messages as permutations of the palette
- ***Reordering the image palette*** and ***reindexing*** the image data correspondingly
- This method is implemented in the steganographic program Gifshuffle

Embedding in palette - Analyze

- Invisiblility?
  - ☺ Does not change the appearance of the image
  - Many image-editing programs order the palette according to luminance, frequency of occurrence, or some other scalar factor.
  - A randomly ordered palette will thus immediately raise suspicion
- Robustness?
  - Displaying the image and resaving it may erase the information because the palette may be reordered ☹
- Capacity?
  - ☹ Small. $\log_2(256!) \approx 1684$ bits ≈ 210 bytes

Embedding by sort palette

- Nonpalette-based image: pixel value = Color
  - Change the LSB bit of the pixel value, the color will change a little
  - The human eye is hard to see ☺
- Palette-based image: pixel value = index of color in pallette
  - Changing the LSB of the pixel value will change the color index slightly
  - Are you sure that two colors that have the same index in the palette will look almost the same?
    - Not sure ☹

Embedding by sort palette

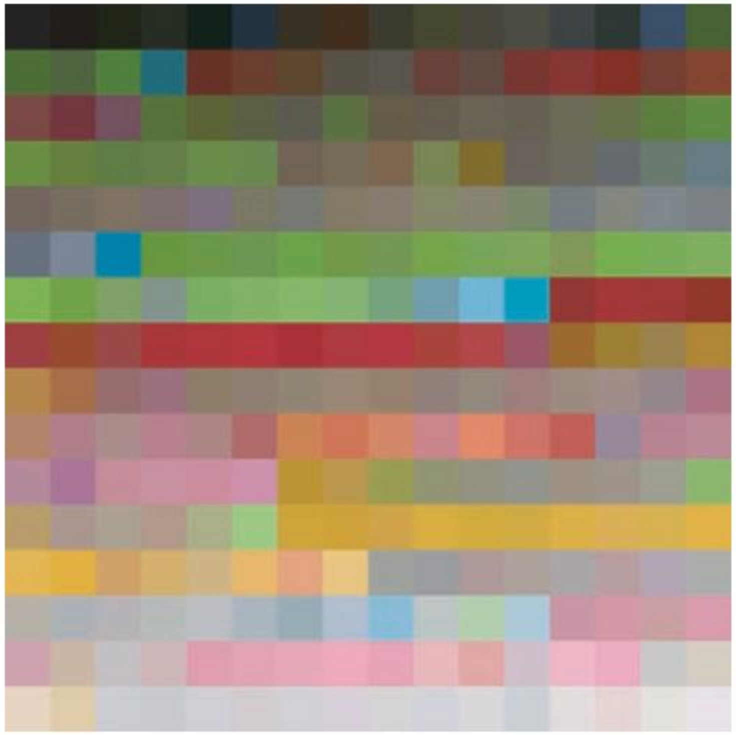
- Nonpalette-based image: pixel value = Color
  - Change the LSB bit of the pixel value, the color will change a little
  - The human eye is hard to see ☺
- Palette-based image: pixel value = index of color in pallette
  - Changing the LSB of the pixel value will change the color index slightly
  - Are you sure that two colors that have the same index in the palette will look almost the same?
    - Not sure ☹

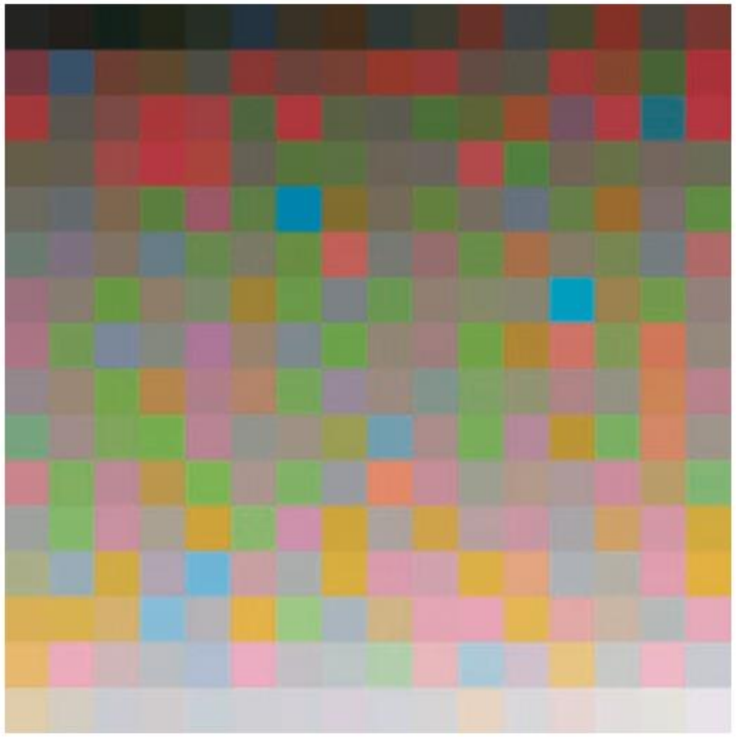
Embedding by sort palette

- One way is to **rearrange** the palette so that colors that look almost the same are next to each other, then apply LSB as usual
- With each color in the palette represented by 3 values (R, G, B), how to arrange the palette?
- One way: for each color, calculate the value $\sqrt{R^2 + G^2 + B^2}$, then sort the palette by this value
    - Two color (128, 0, 0) and (0, 128, 0) are completely different but have the same value according to the calculation above ☹

Embedding by sort palette

- EzStego orders the palette by luminance.
- Occasionally colors with similar luminance values may be relatively far from each other

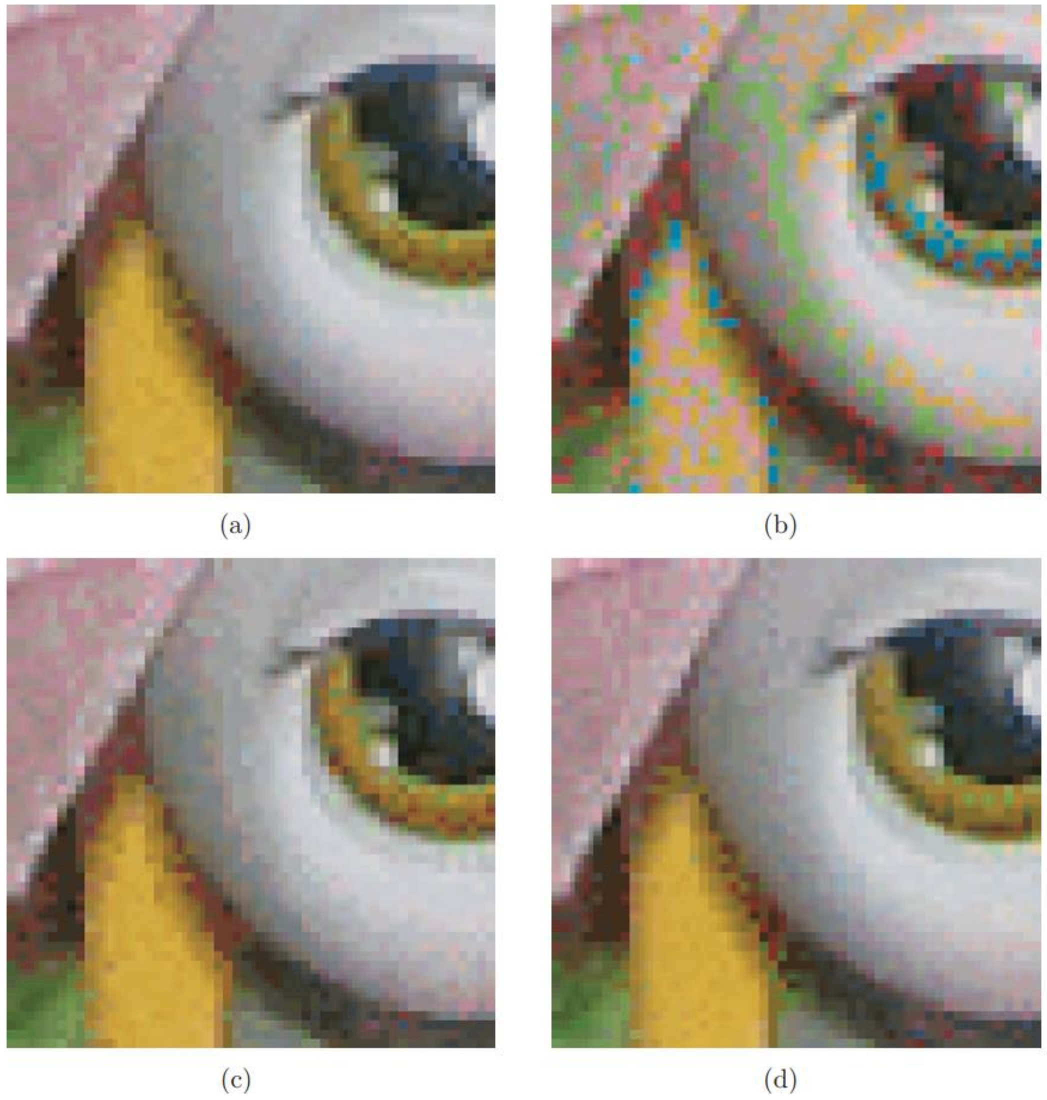

Original palette


Sorted Palette

Embedding by sort palette

Original Image

Embedding by sort palette



(a) Original, (b) embedding a maximum-length message using EzStego, (c) optimal-parity embedding, (d) and embedding while dithering

Optimal-parity embedding

Other way by Jiri Fridrich, rearranging the palette is:

- Embed:

  To embed a cipher bit in a pixel:

  - Find in the palette the closest color to the color of the pixel under consideration (according to the Euclidean distance) and have (R+G+B) % 2 matches the secret bit
  - Change the pixel value to the index of the color just found
- Extract?
  - For a pixel, we get the corresponding color in the palette, and secret bit = (R+G+B) % 2 with (R, G, B) is the value of this color

Embedding while dithering

- The cover image is a true-color 24-bit image, for example in the BMP format, and we wish to save it as a GIF.
- This conversion involves: color quantization and dithering, which spreads the quantization error.
- To embed ***m*** bits:
  - ***m*** pixels are pseudo-randomly selected to carry the message bits.
  - Then, color quantization and dithering are performed as usual by scanning the image by rows with one exception.
  - At each message-carrying pixel, its color is quantized to the closest palette color with the parity that matches the message bit.
- The combined quantization and embedding are thus diffused to neighboring pixels. This way, both the quantization error and the error due to message embedding will be diffused through the whole image