DCGI

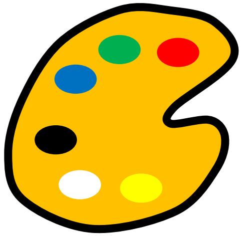
KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

APG – Dithering

JIŘÍ ŽÁRA

Dithering

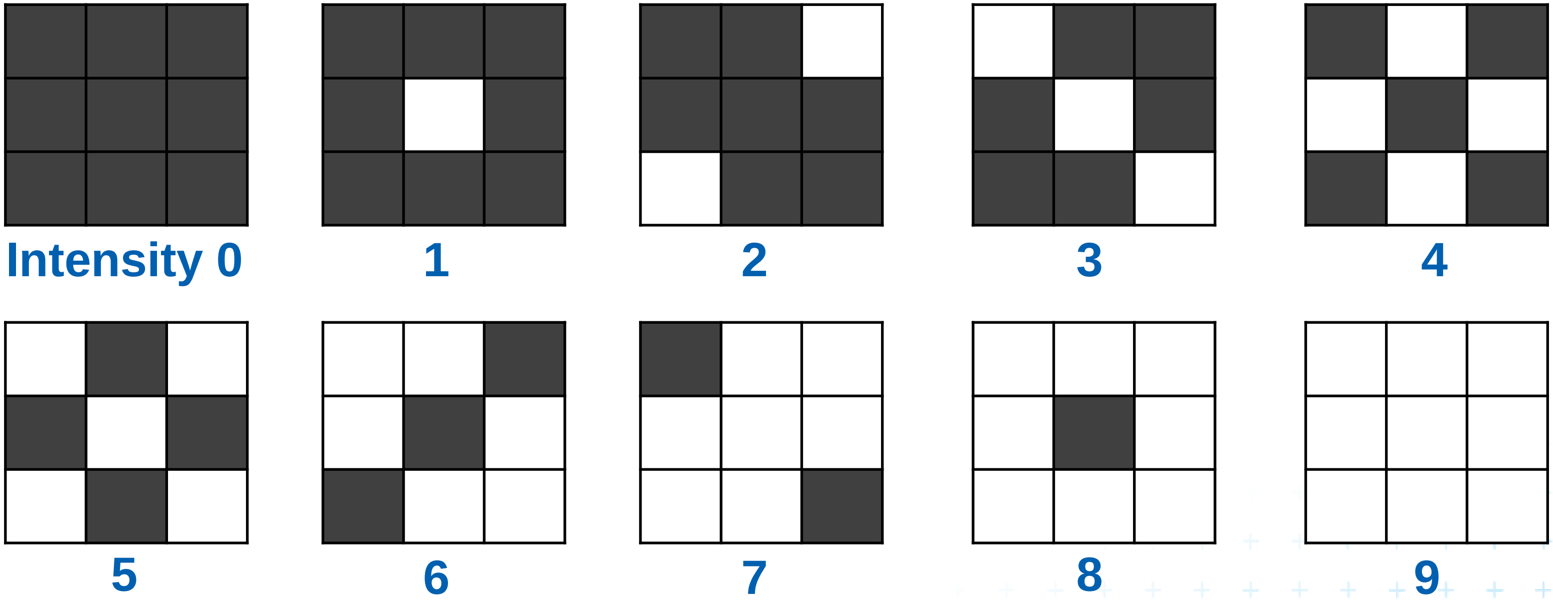
- Problem:
  - To display **many** colors on device with **limited number** of colors (with a small palette)



- Solution:
  - Color composition in human eye
    Near pixels are perceived as one (new, composed) color
- Two principal techniques:
  - **Halftoning** (polotónování) - larger image, N:1
  - **Dithering** (rozptylování) – same size, 1:1

Case: gray image on B/W device (printer)

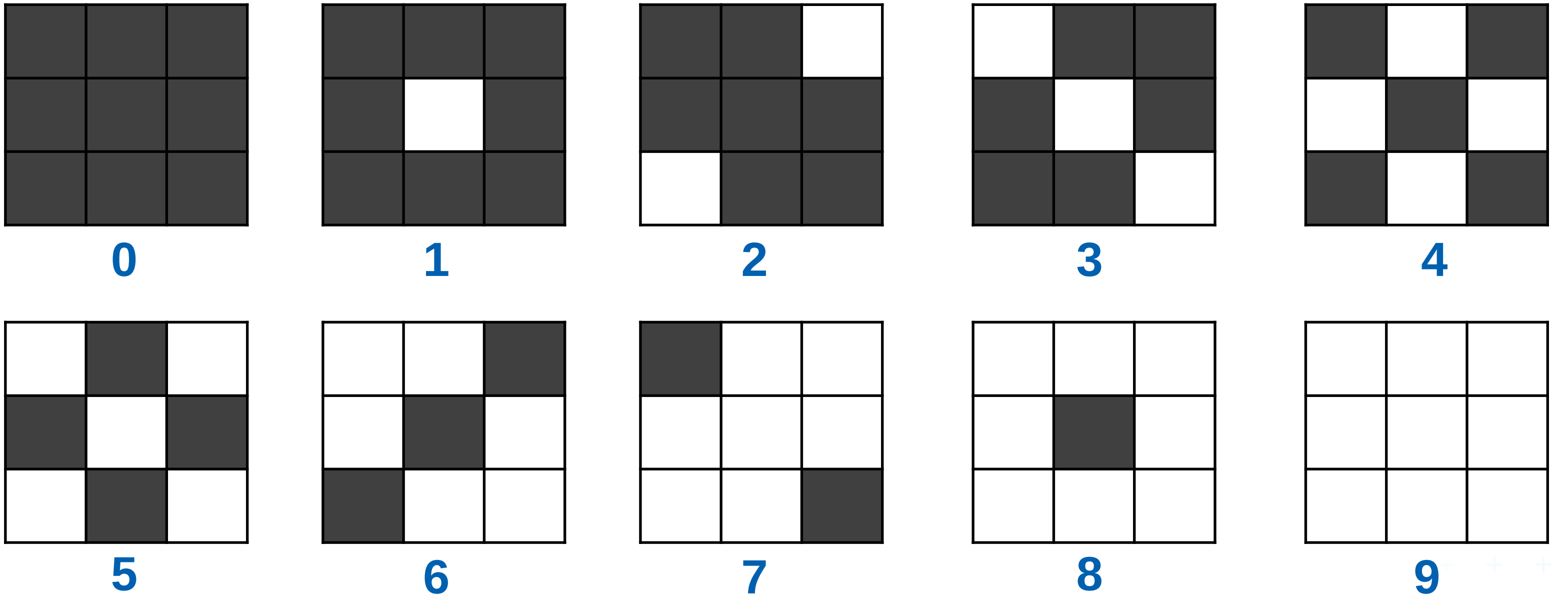
1 gray (input) pixel replaced by B/W (output) pattern



Horizontal and/or vertical lines?

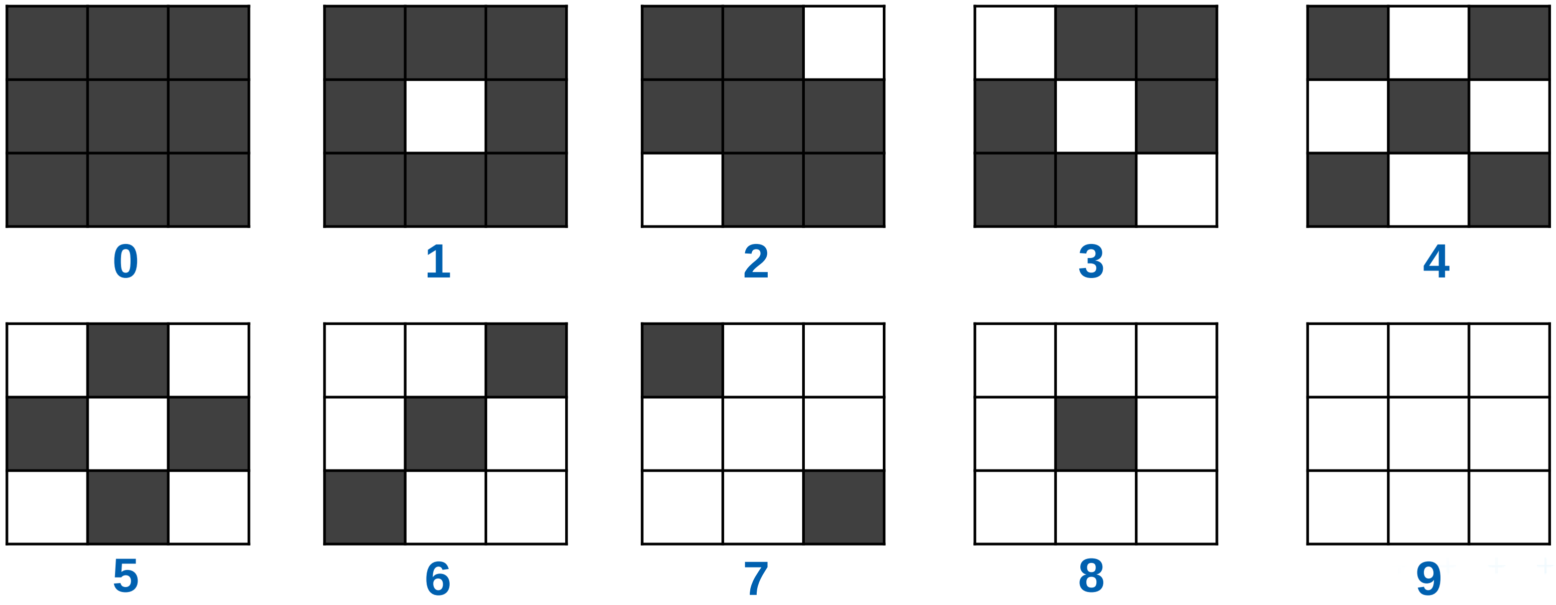
No! Oblique ones are better!

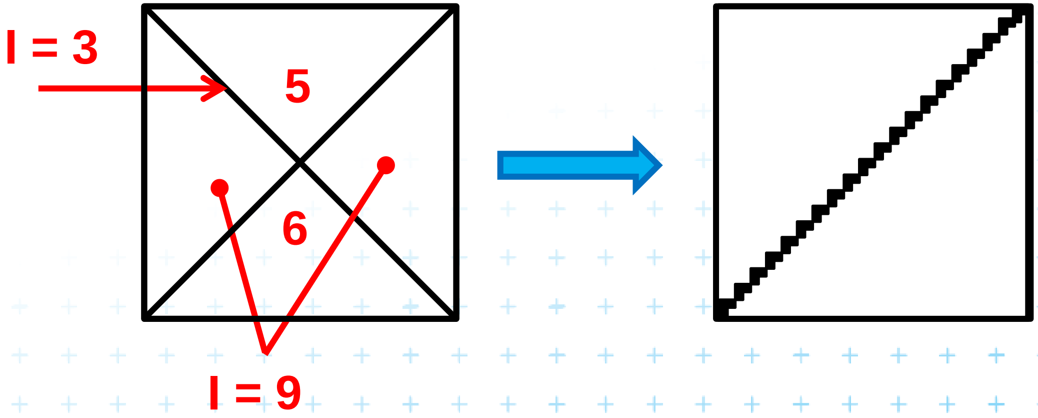
Image 1:1 (=dithering)



$$I_{OUT} = \mathrm{PATTERN}_{INP}[x \; \mathit{mod} \; 3, \, y \; \mathit{mod} \; 3]$$

Artefacts (due to large changes in patterns)





Regularly generated patterns

- **Dithering matrix**

$$M = \begin{bmatrix} 0 & 7 & 3 \\ 5 & 4 & 6 \\ 2 & 8 & 1 \end{bmatrix}$$

$$I_{INP} \in \langle 0,9 \rangle$$

$$I_{OUT} = (I_{INP} > M[\boldsymbol{x}\ mod\ 3, \boldsymbol{y}\ mod\ 3]) \in \langle 0,1 \rangle$$

„Cross-shape “ algorithm (for displays)

- N = power of 2

$$M = \begin{bmatrix} 0 & 3 \\ 2 & 1 \end{bmatrix} \Rightarrow M = \begin{bmatrix} Matrix_0 & Matrix_3 \\ Matrix_2 & Matrix_1 \end{bmatrix}$$

N = 2

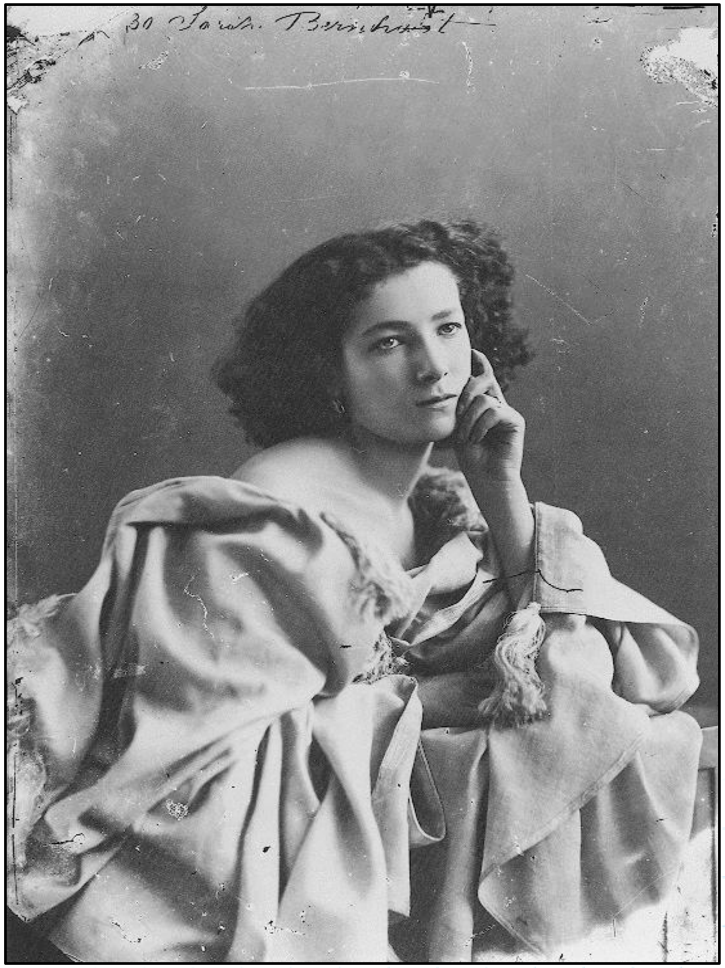
$$\boxed{\boldsymbol{Matrix_i = 4 * M + i}}$$

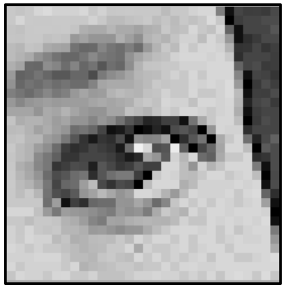
$$M = \left[\begin{array}{cc|cc} 0 & 12 & 3 & 15 \\ 8 & 4 & 11 & 7 \\ \hline 2 & 14 & 1 & 13 \\ 10 & 6 & 9 & 5 \end{array}\right] \dots$$

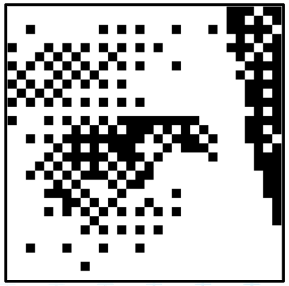
N = 4

Example

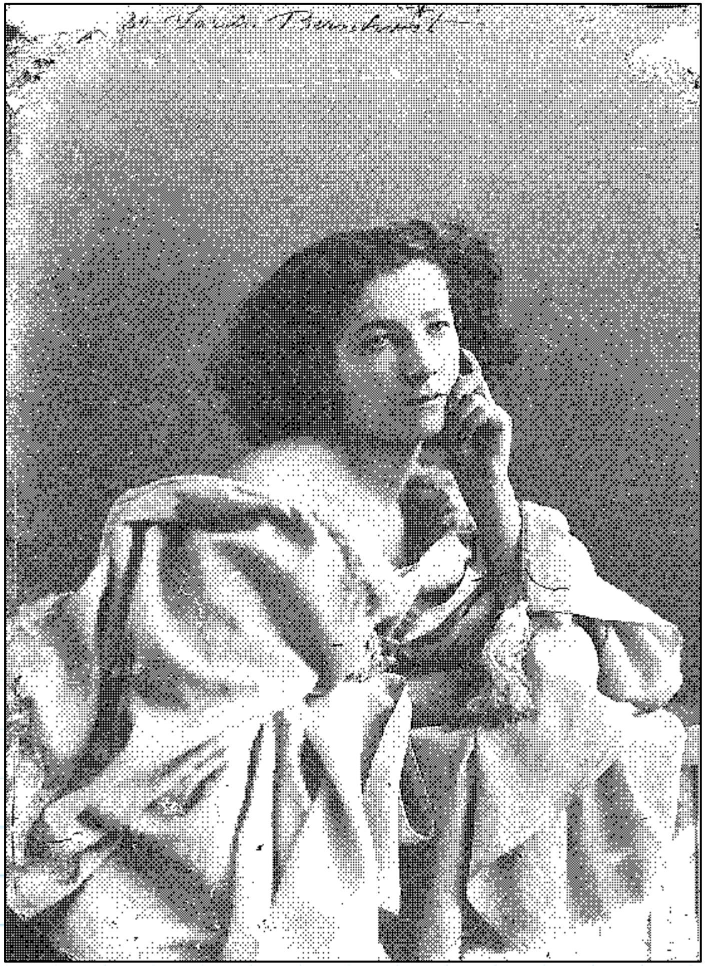
Original







“Screen” matrix

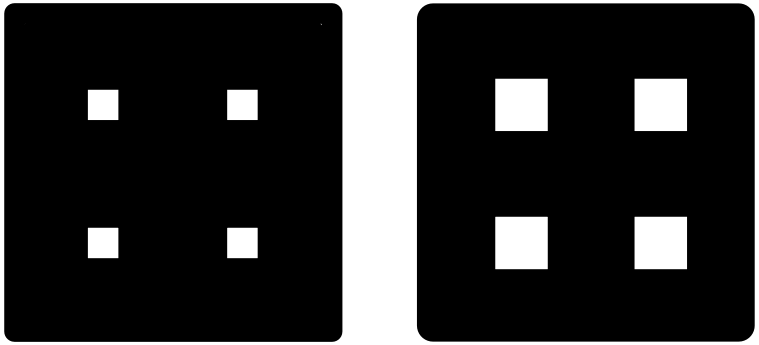


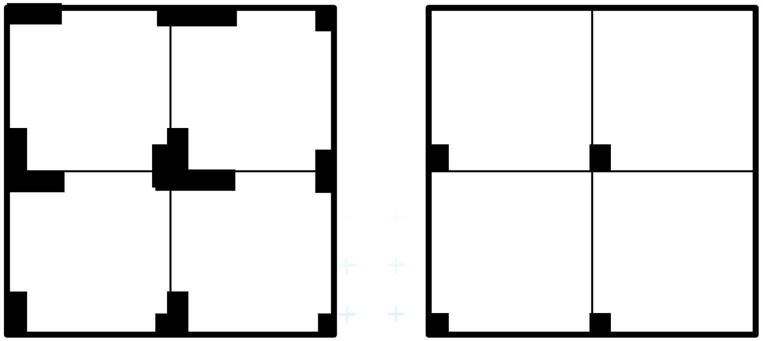
“Newspaper” pattern - clusters of B/W dots

The idea:

Make pattern robust against changes in position/amount of ink

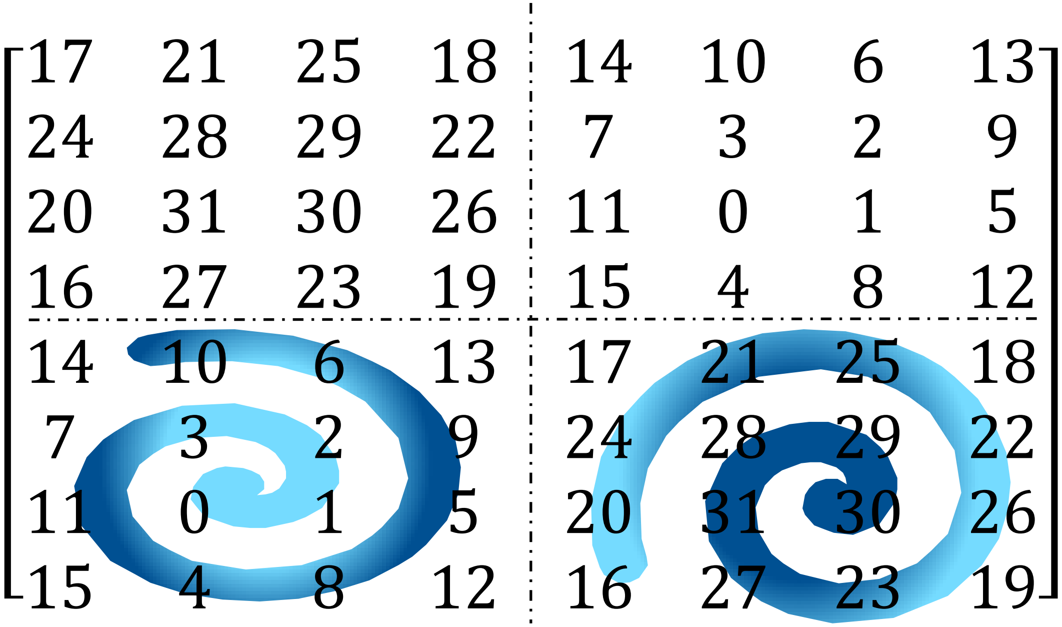
$$M = \begin{bmatrix} 14 & 10 & 6 & 13 \\ 7 & 3 & 2 & 9 \\ 11 & 0 & 1 & 5 \\ 15 & 4 & 8 & 12 \end{bmatrix}$$



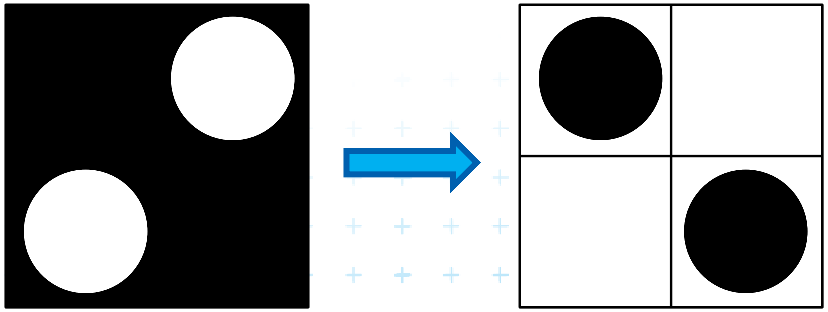


Clusters are horizontally and vertically arranged ☹

Newspaper pattern – dithering matrix

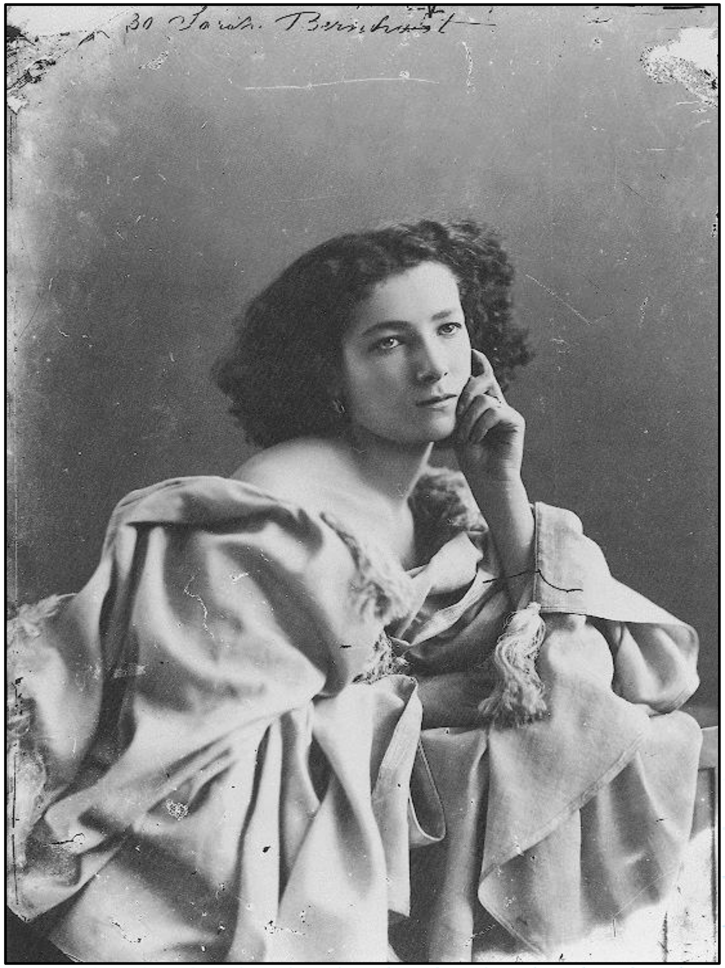


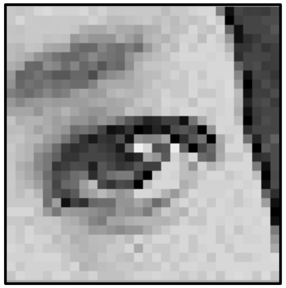
$$\left[\begin{array}{cccc|cccc}
17 & 21 & 25 & 18 & 14 & 10 & 6 & 13 \\
24 & 28 & 29 & 22 & 7 & 3 & 2 & 9 \\
20 & 31 & 30 & 26 & 11 & 0 & 1 & 5 \\
16 & 27 & 23 & 19 & 15 & 4 & 8 & 12 \\
\hline
14 & 10 & 6 & 13 & 17 & 21 & 25 & 18 \\
7 & 3 & 2 & 9 & 24 & 28 & 29 & 22 \\
11 & 0 & 1 & 5 & 20 & 31 & 30 & 26 \\
15 & 4 & 8 & 12 & 16 & 27 & 23 & 19
\end{array}\right]$$

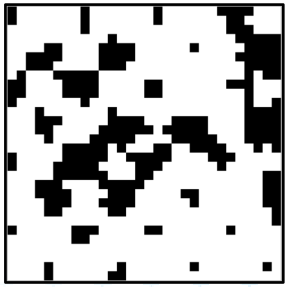


Example

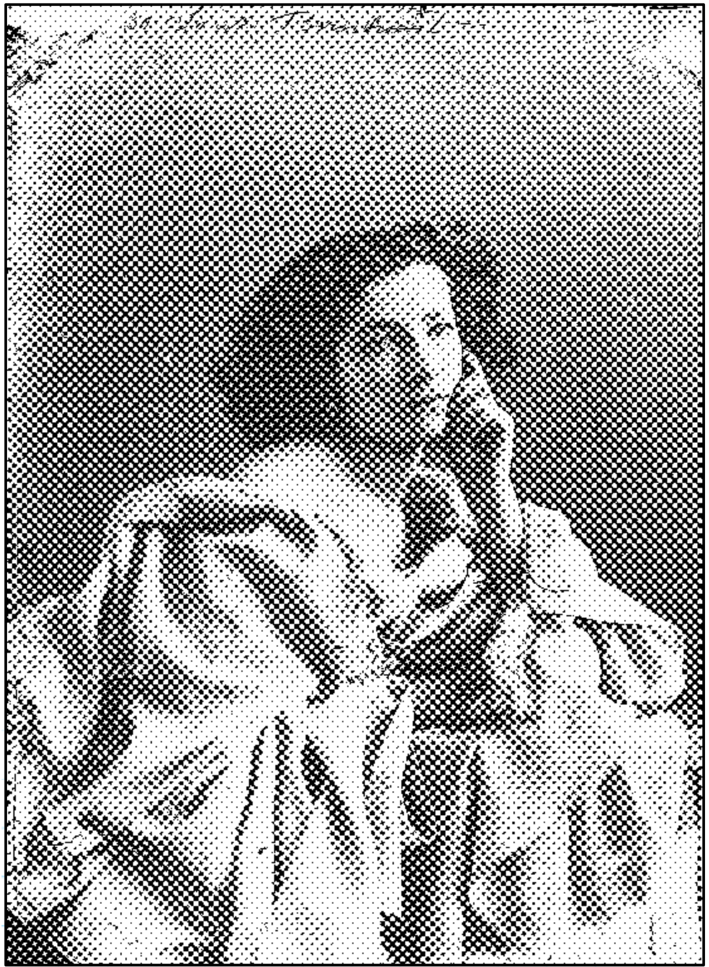
Original







“Newspaper” matrix



Random dithering

- Without information on pixel positions (coordinates)

$$I_{INP} \in \langle 0, N \rangle$$

$$I_{OUT} = (I_{INP} > random(N))$$

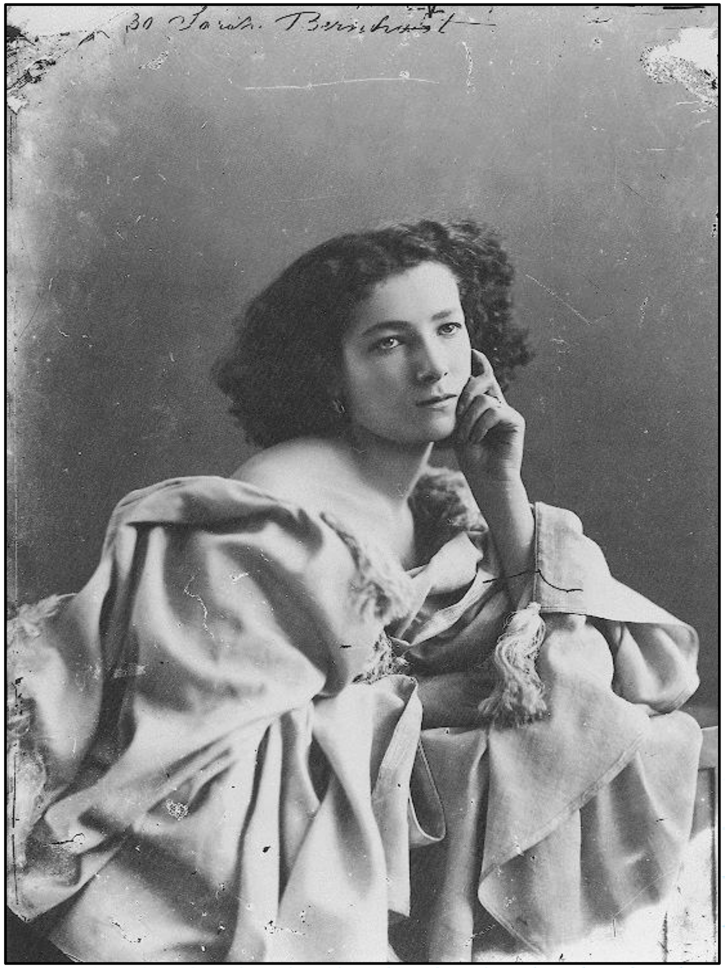
„Yekor“ Dithering

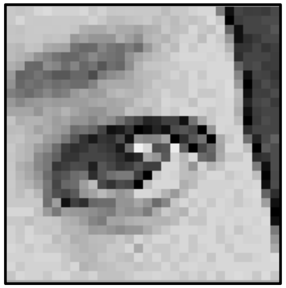
☺

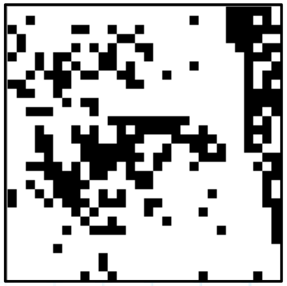
Yekor = a kind of Czech carpet

Example

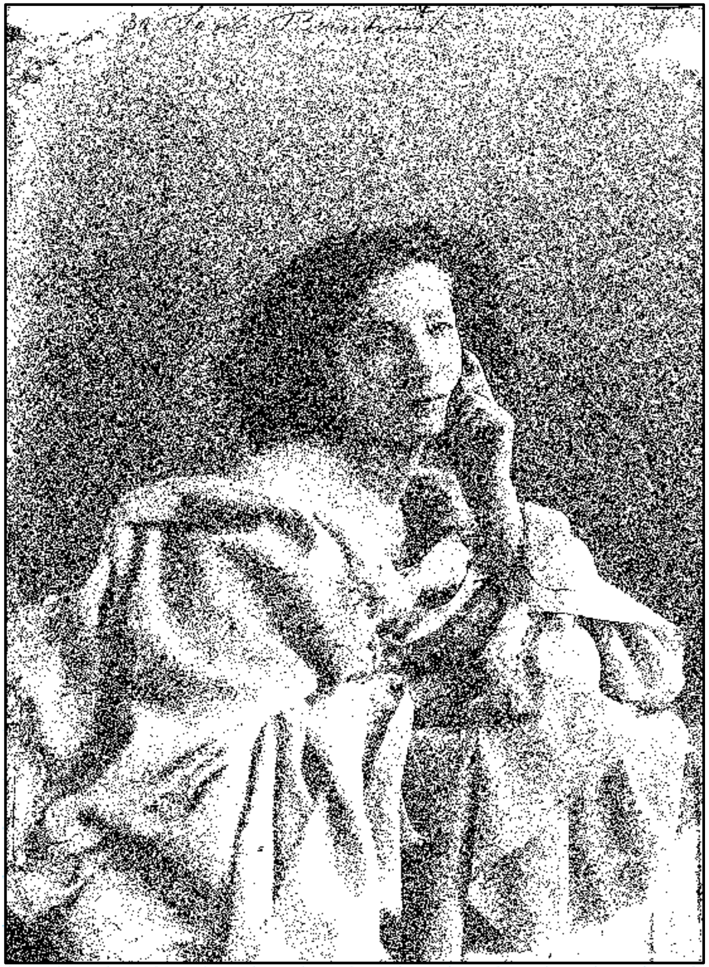
Original





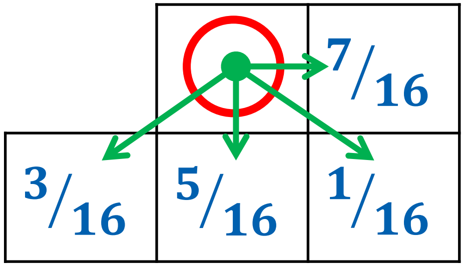


Random dithering



Error diffusion (distribution)

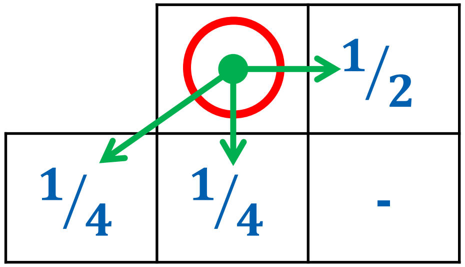
- Floyd-Steinberg



$^{7}/_{16}$

$^{3}/_{16}$ $^{5}/_{16}$ $^{1}/_{16}$

- Frankie Sierra



$^{1}/_{2}$

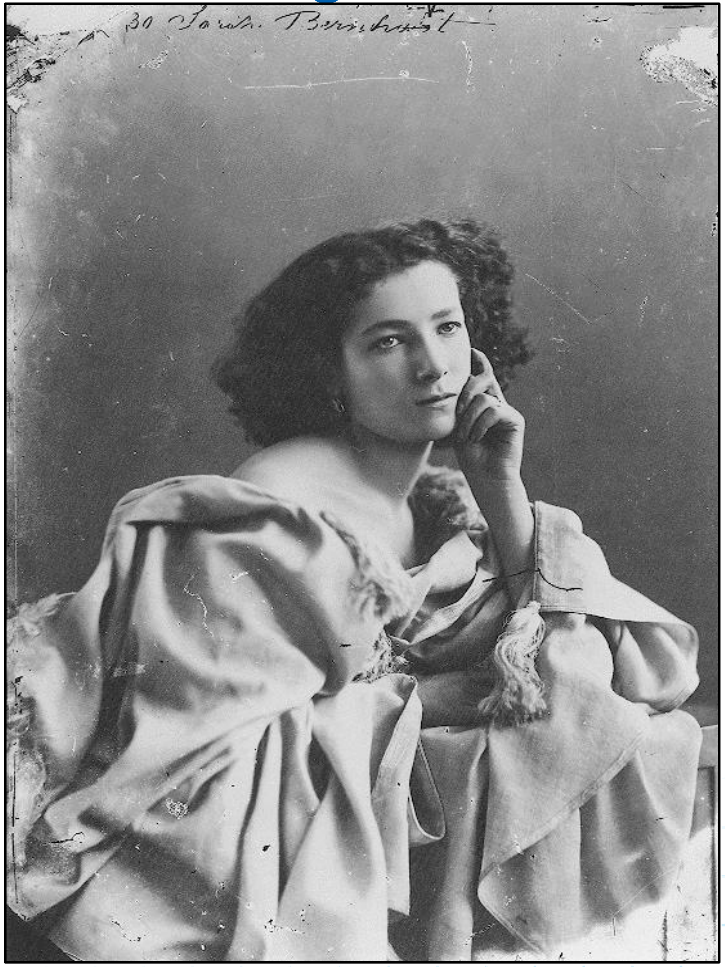
$^{1}/_{4}$ $^{1}/_{4}$ -

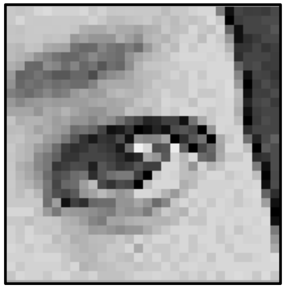
Fast arithmetic

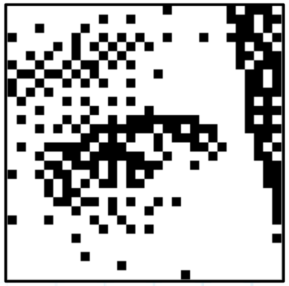
No rounding!
(Compute the last fragment as
a difference from a sum of other ones)

Example

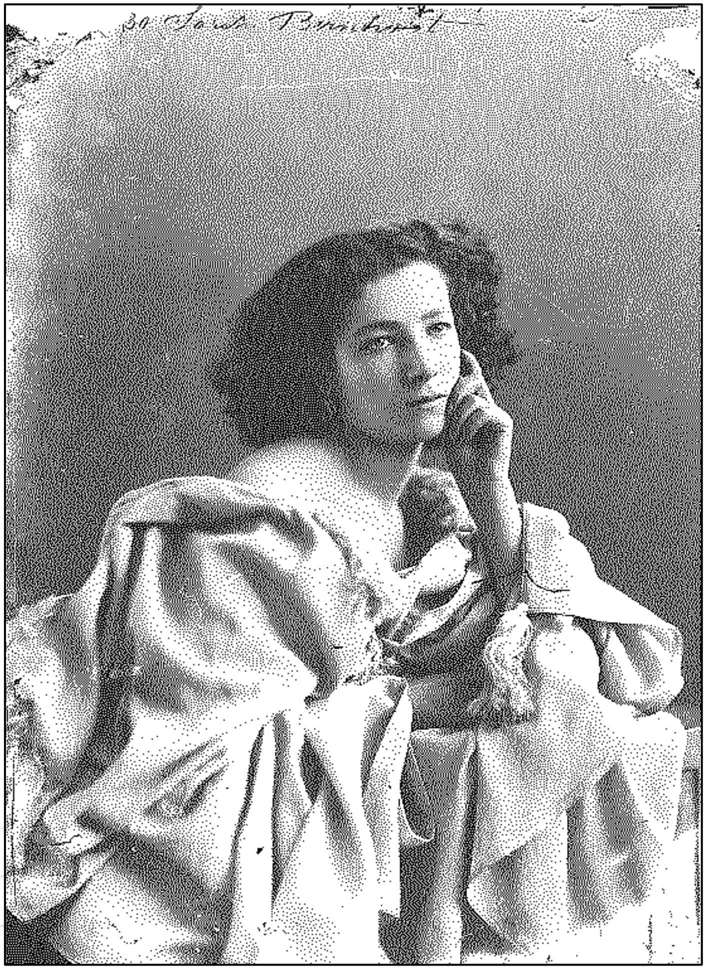
Original



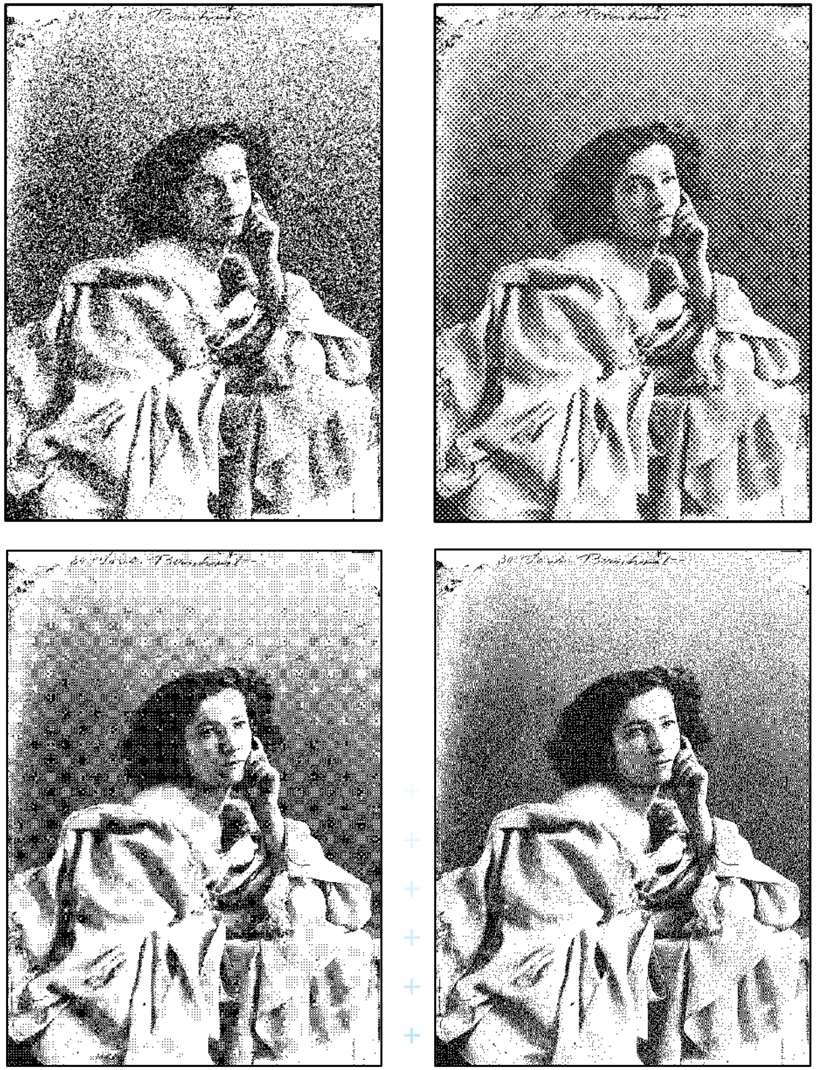


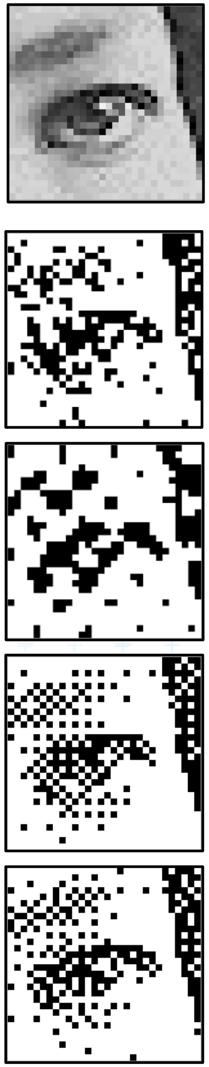


Error diffusion

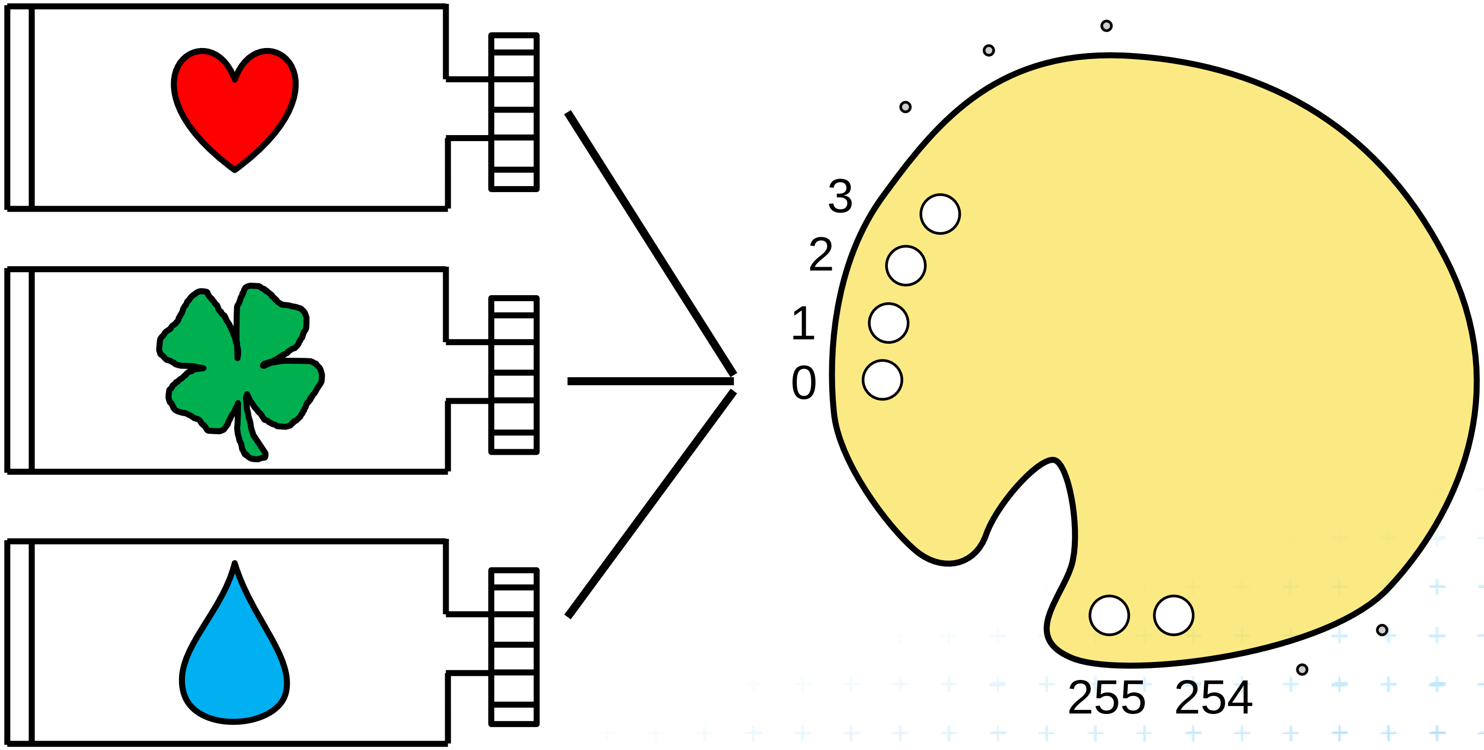


Comparison of dithering techniques





Color palette – Hot to choose colors?



$2^{(8+8+8)}$ = 16 mil.

Two steps: To compute palette + To dither colors

Easy solutions: Without any palette computation

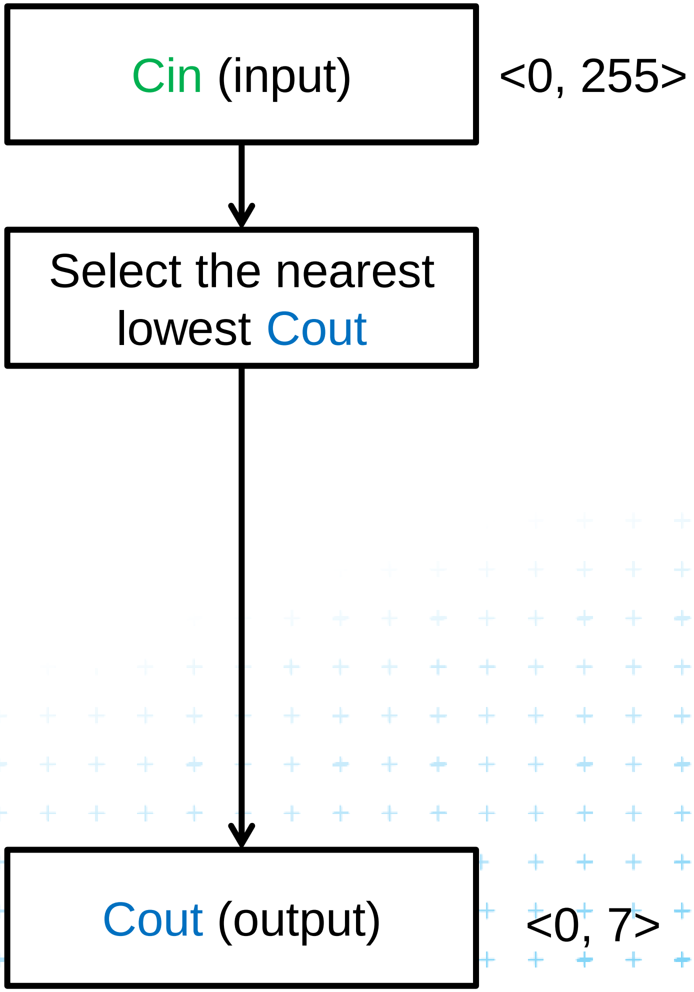
1. No colors => Gray scale (256 grays)

$$I = 0.3 * R + 0.59 * G + 0.11 * B$$

2. Fixed palette **3-3-2** (2^3 = 8 levels of R, etc.)
   - Separated dithering for each color channel

Rounding (= no dithering)

$$Cout = Cin * 7 \; div \; 255$$



Random dithering

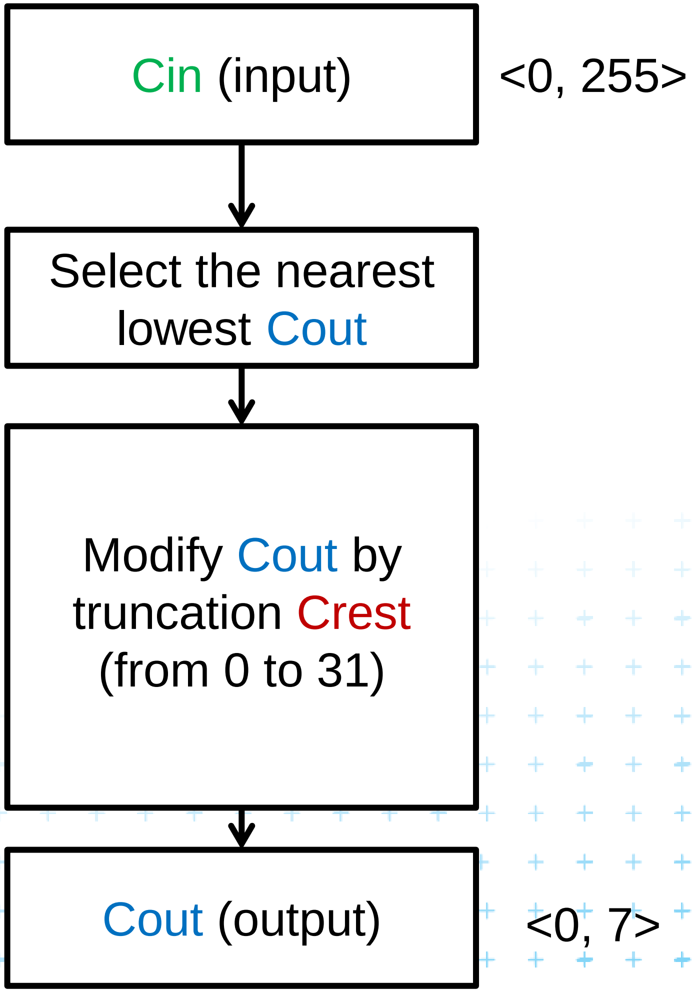
```
$if\ (Crest \geq random(32))\{$
    $inc(Cout)\ ;$
$\}$
```

Example:

Cin = 37

Cout = 1

Crest = 6



Cin (input)

<0, 255>

Select the nearest lowest Cout

Modify Cout by truncation Crest (from 0 to 31)

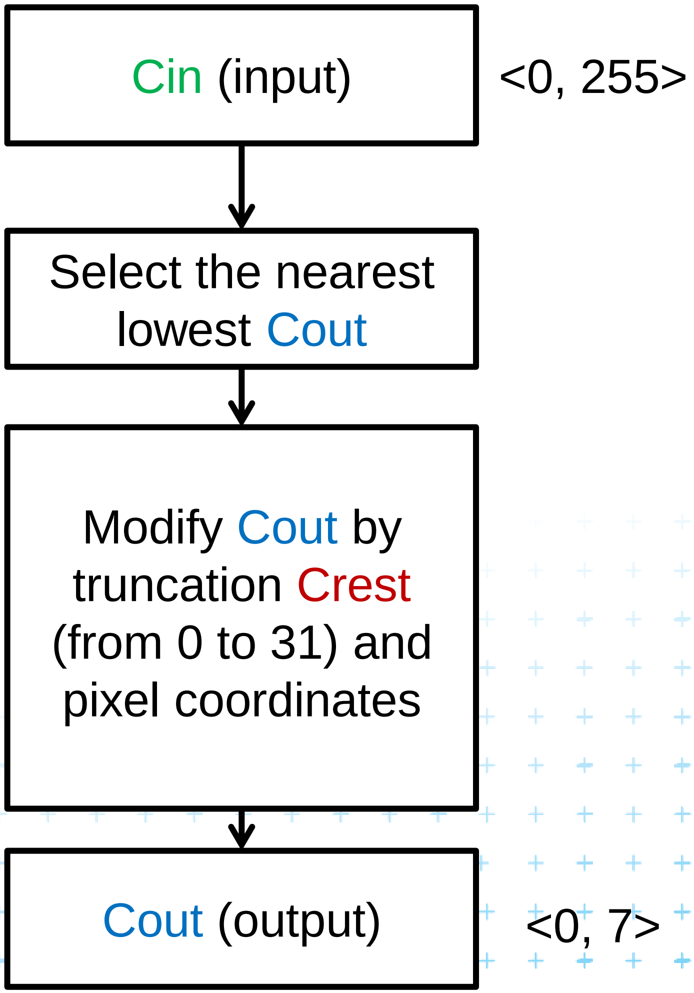
Cout (output)

<0, 7>

Matrix dithering

$$M = \begin{bmatrix} 0 & 25 & 6 & 31 \\ 16 & 8 & 23 & 14 \\ 4 & 29 & 2 & 27 \\ 21 & 12 & 19 & 10 \end{bmatrix}$$

```
$if\ (Crest \geq M[X,Y])\{$
    $inc(Cout)\ ;$
$\}$
```



Cin (input)

<0, 255>

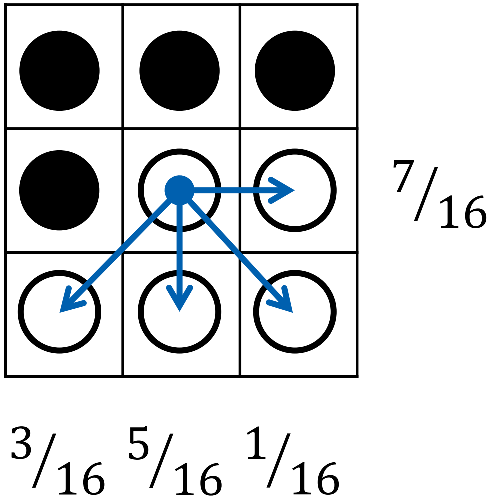
Select the nearest lowest Cout

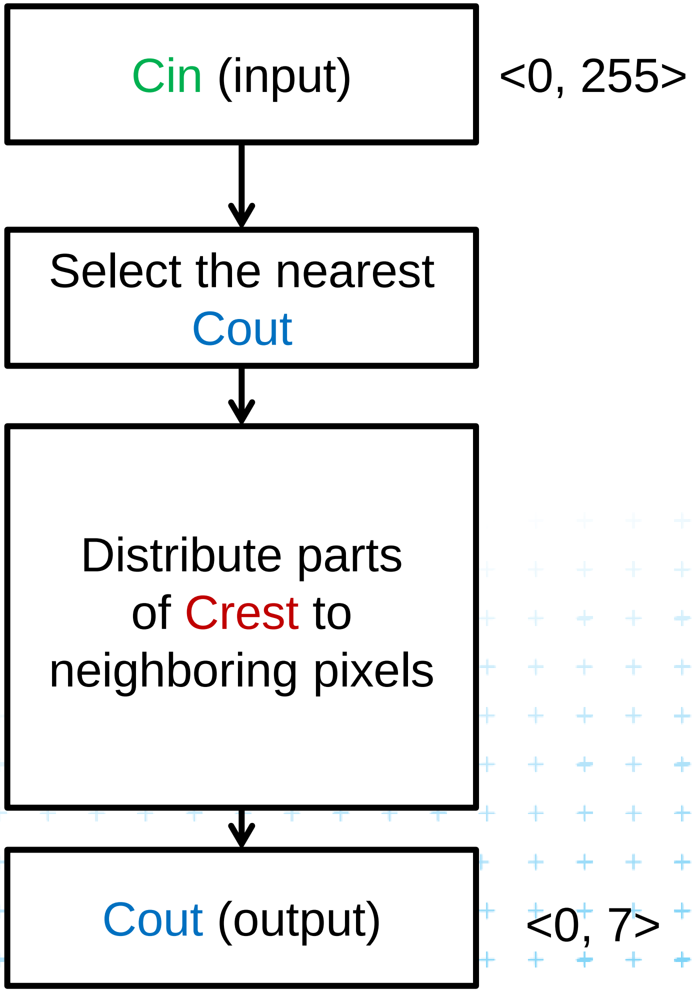
Modify Cout by truncation Crest (from 0 to 31) and pixel coordinates

Cout (output)

<0, 7>

Error diffusion





Cin (input)

<0, 255>

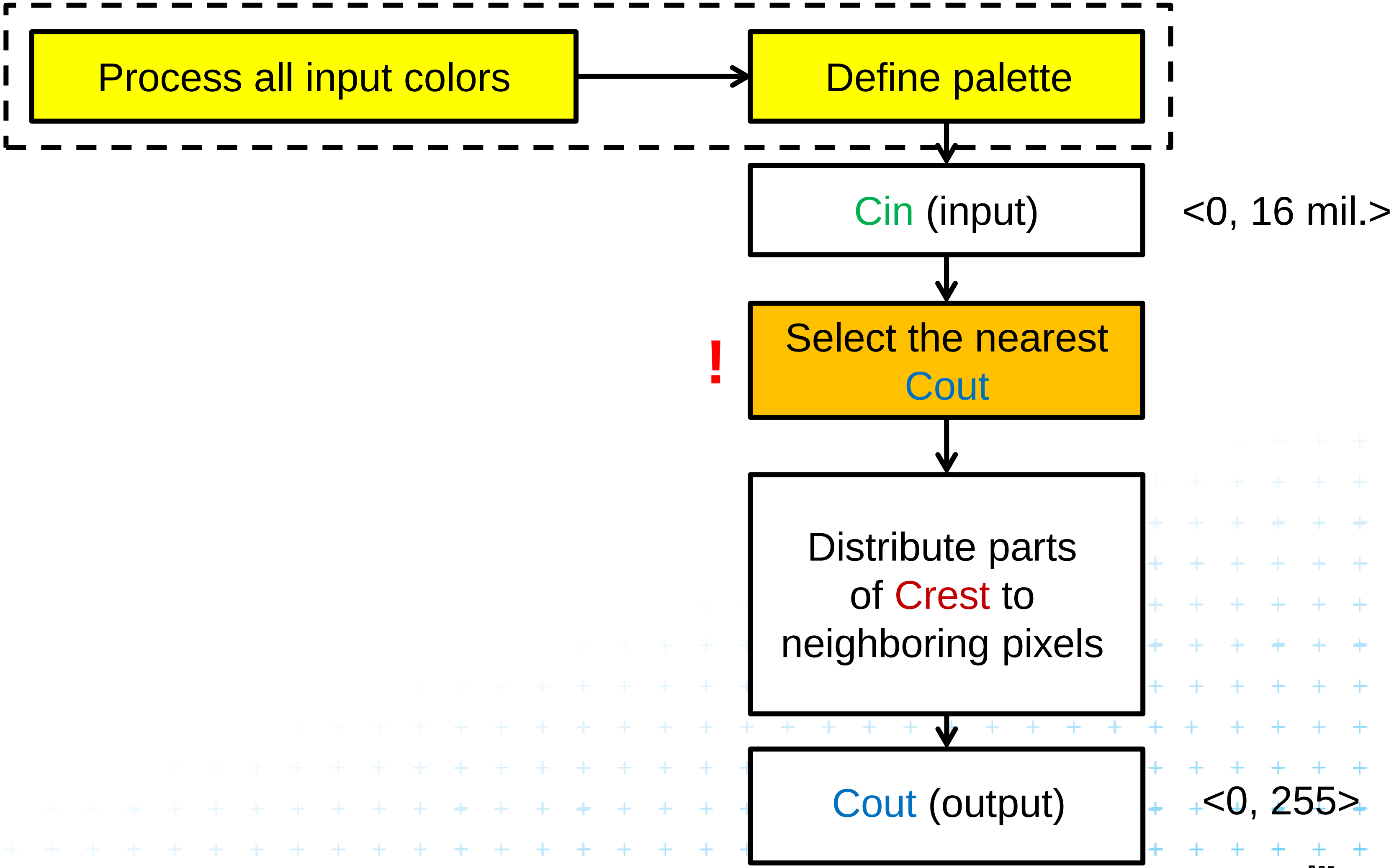
Select the nearest Cout

Distribute parts of Crest to neighboring pixels

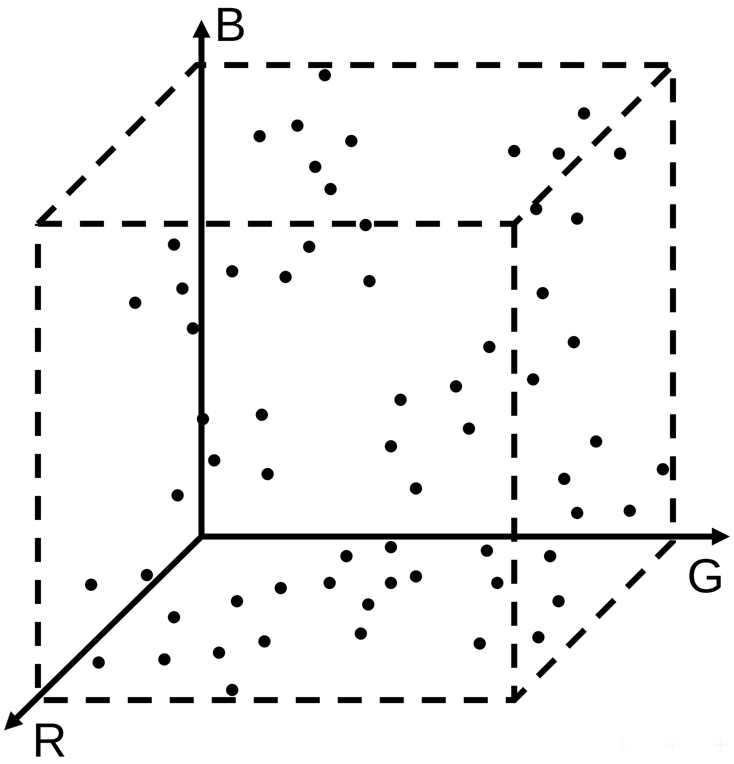
Cout (output)

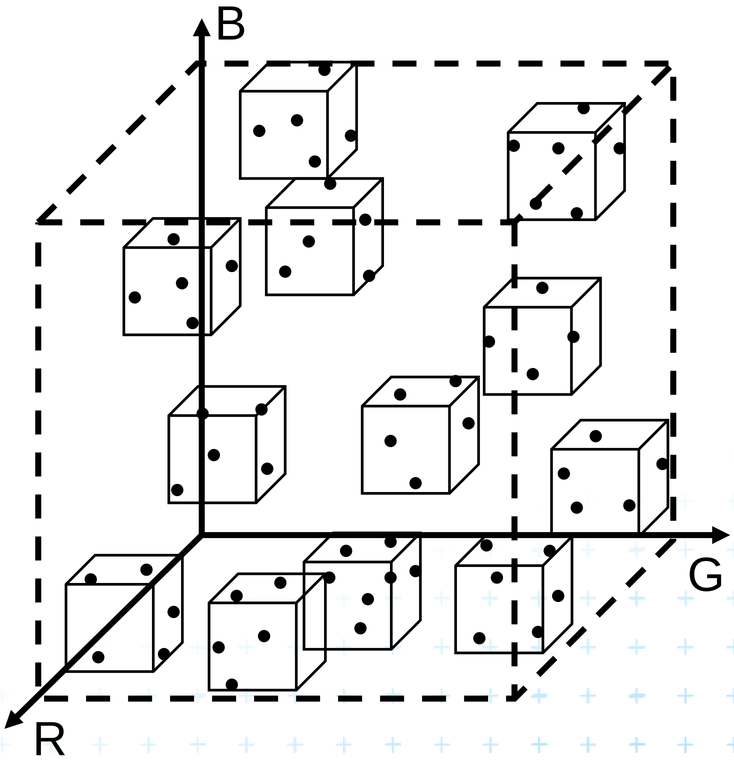
<0, 7>

Adaptive palette



Adaptive palette – clusters in 3D





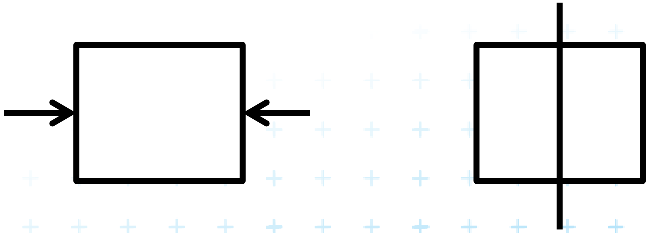
Color Statistics

A. Histogram of all input colors

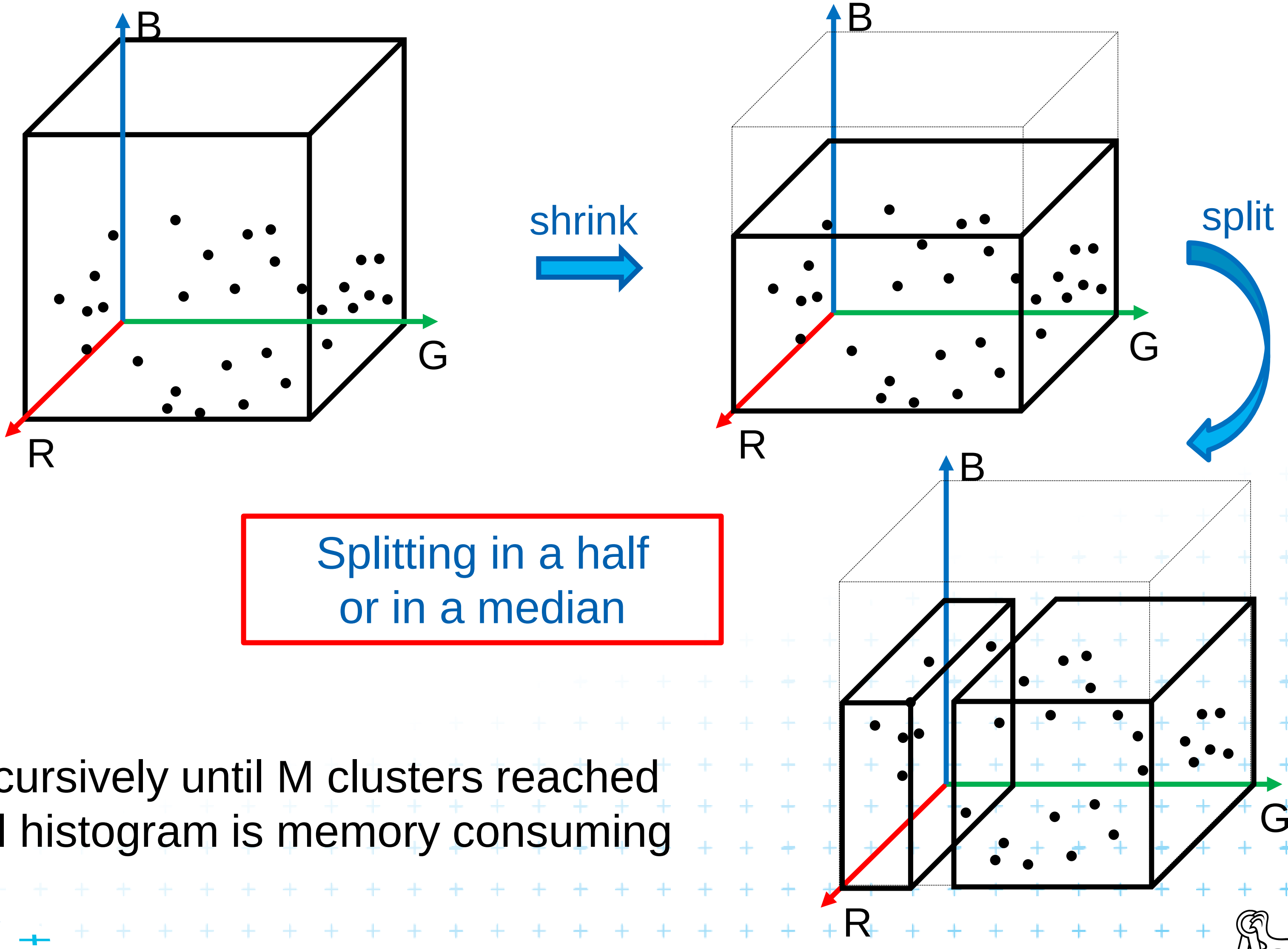
B. Selection of M clusters => reduction to M colors in palette

C. Transformation of input colors to palette colors

- Ad B:
  - – Popularity algorithm (M most frequent colors)
  - – Median-cut algorithm (shrink & split)



Shrink & split



Splitting in a half
or in a median

- Recursively until M clusters reached
- Full histogram is memory consuming

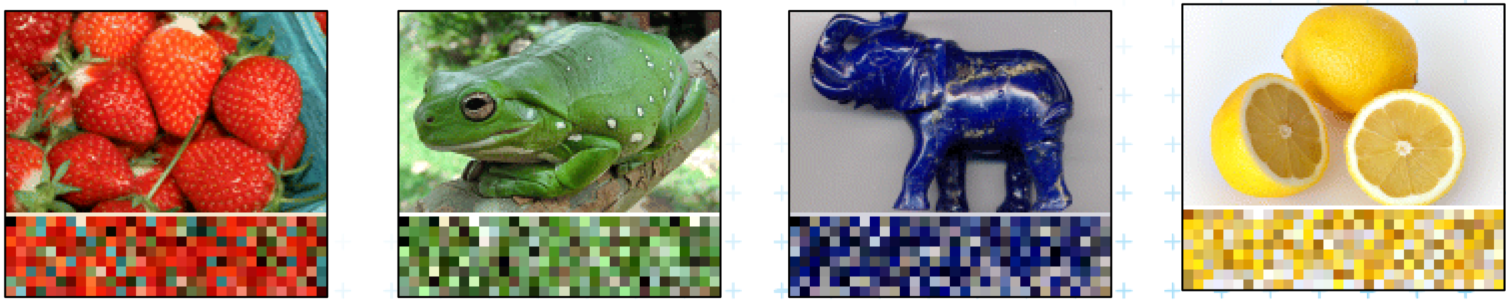
Adaptive palette – Summary

+ Contains „the most suitable“ colors

- Irregular distances among palette colors (obstacle for dithering)

- – Option: combination of partly **static** and partly **adaptive** palette

- Searching the nearest $C_{PALETTE}$ for given input C_i is time consuming (=> k-d tree)



Images from: en.wikipedia.org/wiki/Palette_(computing)

Thank you for your attention

Jiří Žára, 11.1.2017