matfyz

Computer Graphics Charles University

Monochromatické zobrazování

CGG MFF UK Praha

pepca@cgg.mff.cuni.cz

https://cgg.mff.cuni.cz/~pepca/

Vnímání šedých odstínů

Šedý odstín má jediný **atribut**

- **intenzita** (fyzikální smysl, vyzařovaná energie)
- **jas** (subjektivní vjem člověka)

Vztah mezi intenzitou a jasem **není lineární**

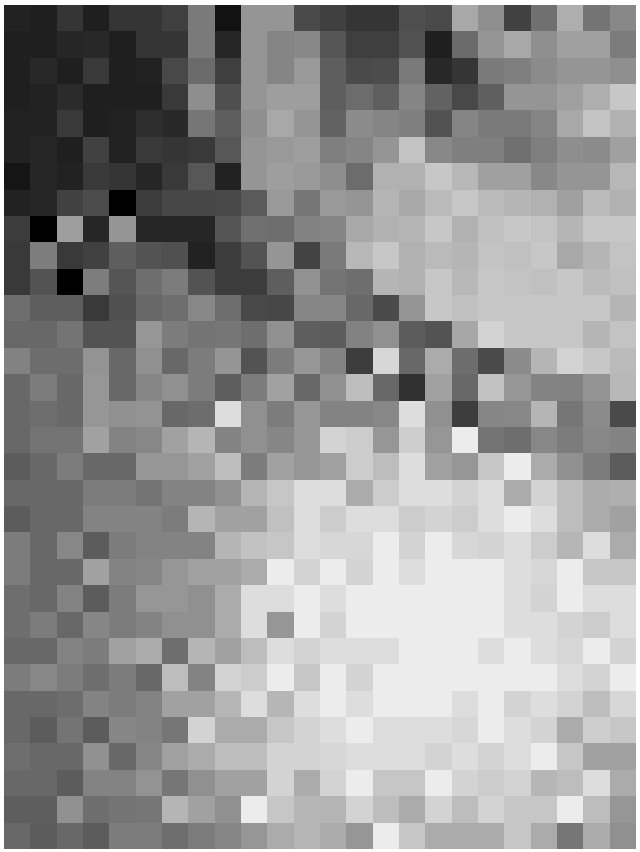
- člověk vnímá intenzity **relativně** (zdravé oko rozezná cca 1% rozdílu)
- pro rovnoměrně odstupňované jasové odstíny je třeba použít **logaritmickou stupnici** intenzit … $I_j = I_0 * r^j$

Počet odstínů šedi

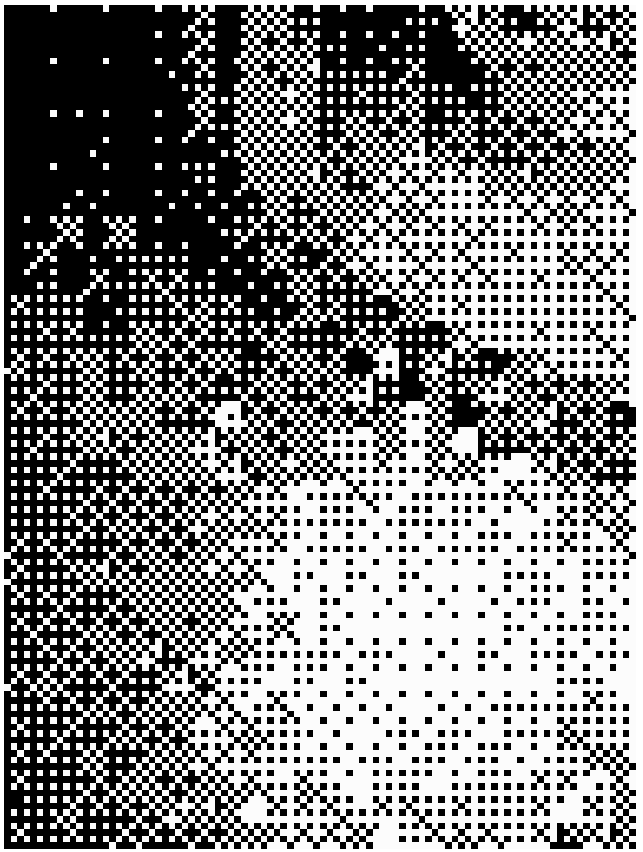
Počet potřebných zobrazovacích odstínů **n** závisí na dynamickém rozsahu výstupního zařízení (předpokládáme **r = 1.01**):

zařízení	dynamika ($1/I_0$)	n
– displej	100-3000	460-800
– fotografie	100	460
– diapozitiv	1000	700
– černobílý tisk	100	460
– barevný tisk	50	400

Půltónování a rozptylování



odstíny šedi



černobílé výstupní zařízení

Půltónování a rozptylování

Napodobení vjemu šedých (barevných) odstínů na zařízení s **malým barevným rozlišením**

- zvětšuji barevné rozlišení na úkor prostorového
- typické použití: **černobílé tiskárny** nebo displeje

Půltónování („halftoning”): na výstupu mohu zvětšit rastrové rozlišení obrázku (1 : N)

Rozptylování („dithering”): musím zobrazovat bez zvětšování (1 : 1)

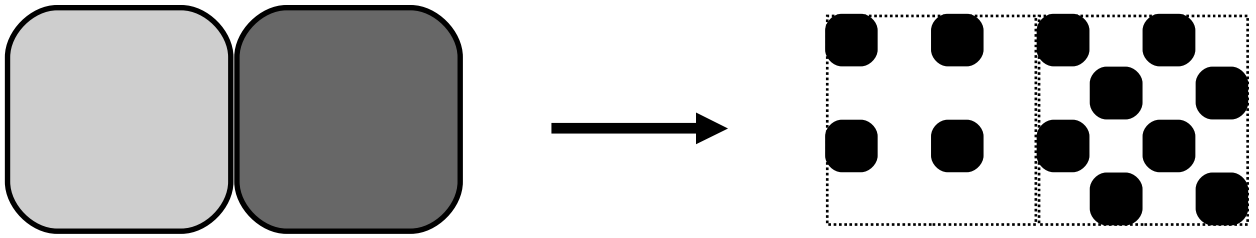
Půltónování

Situace: výstupní zařízení umí zobrazovat pouze **černé body (1)** na **bílém pozadí (0)**

Jeden vstupní pixel (s rozsahem hodnot **0 ÷ N²**) nakreslím jako **čtverec N×N pixelů** na výstupu

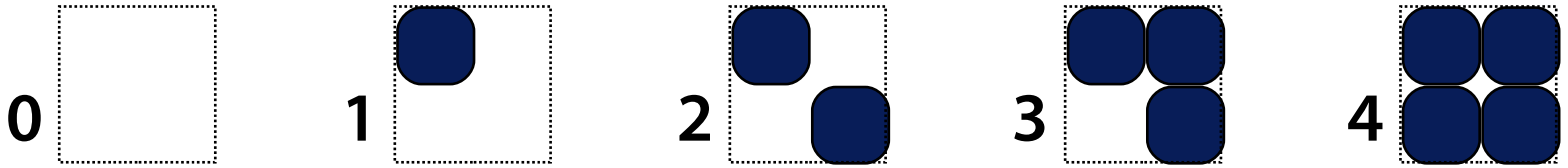
- výsledný vjem šedého odstínu závisí na počtu černých bodů v rastru **N×N**

odstíny číslo 4 a 8 (ze škály 0÷16)

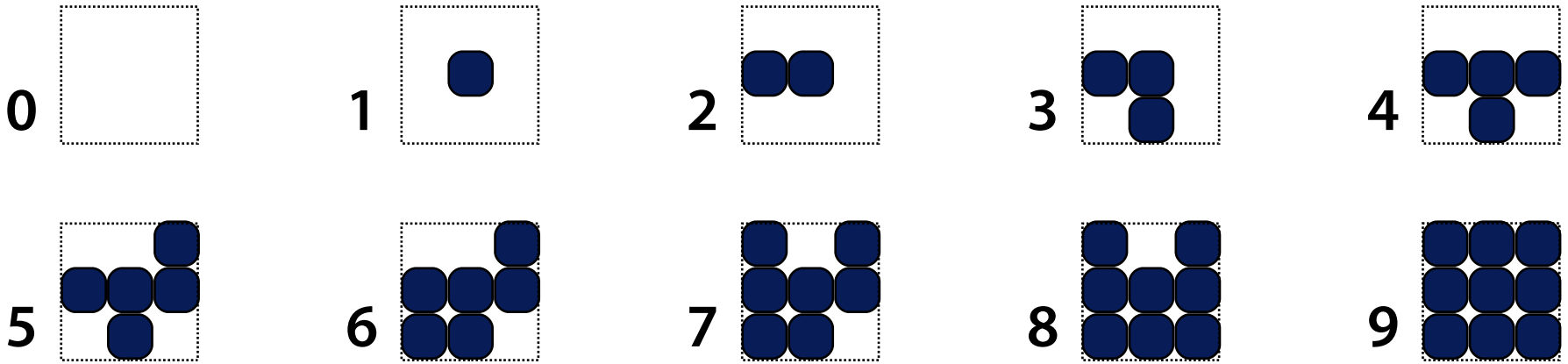


Půltónovací rastry

pravidelný rastr 2×2



rastr 3×3



Inkrementální rastry

Půltónovací rastr je **inkrementální**, jestliže

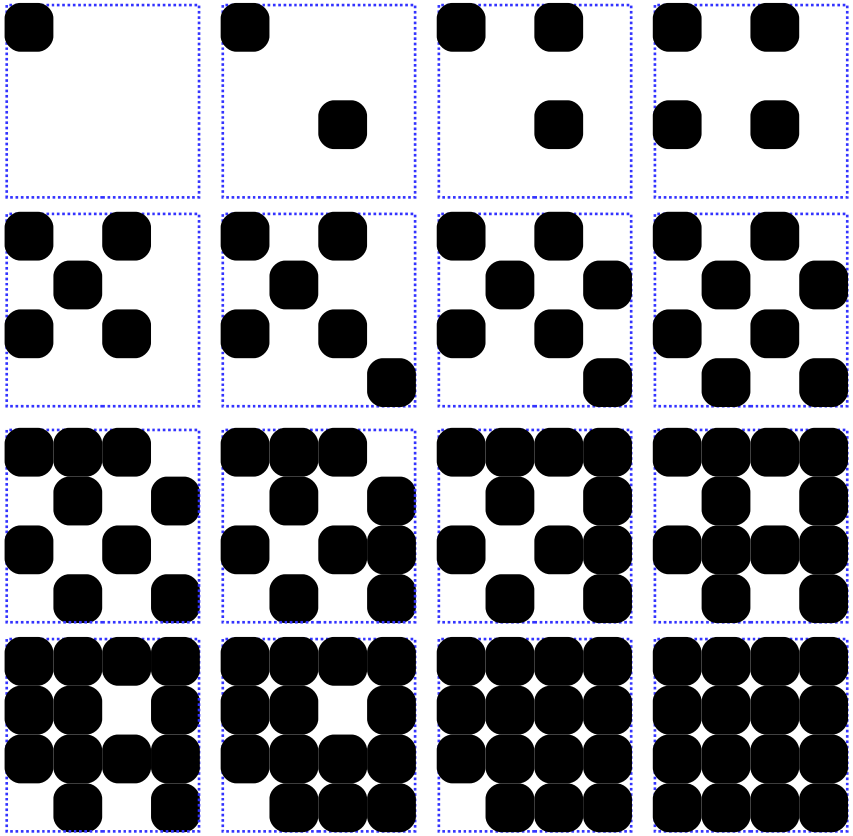
- vzorek odstínu **k** obsahuje právě **k** černých pixelů
- dva sousední vzorky (**k** a **k+1**) se mezi sebou liší právě v jednom pixelu (**k+1** má o jeden černý pixel více)

Inkrementální rastr lze uložit do **matice** velikosti **N×N** obsahující celá čísla **0 ÷ N^2-1**

- např. $\mathbf{M} = \begin{matrix} \mathbf{6} & \mathbf{8} & \mathbf{4} \\ \mathbf{1} & \mathbf{0} & \mathbf{3} \\ \mathbf{5} & \mathbf{2} & \mathbf{7} \end{matrix}$

Pravidelný rastr

I) velikost 2×2 $\quad \mathbf{M}^{(2)} = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}$



II) přechod $\mathbf{N \times N \rightarrow 2N \times 2N}$

$$\mathbf{M}^{(2N)} = \begin{bmatrix} 4\mathbf{M}^{(N)} & 4\mathbf{M}^{(N)} + 2\mathbf{J}^{(N)} \\ 4\mathbf{M}^{(N)} + 3\mathbf{J}^{(N)} & 4\mathbf{M}^{(N)} + \mathbf{J}^{(N)} \end{bmatrix}$$

Matice $\mathbf{J}^{(N)}$ je typu $\mathbf{N \times N}$ a obsahuje samé jedničky

Pravidelný rastr II

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

Body pravidelných vzorků jsou vždy rozmístěny **rovnoměrně**

Pravidelný rastr je vhodný pro **obrazovku** a některé tiskárny (jehličkové s malým rozlišením)

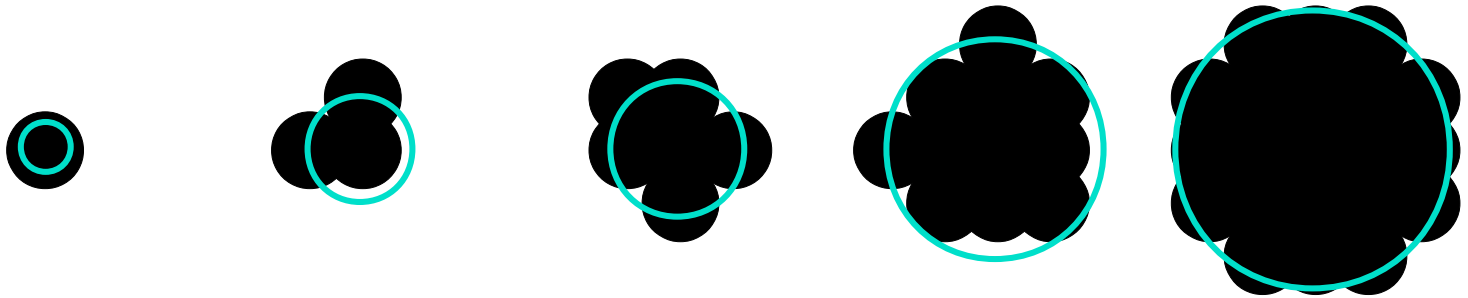
Pravidelný rastr na tiskárně

Odstín 8 na obrazovce a tiskárně s velkým rozlišením

U vyšších odstínů se sousední kapičky barvy **slévají** (důsledek „dot gain“)

Nižší odstíny obsahují **samostatné tečky**, které se špatně udrží na papíře

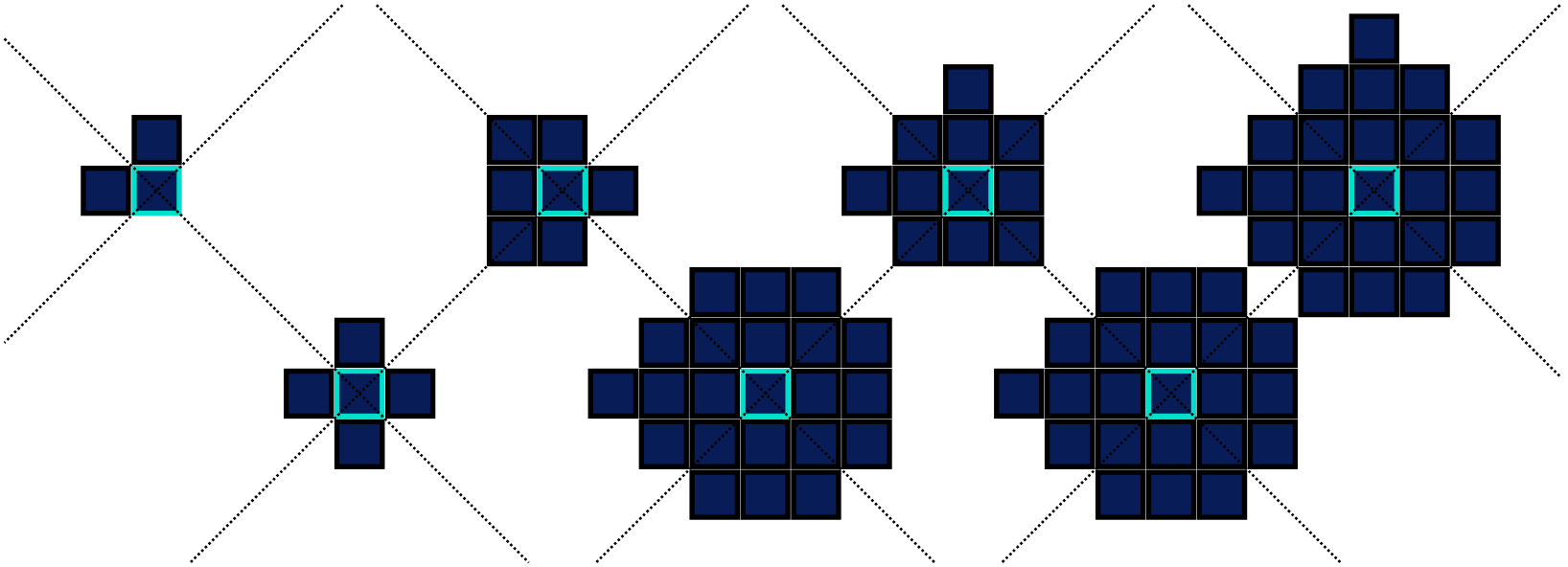
Tečkový rastr („screen”)



Jednotlivé vzorky jsou tvořeny **tečkami** různých poloměrů

- netisknou se samostatné kapičky (až na odstín č. 1)
- rozpíjení kapiček způsobí pouze malou změnu poloměru teček

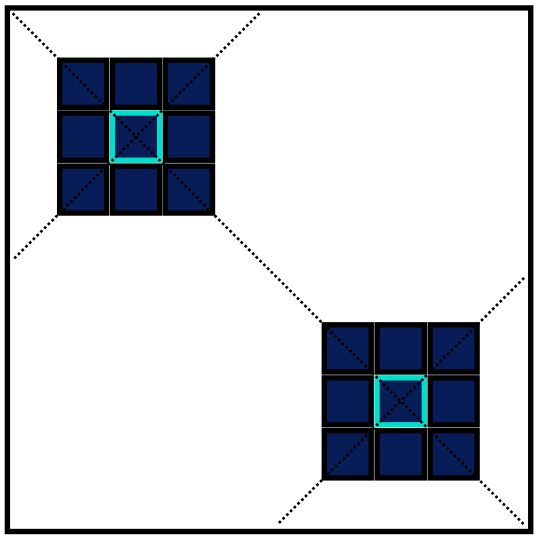
Tečkový rastr – otáčení

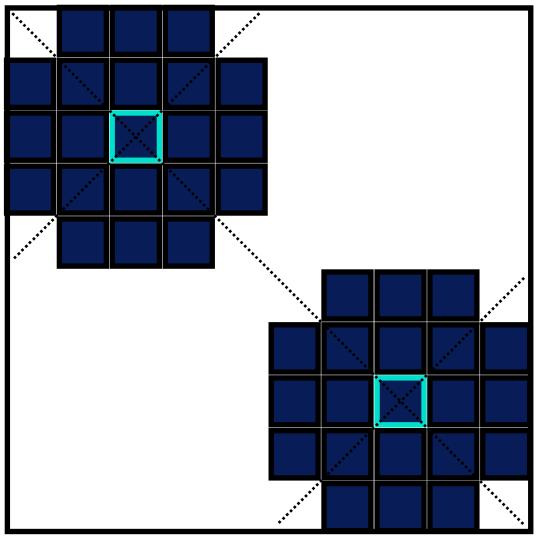


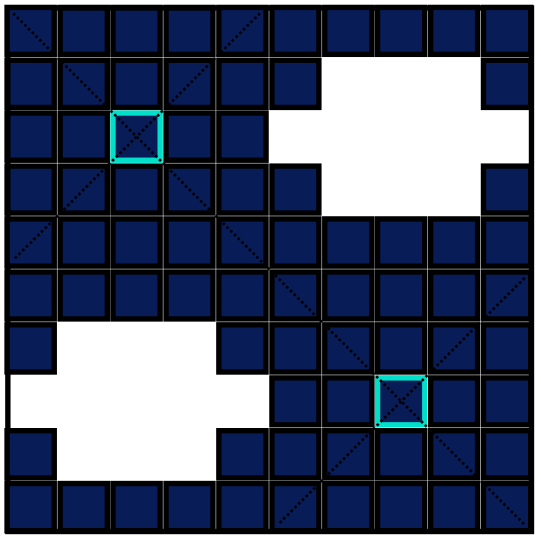
Tečkový rastr se často **otáčí** (o **45°**, 15°, 75° …)

- eliminují se svislé a vodorovné linie (zřetelné pro oko)
- pro racionální směrnice lze tečky ukládat v matici

Varianty tečkového rastru



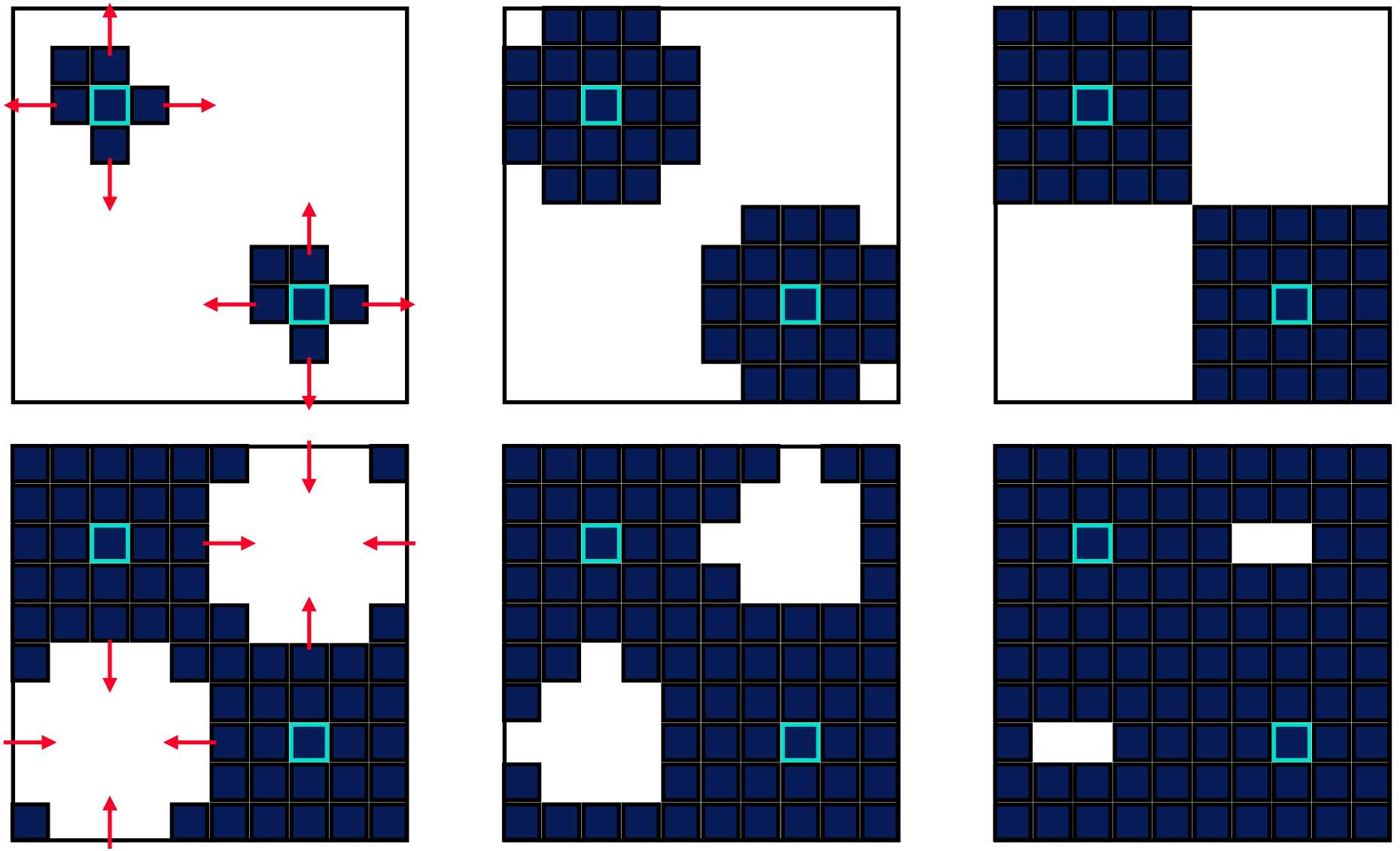




Čtvercové tečky (problémy při jemných přechodech odstínů – „vignettes”)

Kruhové tečky (plus různé modifikace)

Konstrukce tečkového rastru



Maticové rozptylování

Zobrazuje se v **měřítku 1:1** (jeden vstupní pixel na jeden výstupní pixel)

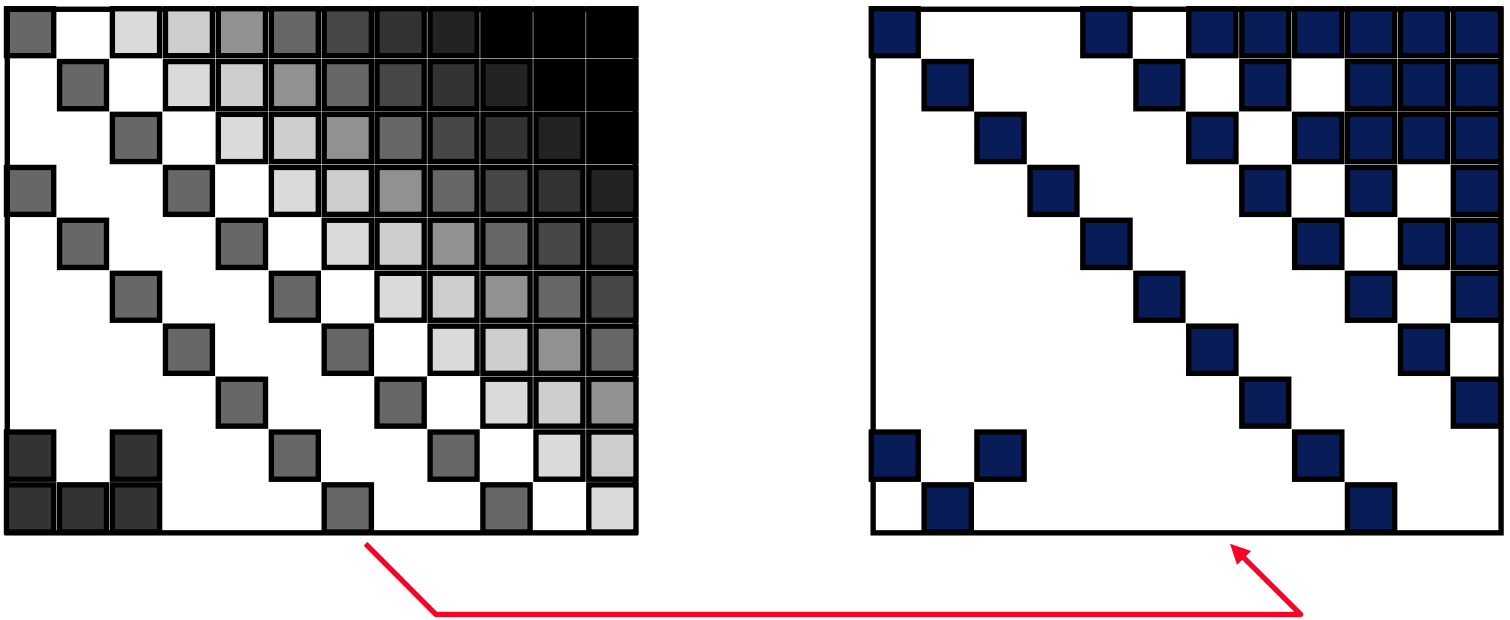
Lze použít libovolnou **půltónovací matici**

- nejčastěji se používá matice pravidelného rastru

Několik sousedních pixelů sdílí jednu matici („M“):

```
void MatrixDither (int x, int y, int gray)
{
  if (M[y % N, x % N] < gray)
    PutPixel(x, y, 1);
  else
    PutPixel(x, y, 0);
}
```

Maticové rozptylování



Drobné detaily (čáry) bývají velmi zkreslené

Při použití **neinkrementálního rastru** by mohly být zvýrazněny hranice mezi sousedními odstíny

Náhodné rozptylování

Šum a nahodilost jsou pro lidské oko přirozenější než pravidelný rastr

Velmi jednoduchá implementace:

```
void RandomDither (int x, int y, int gray)
{
  if (Random(MaxGray) < gray)
    PutPixel(x, y, 1);
  else
    PutPixel(x, y, 0);
}
```

U **Č/B obrázků** je výstup příliš zašuměný

- lepší výsledky dává při **více výstupních odstínech**

Metody distribuce chyby

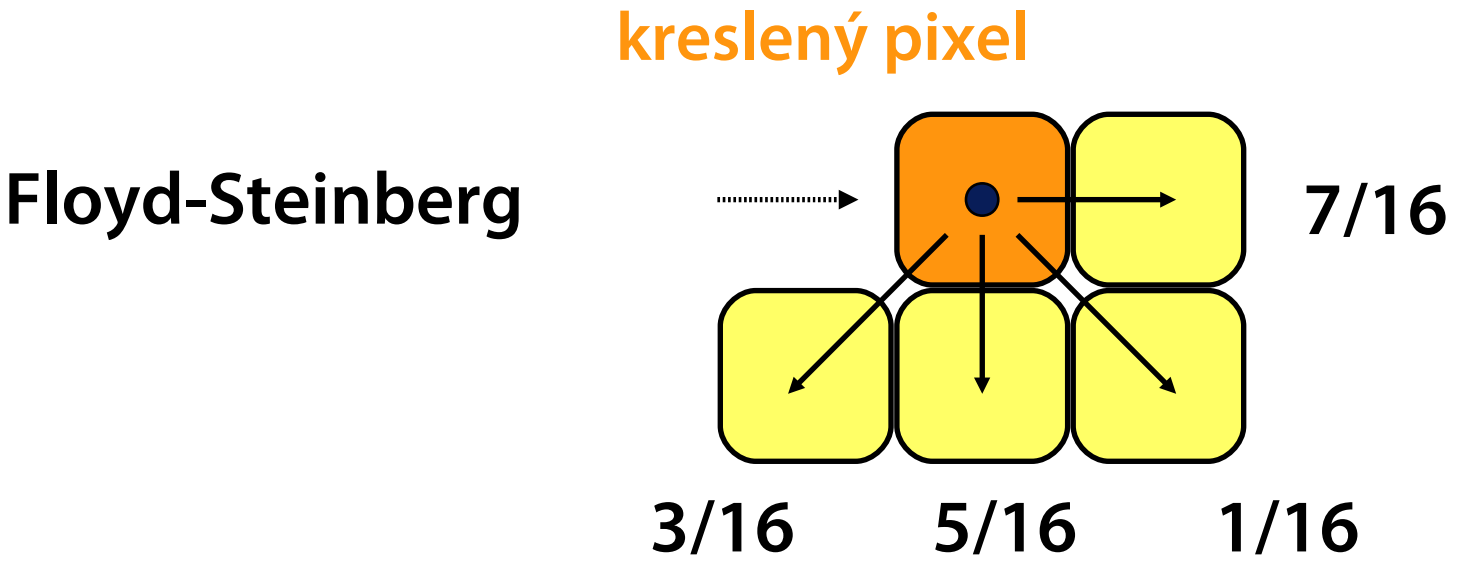
Intenzitu kresleného pixelu **zaokrouhlím** na nejbližší zobrazitelnou hodnotu a nakreslím ji:

- **0/1** pro Č/B výstupní zařízení
- **0, 1, … K** pro víceúrovňové zařízení

Rozdíl mezi požadovanou a skutečně zobrazenou intenzitou rozdělím ve vhodném poměru do sousedních pixelů

- je zachován lokální poměr počtu černých a bílých pixelů (odpovídající vstupnímu odstínu)
- chyba se předává jen do dosud nenakreslených pixelů

Metoda distribuce chyby



Kreslení musí postupovat **po řádkách**

- řádky lze procházet střídavě zleva a zprava

Pro **akumulaci chyb** na následující řádce je nutné použít **pomocný buffer**

Jiné distribuční filtry

F. Sierra

		●	1/2
1/4	1/4	0	

J. Jarvis, C. Judice, W. Ninke

		●	7	5
3	5	7	5	3
1	3	5	3	1

/ 48

Stucki

		●	8	4
2	4	8	4	2
1	2	4	2	1

/ 42

Metody distribuce chyby

Vysoká kvalita výstupu na monitoru

- vzorek je nepravidelný a příjemný pro lidské oko

Nevýhody

- nutnost kreslit výstup **po řádkách**
- není možné se **vracet zpět** (proto se nepoužívá ve vyplňovacích rutinách grafických knihoven)
- je potřeba **pomocný buffer** minimálně na 1 řádku
- větší **časová náročnost**

Randomizace distribuce chyby

Viditelné artefakty

- drobné pravidelnosti, řetízky černých pixelů…

Redukce pravidelností

- **střídání směru** při průchodu jednotlivými řádky
- **randomizace** … zavedení „**modrého šumu**“
  - » randomizace distribučního filtru
  - » randomizace zaokrouhlovací meze
  - » různé alternativní distribuce (PDF) řízené filtrem

Více výstupních odstínů

Na výstupu předpokládáme **K+1 odstínů**

- **0** ÷ **K** (**0** … bílá, **K** … černá)

Naše rozptylovací metoda umí zpracovat **M+1 vstupních odstínů** do dvou výstupních barev:

- vstup: **0** ÷ **M**
- výstup: **0** / **1**

Na vstupu kombinované metody může být **K** · **M+1** odstínů

Více výstupních odstínů

```
int Dither (int x, int y, int gray);
  // vstupní odstín: 0 až M
  // vrací: 0 nebo 1
  // libovolný rozptylovací předpis...


void MultiDither (int x, int y, int gray)
  // vstupní odstín: 0 až K * M
  // kreslený odstín: 0 až K
{
  int base = gray / M;      // 0 <= base <= K
  PutPixel(x, y, base + Dither(x, y, color % M));
}
```

Literatura

J. Foley, A. van Dam, S. Feiner, J. Hughes: ***Computer Graphics, Principles and Practice***, 563-573

R. Ulichney: ***Digital Halftoning***, MIT Press, 1987

D. Lau, G. Arce: ***Modern Digital Halftoning***, M. Dekker, 2001

[**J. Jarvis, C. Judice, W. Ninke:** ***A Survey of Techniques for the Image Display of Continuous Tone Pictures on Bilevel Displays***, CGIP vol.5, #1, March 1976]