Computer Graphics

Lecture 4

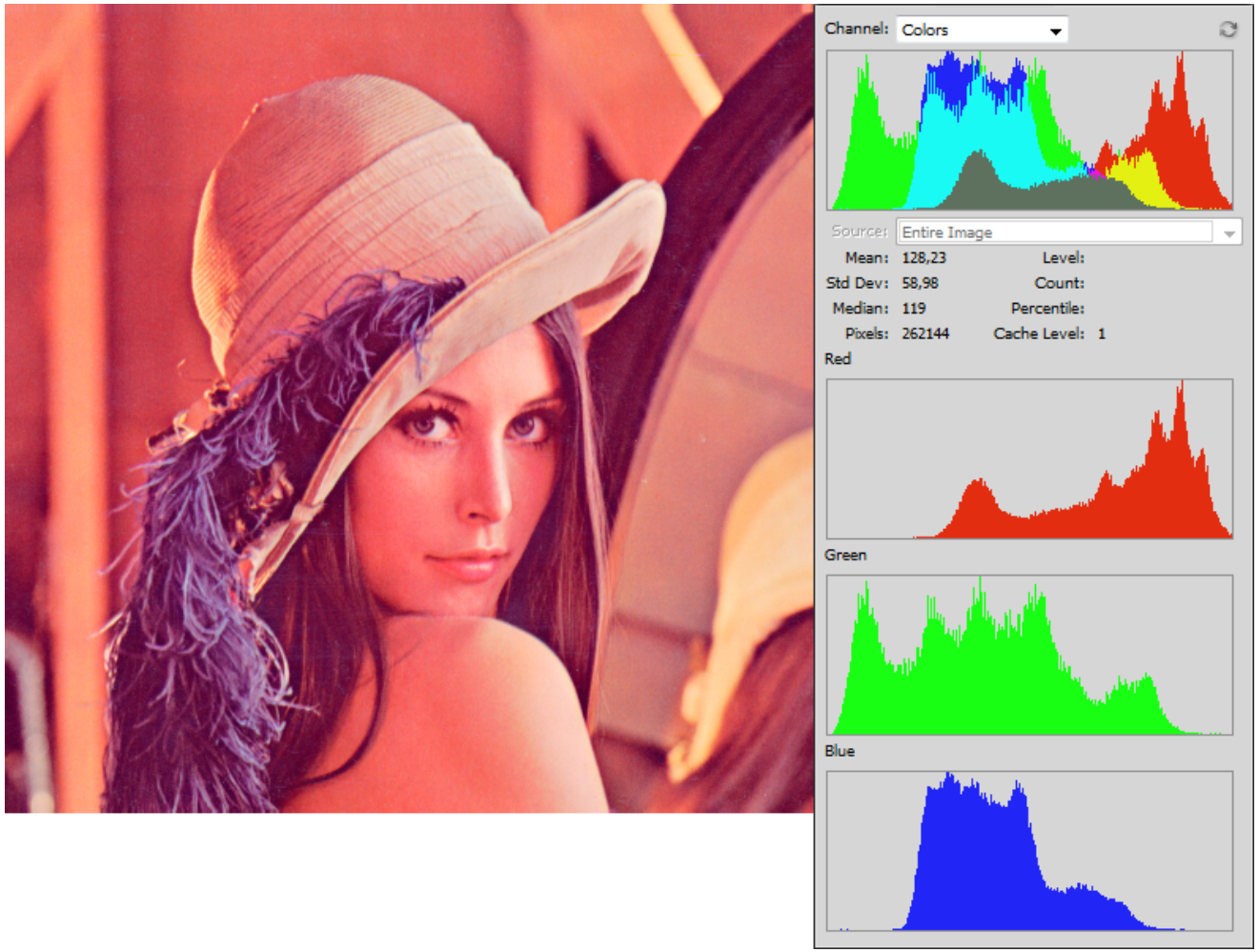
Paweł Aszklar
P.Aszklar@mini.pw.edu.pl

Faculty of Mathematics and Information Science
Warsaw University of Technology

Warsaw 2013

Histogram

- Graphical representation of tonal distribution
- Plots the number of pixels in each tonal value



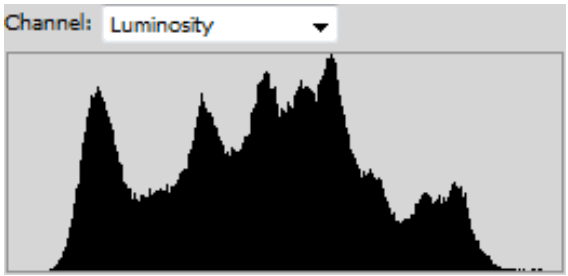
Histogram Stretching

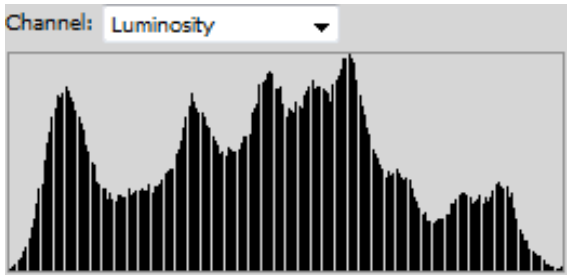
- Equivalent as contrast enhancement filter
- All channels stretched equally or each channel stretched independently
- Image transformed to use the full range of available intensity values

$$I(x,y) = \frac{255}{I_{\max} - I_{\min}} \left(I(x,y) - I_{\min} \right)$$

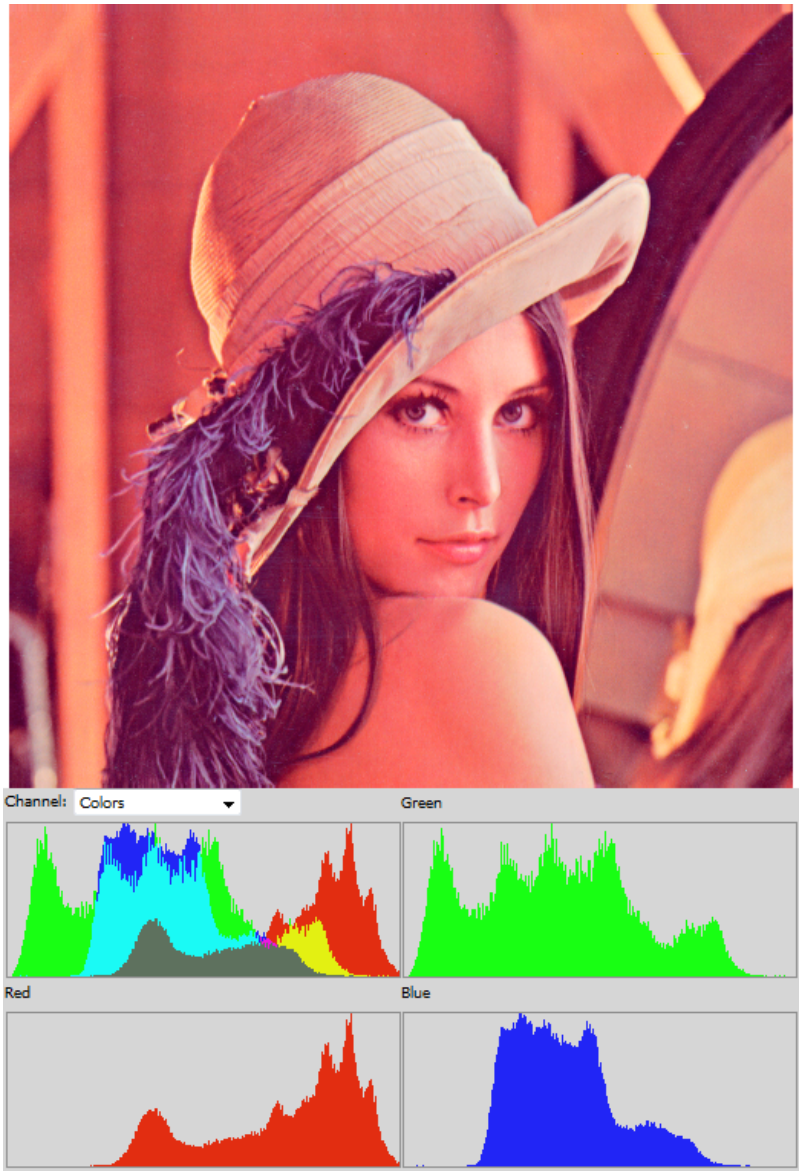
- When determining values of $I_{\min}$ and $I_{\max}$ we can ignore intensities for which the histogram value is lower than a certain threshold

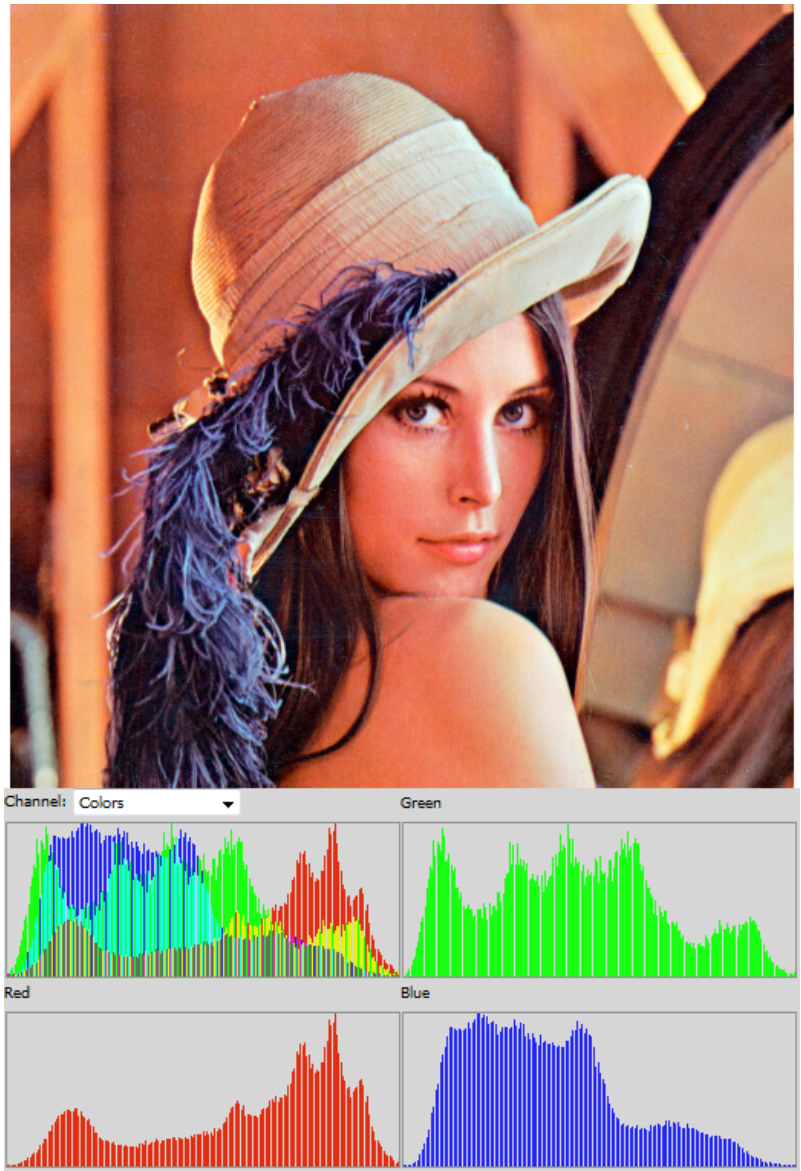
Histogram Stretching





Histogram Stretching





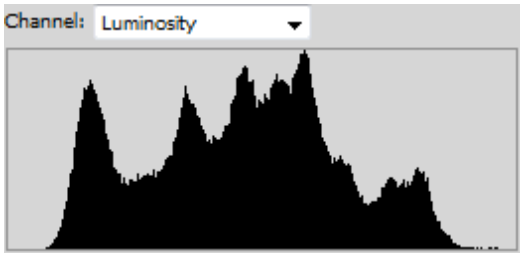
Histogram Equalization

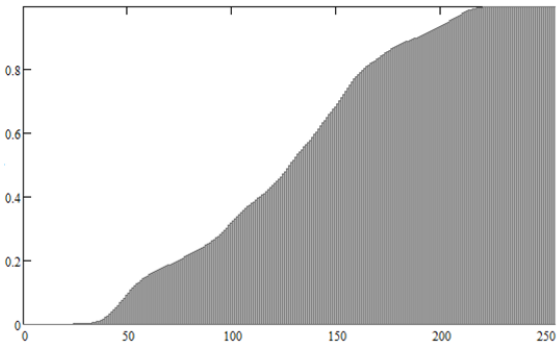
- The goal is to get more uniform distribution of intensities across the available range
- Effectively spreads out most frequent intensity values on the histogram
- Increases the contrast of lower local contrast
- Improves the linearity of the cumulative distribution function

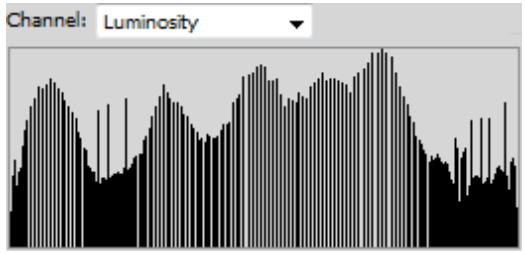
$$D(i) = \frac{\sum h(i)}{n_{\text{pixels}}}$$

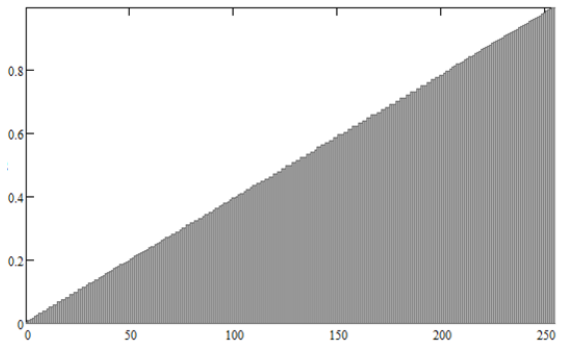
$$I(i) = \frac{D(i) - D_0}{1 - D_0}(K - 1)$$

Histogram Equalization









Histogram Equalization

Color Quantization

- Process of representing an image with a limited number of colors
- The set of colors used is called a *color palette*
- Applications:
  - Displaying images on (legacy or industrial) hardware supporting restricted number of colors (often due to memory and processing power limitations rather than display capabilities)
  - Conversion to file formats supporting color palettes
  - Image and video compression
- First step is to select the color palette
- Second step is to replace each pixel with a color from the palette
  - Usually nearest in palette neighbor search (any norm, e.g.: Euclidean norm, Manhattan norm, etc.) in any color space (usually RGB)
  - Euclidean distance in CIELAB color space often gives the best results

Color Quantization Methods

Fixed Quantization

- A predefined color palette (imposed by hardware or chosen arbitrarily) and fixed color mapping
- Potential artifacts: color shift, false contour, etc.
- E.g.: Uniform quantization

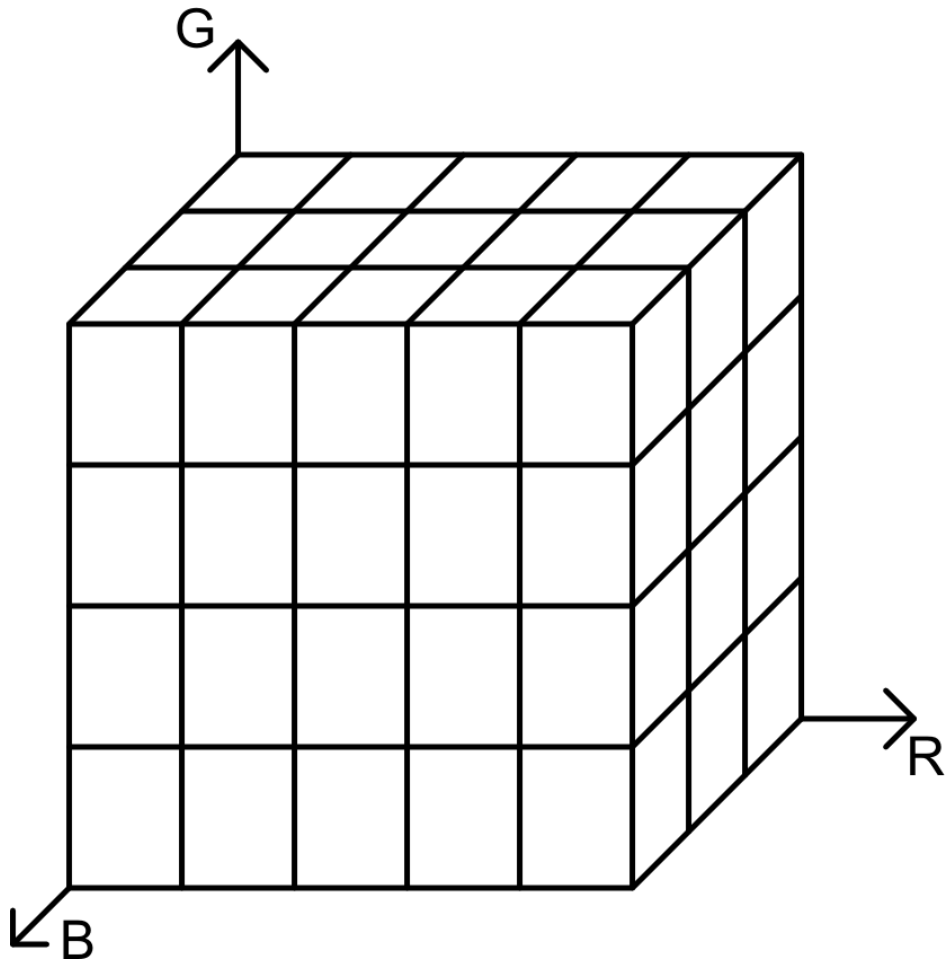
Color Quantization Methods

Adaptive Quantization

- Selects image-dependent color palette
- Better quality, but more time consuming
- Any three-dimensional clustering algorithm will work (octree, popularity algorithm, median-cut, k-means)
- Specialized algorithms:
  - Spatial Color Quantization `http://www.cs.berkeley.edu/~dcoetzee/downloads/scolorq/`
  - NeuQuant Algorithm `http://members.ozemail.com.au/~dekker/NEUQUANT.HTML`

Uniform Quantization

- Arbitrary palette, image-independent
- Uniform subdivision of RGB cube for each coordinate
- Number of colors: $k_R \times k_G \times k_B$
- Palette colors — centers of cuboids



Popularity Algorithm

- Adaptive method
- Palette consists of the K most frequently occurring colors
  - To improve performance only check colors at local maxima of the histograms
  - Alternatively perform preliminary uniform subdivision (like uniform quantization) and search for cuboids with the most points in them
- Data structures:
  - Linked list — very high time complexity
  - Array indexed with colors — very high space complexity
  - BST, Octree

Octree Color Quantization

- Building Octree of colors
  - Colors of pixels are sequentially added to the leaves of the octree
  - If no leaf is found for a color, a new leaf is inserted at certain depth (usually depth of 8 for 8-bit per channel RGB)
  - Leaves store the count of pixels added to them and the sum of their colors
  - If after an insertion the number of leaves exceeds the size of the palette, reduce the tree until it doesn't

```
void insert(OctreeNode** node, Color c, int depth)
{
    if (*node == null)
        CreateAndInit(node, depth);
    if (isLeaf(*node)) {
        (*node)->pixels_count++;
        addColors(*node, c);
    } else
        insert(&(*node)->child[Branch(c, depth)], c, depth + 1);
}
```

Octree Color Quantization — Tree Reduction

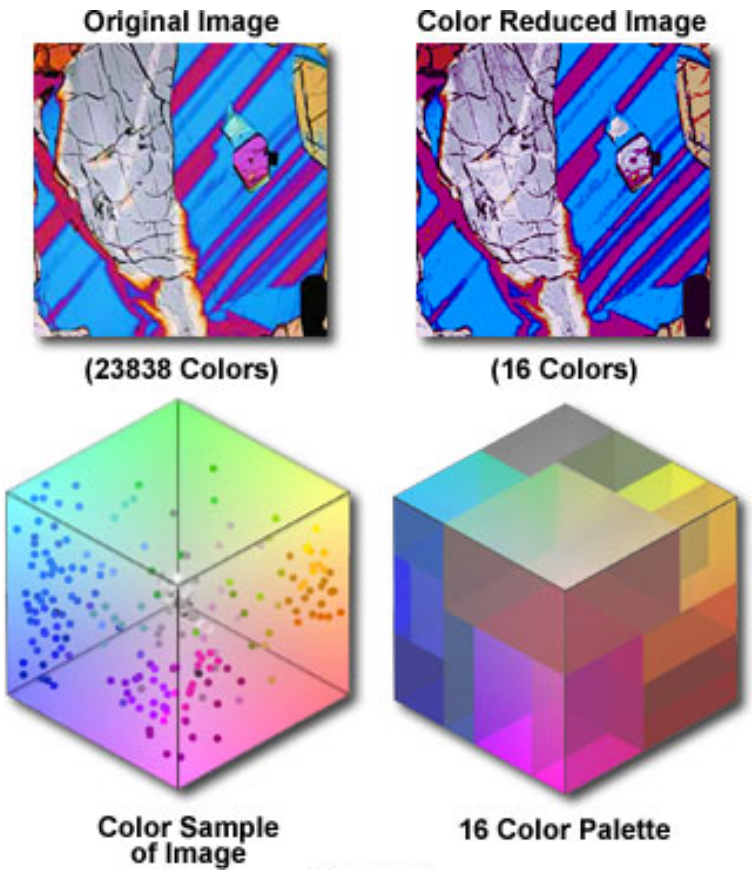
- Octree reduction is performed if number of leaves exceeds the available size of the color palette
- Reduction criteria
  - Select a non-leaf node with the highest depth value
  - If more than one exists, chose the one representing the fewest pixels (or the most, or any)
  - Make that node a leaf (number of pixels and the sum of colors is the total of values from children)
- Reduction may need to be repeated if selected node had only one child

Octree Color Quantization — Color Mapping

- After the tree has been built, for each leaf select a color in the final palette (average of colors added to that leaf)
- Color mapping — for each pixel traverse the final tree again to find a corresponding leaf. Assign to that pixel a color from the palette represented by that leaf

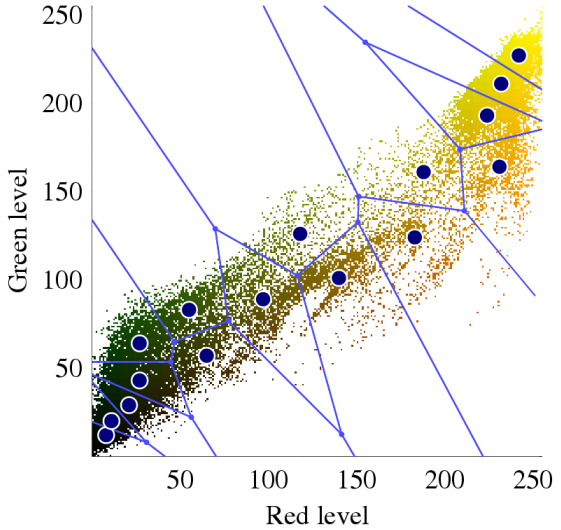
Median-Cut Algorithm

- Subdivide the color space (e.g. the RGB cube) into K cuboids in such a way that each of them contains (approximately) the same number of pixels
- Each cuboid represents a color in the palette (either center of the cuboid or average of pixels)



K-Means Algorithm

- Well-known statistical algorithm
- Iterative
- Select K random colors — *centroids* (K — size of the color palette)
- Assign each pixel to its closest centroid
- For each group calculate the center of gravity (average) which becomes a new centroid (replacing the old one)
- Reassign the pixels to centroids and repeat until no changes occur



Color Quantization and Dithering

- Once the color palette is established, we can always use some metric (e.g. Euclidean distance) to select a color from the palette for each pixel
- This makes it easy to incorporate dithering into the color selection (especially using any of the error propagation methods)

Color Quantization and Dithering

Example:

- Color palette size $= 8$
- 2 levels per channel

Original image:

Octree quantization:

Error diffusion:

Color Quantization and Dithering

Example:

- Color palette size = 256
- Levels per channel — 6R/7G/6B

Original image:

Octree quantization:

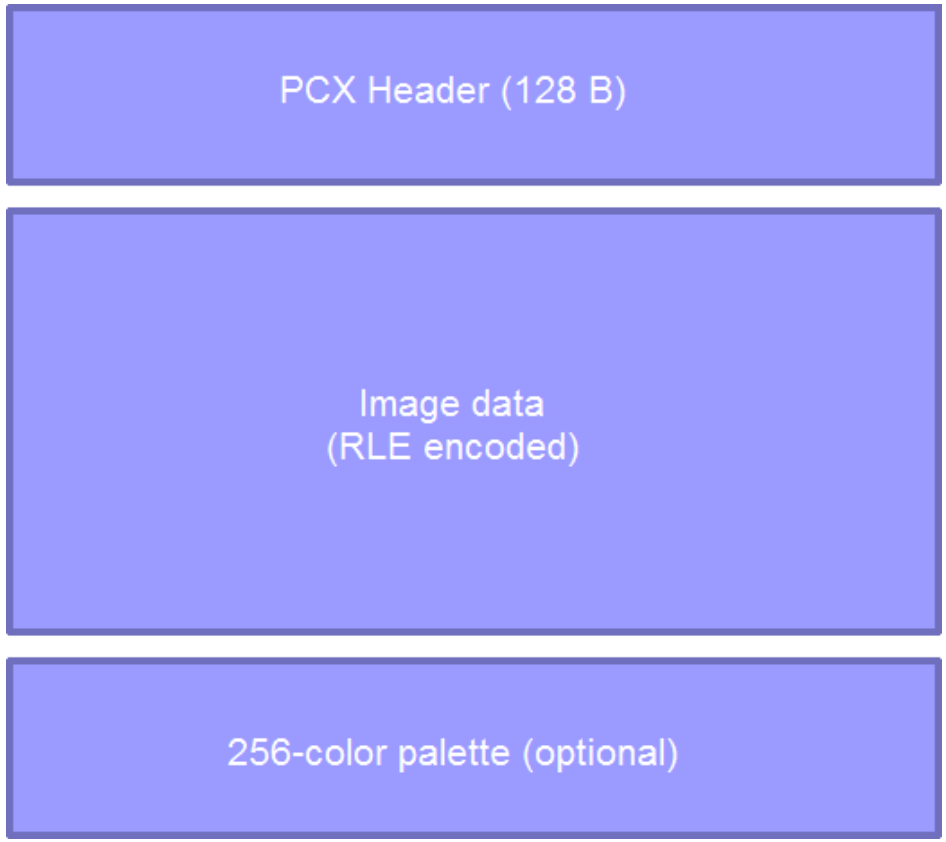
Error diffusion:

Popular Raster Image File Formats

- PCX (legacy)
- BMP
- GIF
- PNG
- JPEG

PCX

- **P**ersonal **C**omputer e**X**change
- Developed by ZSoft company
- Simple, fast, lossless compression
- Designed for 16 (stored in header) or 256 (appended at the end) color palette, extended to true-color (24-bit) images
- Image data compressed using run-length encoding algorithm (RLE)



PCX — RLE Compression

- Bytes represent either
  - Value of the pixel (or palette index)
  - Beginning of a pair. First byte is the count, second is the value
- First two bits set indicate a pair, for example $(240, 167)$ indicate that value 167 should be repeated $240 - 191 = 51$ times.
- Some values (greater than 190) need to be encoded as a pair even if they are not repeating
- Allows for 0 run-length data

Bitmap (BMP)

- File structure
  - File header
  - Bitmap information header
  - Optional palette
  - Image data
  - Some versions allow to embed ICC color profile
- Huge file of uncompressed images
- Lossless — no quality loss during multiple edits, however other lossless formats are preferred nowadays
- Color components stored in reverse order: $[B, G, R]$
- Row of data always stored as a multiple of 4 bytes

Bitmap Header

```
typedef struct {
    unsigned short int type;                /* Signature (usually BM)                   */
    unsigned int size;                      /* File size in bytes                       */
    unsigned short int reserved1, reserved2;
    unsigned int offset;                    /* Offset to image data in bytes            */
} HEADER;

typedef struct {
    unsigned int size;                      /* Header size in bytes                     */
    int width, height;                      /* Image width and height                   */
    unsigned short int planes;              /* Number of color planes (usually 1)       */
    unsigned short int bits;                /* Bits per pixel                           */
    unsigned int compression;               /* Compression type                         */
    unsigned int imagesize;                 /* Image size in bytes                      */
    int xresolution, yresolution;           /* Pixels per meter                         */
    unsigned int ncolours;                  /* Number of colors in palette              */
    unsigned int importantcolours;          /* Number of significant colors in palette */
} INFOHEADER;
```

GIF

- **G**raphics **I**nterchange **F**ormat
- Developed in 1987 by CompuServe company
- Uses 256 global color palette
- Allows to store multiple images in blocks
- Each block can have it's own local 256-color palette
- Allows for extension blocks supporting animations, selecting one transparent color, interlacing etc.
- Uses LZW (Lempel, Ziv, Welch, '84) compression — it was subject of patent claims by Unisys company, but as of 2004 those patents expired.
- LZW is a lossless dictionary compression algorithm
- Good for graphics, logos, etc. Poor for photographs (difficult to represent true-color images)

GIF - LZW compression

Encoding:

1. Initialize dictionary with data alphabet
2. q = first input symbol
3. While input not empty
   1. read symbol s
   2. if $q{+}s$ in dictionary, $q{=}q{+}s$
   3. otherwise
      - output dictionary code for q
      - add $q{+}s$ to dictionary
      - $q{=}s$
4. output dictionary code for q

Decoding:

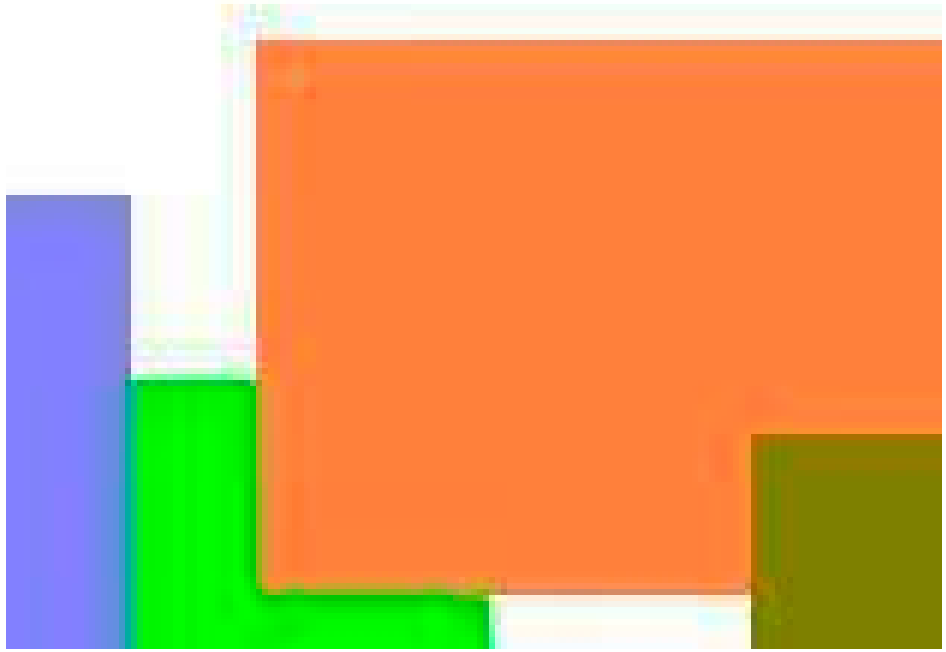
1. Initialize dictionary with data alphabet
2. pc = first input code
3. Output dictionary entry for pc
4. While input not empty
   - read code c
   - pq=dictionary[pc]
   - if code c in dictionary, add (pq+first symbol of dictionary[c]) to dictionary and output dictionary[c]
   - otherwise add (pq+first symbol of pq) to dictionary (special case)
   - $pc{=}c$

PNG

- **P**ortable **N**etwork **G**raphics
- Developed as an alternative to GIF when it was still subject to patents
- Two step compression algorithm
  - pre-compression filtering (usually improves compression)
  - dictionary compression using DEFLATE algorithm (modified LZ77)
- Supports grayscale aand RGB images (up to 16-bit per channel; both w/ and w/o alpha or with single transparent color) and color palettes (with optional transparency settings for colors in palette)
- Improved interlacing (over GIF), no animations
- Allows for extensions specifying absolute color space and white point, gamma correction, histograms, color profiles, pixel aspect ratio, stereoscopy and many more features

JPEG

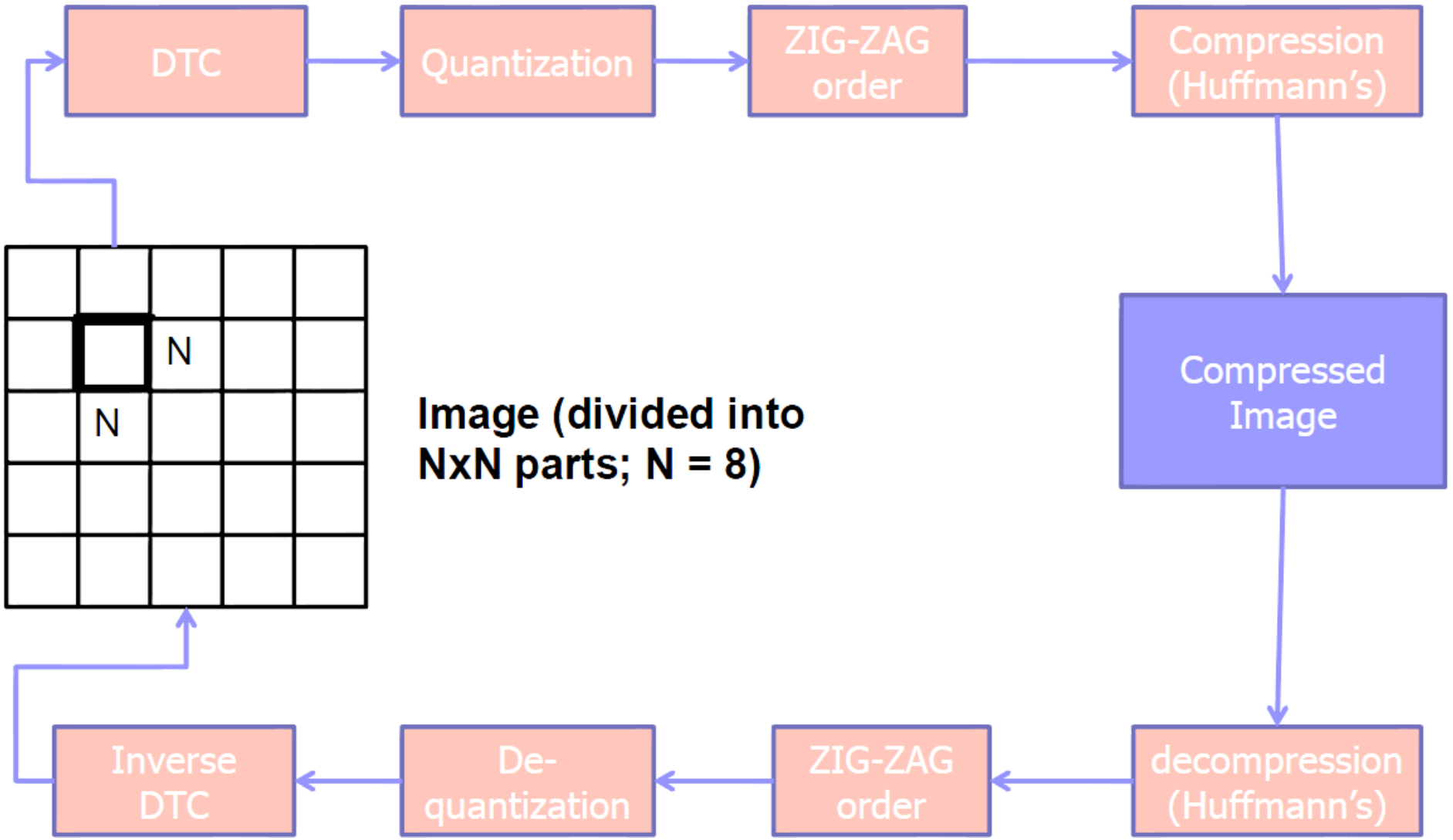
- Lossy compression method for digital photography created by **J**oint **P**hotography **E**xperts **G**roup
- Very good compression with acceptable quality loss for photography. Poor performance with more regular graphics, logos, etc.



JPEG Compression Algorithm

1. Image data converted into luminance and two chromaticity components (Y'CrCb)
2. Image channels is divided into $N \times N$ blocks
3. Discrete Cosine Transform (DCT) is applied to every block — Instead of pixel values we receive value means and frequencies of value variations within the blocks. No data loss here.
4. Data is subject to quantization — floating-point values are replaced with integers. **Here is where the data loss takes place.**
5. Zig-zag order is applied to DTC coefficients, so 0 values lie close together.
6. Non-zero coefficients are encoded (losslessly) using Huffmann's coding

JPEG Compression/Decompression



End of lecture 4