Programování grafických aplikací
základní operace s obrazem

Radek Richtr

Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic
Prague, Czech Republic

October 6, 2020

Outline I

Intro

- teoretická přednáška

Intro

- teoretická přednáška
- přehled algoritmických podkladů

Intro

- teoretická přednáška
- přehled algoritmických podkladů
- ... a matematických podkladů pro grafické algoritmy

Intro

- teoretická přednáška
- přehled algoritmických podkladů
- ... a matematických podkladů pro grafické algoritmy
- matematika je základ, bez kterého to nejde (alespoň její intuitivní pochopení)

Matematické minimum

- rovnice, parametrické rovnice

Matematické minimum

- rovnice, parametrické rovnice
- sumy, integrály

Matematické minimum

- rovnice, parametrické rovnice
- sumy, integrály
- matice, násobení matic

Matematické minimum

- rovnice, parametrické rovnice
- sumy, integrály
- matice, násobení matic
- transformace

Matematické minimum

- rovnice, parametrické rovnice
- sumy, integrály
- matice, násobení matic
- transformace
- determinant

Obraz

- Nejzákladnějí pojmy bývají nejobtížněji definovatelné:
  - obraz,
  - textura, atd.
- Objekty jsou popsány matematicky (vektor) - usečky, křivky, text, ...
  - co nejpřesnější vykreslení spojitých útvarů na diskrátní mřížce
  - ořezávání objektů na výřez daný plochou obrazovky
  - nízkoúrovňové operace
- K vykreslení ale chceme matici pixelů (frame) v *rastru*
- Tedy nějaké X, Y a hodnotu
- Jaká je programová reprezentace restru?
  - `int[X][Y]`?

Reprezentace obrazu

- X, Y - pozice v rastru
- S - Spektrum
  - Spektrum je dáno *barevým modelem* zvoleným k reprezentaci
  - *Barevné modely* - RGB (BGR), YUV, HSV, L*A*B, ...
- Rozsah obvykle 0-255 $=> 256^3 = 16777216$ odstínů
- Kousek historie
  - High color - 5-6-5 (2B na RGB)
  - True color - 8-8-8 (3B na RGB)
- kvantování, vzorkování - historie, nebo současnost?

Reprezentace obrazu

- Reprezentace obrazu pomocí např. RGB hodnot je tzv. *přímá* (direct color)
- Alternativou je *indexový mod*
  Každá barva má svůj index, neuvádní se u každého pixelu složení kažé barvy, ale jen index (ukazatel) dané barvy
- Který je výhodnější a kdy?
  - při ztmavení barev
  - v obrazu zapadajícího slunce
  - při kompresi obrazu

Reprezentace obrazu

- Formáty
  - bmp
  - gif
  - jpg
  - png
  - atd.

Reprezentace obrazu

- A co video?
- Sekvece framů?
- Jaký datový typ a rozsah, organizace v paměti?
- Programová reprezentace?

2D objekty

- Úsečky
- Kružnice a elipsa (Bresenham)
- Vyplňování (semínkové vyplňování)
- Polygon
- 4-okolí a 8-okolí
- Ořezávání (Cohen-Sutherland)

2D objekty

- Objekty mohu být vektorové…

2D objekty

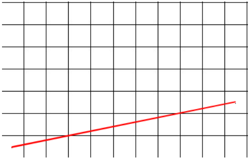
- Objekty mohu být vektorové. . .
- ale jsou vykreslené v rastru!

2D objekty

- Objekty mohu být vektorové. . .
- ale jsou vykreslené v rastru!
- Které z pixelů 'rosvítit'?

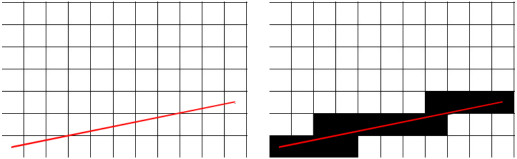
2D objekty

- Které pixely v rastru 'rozsvítit'?



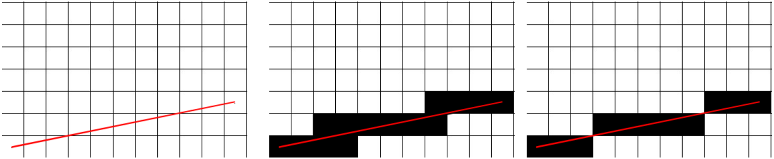
2D objekty

- Které pixely v rastru 'rozsvítit'?



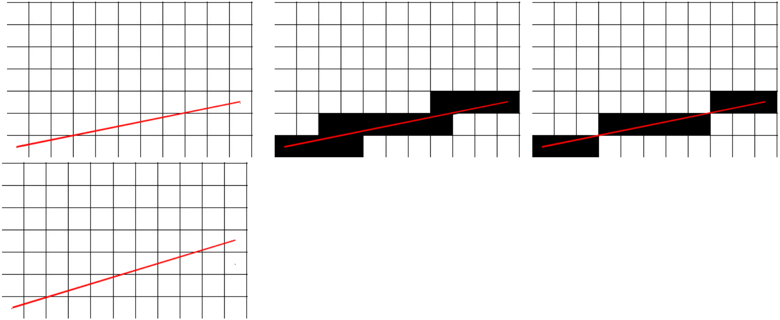
2D objekty

- Které pixely v rastru 'rozsvítit'?



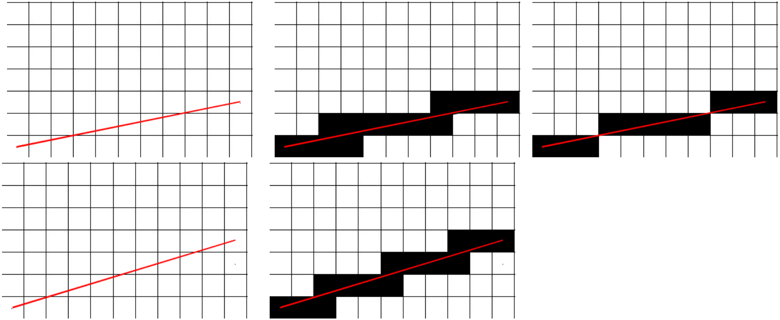
2D objekty

- Které pixely v rastru 'rozsvítit'?



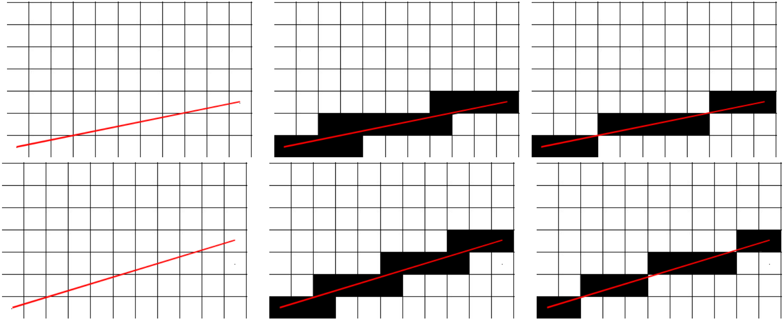
2D objekty

- Které pixely v rastru 'rozsvítit'?



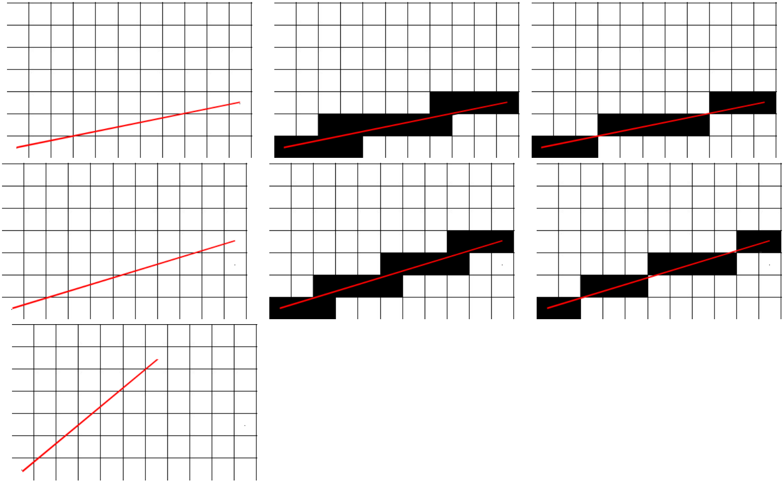
2D objekty

- Které pixely v rastru 'rozsvítit'?



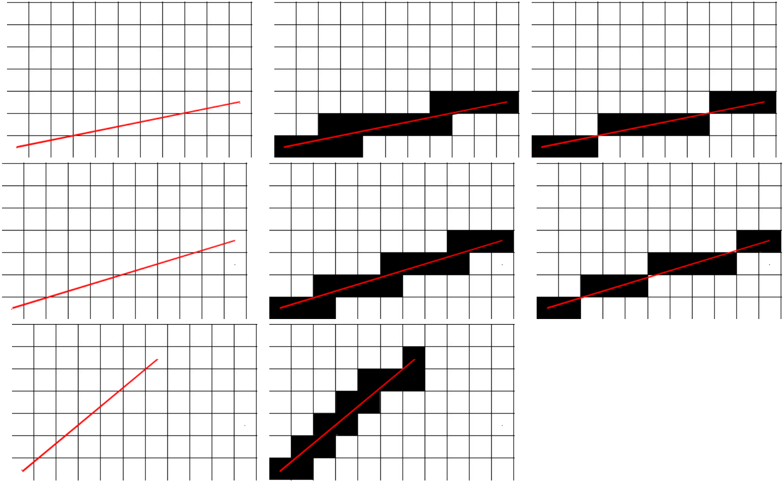
2D objekty

- Které pixely v rastru 'rozsvítit'?



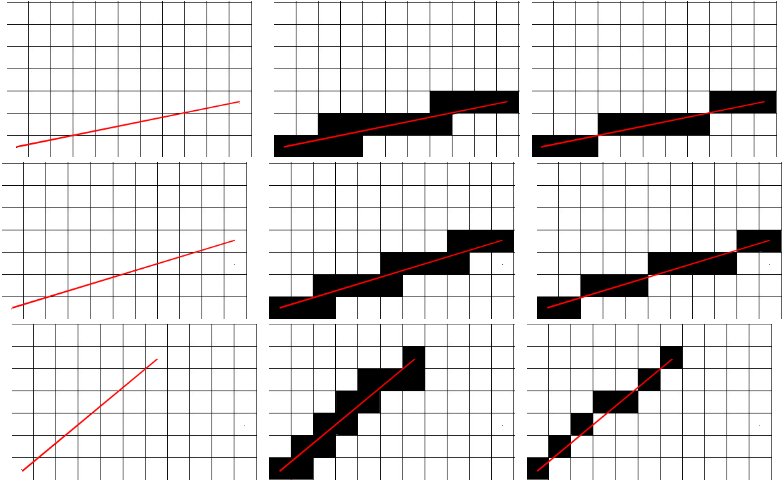
2D objekty

- Které pixely v rastru 'rozsvítit'?



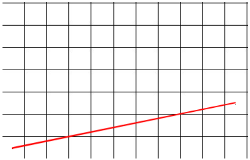
2D objekty

- Které pixely v rastru 'rozsvítit'?



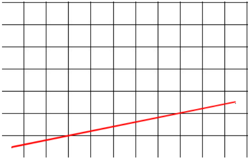
2D objekty

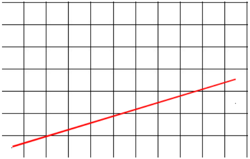
- Které pixely v rastru 'rozsvítit'?



2D objekty

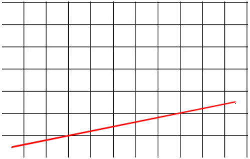
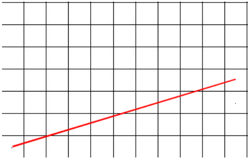
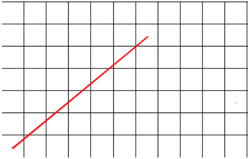
- Které pixely v rastru 'rozsvítit'?





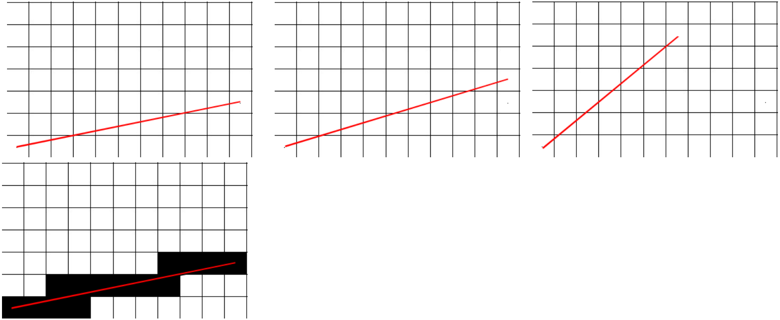
2D objekty

- Které pixely v rastru 'rozsvítit'?

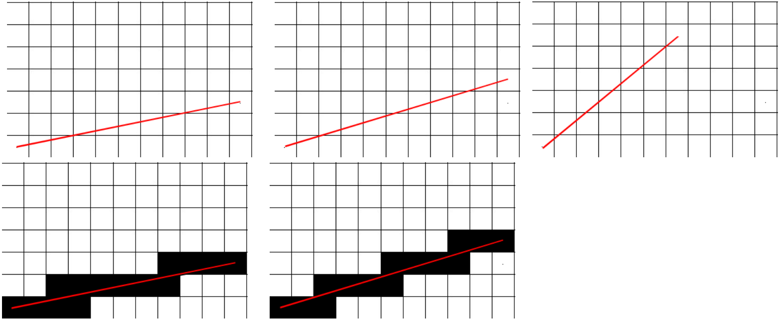
2D objekty

- Které pixely v rastru 'rozsvítit'?



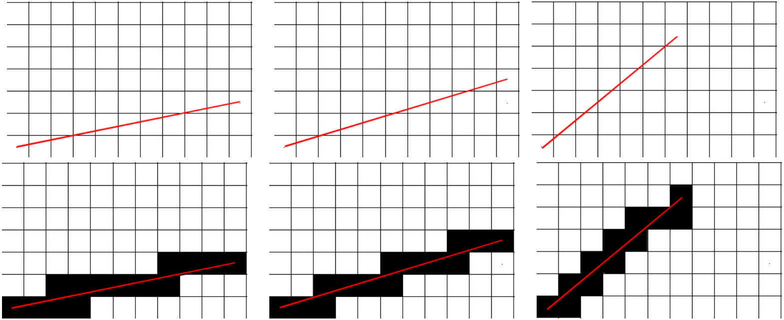
2D objekty

- Které pixely v rastru 'rozsvítit'?



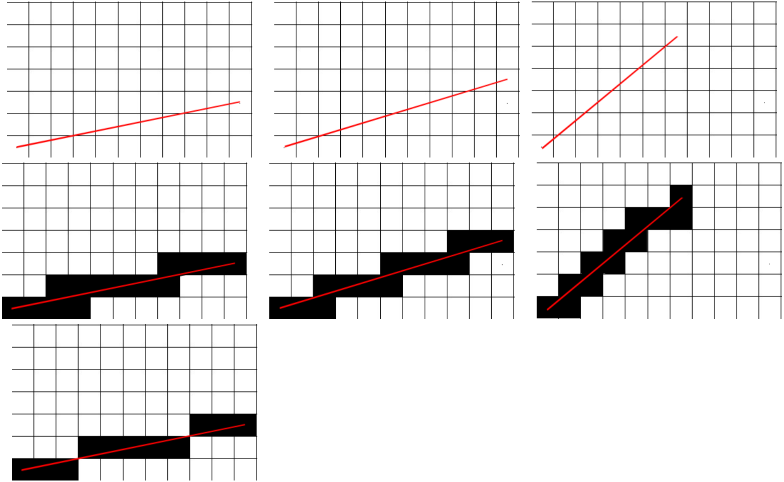
2D objekty

- Které pixely v rastru 'rozsvítit'?



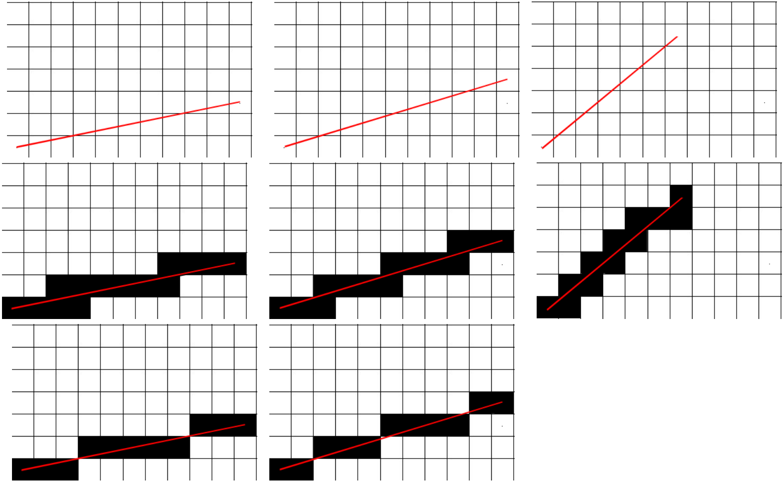
2D objekty

- Které pixely v rastru 'rozsvítit'?



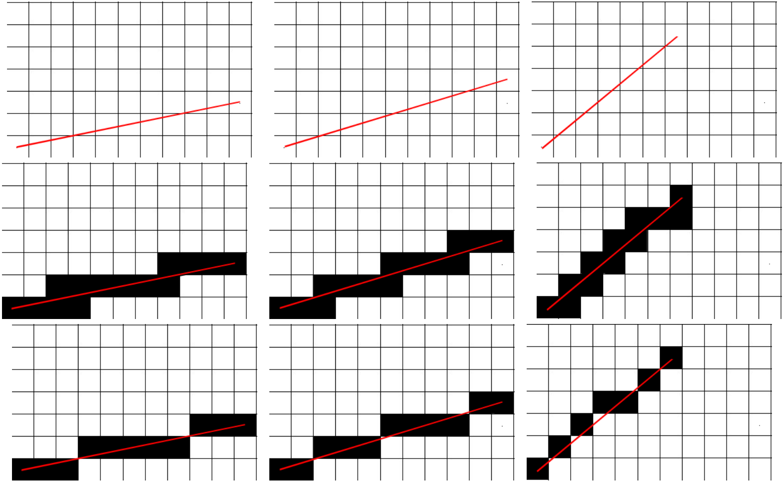
2D objekty

- Které pixely v rastru 'rozsvítit'?



2D objekty

- Které pixely v rastru 'rozsvítit'?

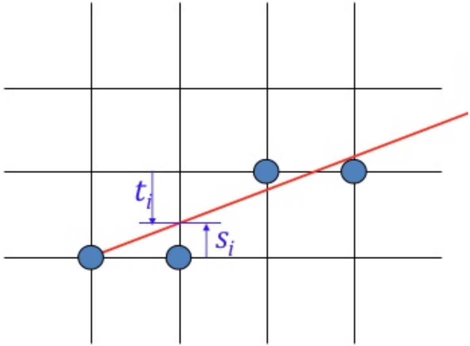


Bresenhamův algoritmus

- To, který pixel rastru vykreslit je potřeba vyřešit efektivně!
- A efektivně nejen na úrovni logiky:
  - dělení je náročná operace
  - a co teprve dělení neceločíselné
  - a co přesnost?
- Rychlý algoritmus, který je navíc celočíselný!

Bresenhamův algoritmus

- Nachází body nejbíže kreslené přimce a to pomocí *celočíselné aritmetiky*
- Funguje symetricky k určité řídící ose
- Rozhoduje, jestli je další pixel na stejné, nebo vyšší úrovni

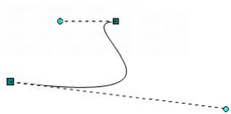
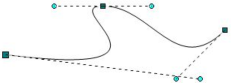
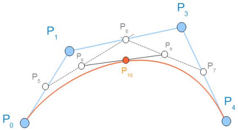


Bresenhamův algoritmus

```
var d = 2*dy - dx
var dy = y2-y1; var dx = x2-x1
var x = x1 ; var y = y1
while (x <= x2)
 {
 Draw pixel at (x,y)
 x++
 if( d<0 )
   d += dy + dy
 else
   {
   d += 2*(dy-dx)
   y++
   }
 }
```

2D objekty

- Kružnice a elipsy
  - obdobný problém jako u přímek
  - výhody ve větší symetrii
- Křivky
  - i křivky mohou být 3D!
  - rozdíl mezi 'křivkou' a 'náhodnou čarou'
- Polynomiální křivky
- Spline křivky
- Beziérovy křivky
- NURBS

Spojitost

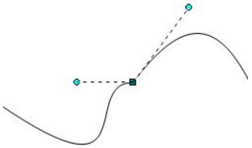
- Spojitost
  - očekávaný tvar 'budoucí' křivky na základě 'minulé'
  - stejný tečný vektor i jeho velikost
  - C^m parametricá spojitost stupně m
  - G^m geometrická spojitost stupně m

Spojitost

- C^0: Spojité napojení. Koncový bod první křivky a počáteční bod druhé křivky jsou totožné.
  Animace pohybu: animovaný objekt nezmění skokem *polohu*.
- C^1: Tečné vektory (první derivace) jsou si v bodě spojení rovny.
  Animace pohybu: animovaný objekt nezmění skokem *směr*.
- C^2: Druhé derivace v bodě spojení jsou si rovny.
  Animace pohybu: animovaný objekt nezmění skokem *rychlost*.

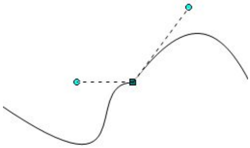
Spojitost

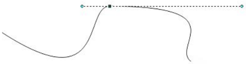
- C^0: Spojité napojení. Koncový bod první křivky a počáteční bod druhé křivky jsou totožné.
  Animace pohybu: animovaný objekt nezmění skokem *polohu*.
- C^1: Tečné vektory (první derivace) jsou si v bodě spojení rovny.
  Animace pohybu: animovaný objekt nezmění skokem *směr*.
- C^2: Druhé derivace v bodě spojení jsou si rovny.
  Animace pohybu: animovaný objekt nezmění skokem *rychlost*.



Spojitost

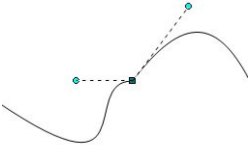
- C^0: Spojité napojení. Koncový bod první křivky a počáteční bod druhé křivky jsou totožné.
  Animace pohybu: animovaný objekt nezmění skokem *polohu*.
- C^1: Tečné vektory (první derivace) jsou si v bodě spojení rovny.
  Animace pohybu: animovaný objekt nezmění skokem *směr*.
- C^2: Druhé derivace v bodě spojení jsou si rovny.
  Animace pohybu: animovaný objekt nezmění skokem *rychlost*.

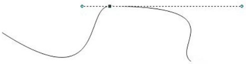


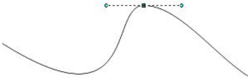


Spojitost

- C^0: Spojité napojení. Koncový bod první křivky a počáteční bod druhé křivky jsou totožné.
  Animace pohybu: animovaný objekt nezmění skokem *polohu*.
- C^1: Tečné vektory (první derivace) jsou si v bodě spojení rovny.
  Animace pohybu: animovaný objekt nezmění skokem *směr*.
- C^2: Druhé derivace v bodě spojení jsou si rovny.
  Animace pohybu: animovaný objekt nezmění skokem *rychlost*.

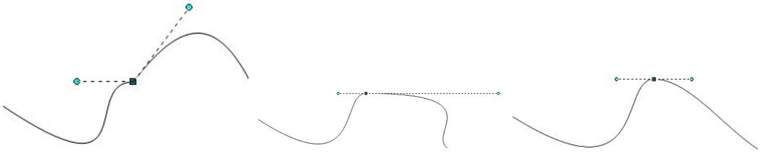






Spojitost

- C^0: Spojité napojení. Koncový bod první křivky a počáteční bod druhé křivky jsou totožné.
  Animace pohybu: animovaný objekt nezmění skokem *polohu*.
- C^1: Tečné vektory (první derivace) jsou si v bodě spojení rovny.
  Animace pohybu: animovaný objekt nezmění skokem *směr*.
- C^2: Druhé derivace v bodě spojení jsou si rovny.
  Animace pohybu: animovaný objekt nezmění skokem *rychlost*.



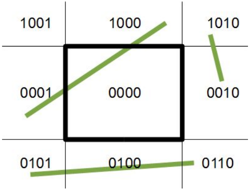
- G^0 - počáteční a koncový bod je totožný
- G^1 - tečné vektory jsou lineárně závislé

2D objekty

- Vyplňování hranice
  - řádkové se seznamem hran
  - semínkový algoritmus
- Pozor na spojitost v rastru
  - 4-okolí (návaznost při posunu v ose x nebo y)
  - 8-okolí (návazost i diagonálně)
- Hranice $\times$ vnitřek
  - Obojí 4-okolí?
  - Obojí 8-okolí?
  - Hranice 8-okolí a vnitřek 4-okolí?
  - Hranice 4-okolí a vnitřek 8-okolí?

2D objekty

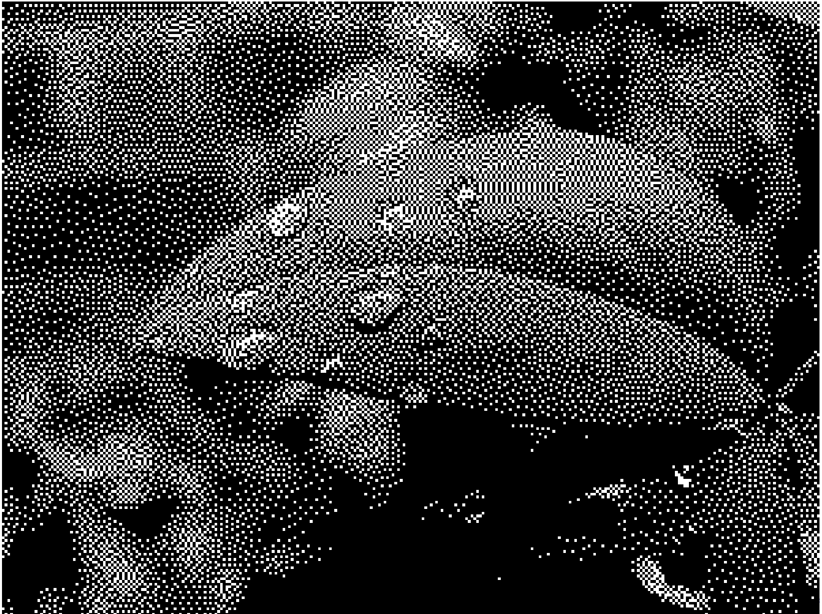
- Ořezávání useček
  - Cohen-Sutherland
- Koncové body p_1 a p_0 mají bitový kód - p_1 or $p_0 = 0 \Rightarrow$ úsečka v okně
  - p_1 and $p_0 \neq 0 \Rightarrow$ úsečka celá mimo
  - jinak $\Rightarrow$ úsečka prochází několika oblastmi
- jinak: Zkrácení úsečky
  - podle bitového kódu zvolíma další přímku, spočteme průsečík
  - opakovat dokud úsečka není celá v okně



Transformace barev

- Změna barevné charakteristiky
  - jas, kontrast, gama
  - úprava k tisku nebo zobrazení
- Konstantní prah
  - informace jen o hodnote barvy
- Náhodné rozptýlení
  - informace jen o hodnote barvy
- Pravidelné (maticové) rozptýlení
  - vyžaduje i souřadnice vstupního pixelu
- Distribuce zaokrouhlovací chyby
  - pixely je nutno navíc zpracovat v určitem pořadí (řádky, ...)

Transformace barev - motivace



Transformace barev

- Konstantní práh
  - pevný převod do (obvykle značně) omezené palety barev
  - paleta buď rovnoměrně rozdělená, nebo optimalizovaná
  - často například u tisku posterů, plakátů

Transformace barev

- Konstantní prah
  - pevný převod do (obvykle značně) omezené palety barev
  - paleta buď rovnoměrně rozdělená, nebo optimalizovaná
  - často například u tisku posterů, plakátů

Transformace barev

- Konstantní prah
  - pevný převod do (obvykle značně) omezené palety barev
  - paleta buď rovnoměrně rozdělená, nebo optimalizovaná
  - často například u tisku posterů, plakátů



Transformace barev

- Náhodné rozptýlení
  - intenzita zdrojové černé $\Rightarrow$ pravděpodobnost

Transformace barev

- Náhodné rozptýlení
  - intenzita zdrojové černé $\Rightarrow$ pravděpodobnost

Transformace barev

- Maticové rozptýlení
  - pixel nestojí sám o sobě, převádí se malé oblasti
  - konkrétních rozptylovacích matic je mnoho

Transformace barev

- Maticové rozptýlení
  - pixel nestojí sám o sobě, převádí se malé oblasti
  - konkrétních rozptylovacích matic je mnoho

Transformace barev

- Maticové rozptýlení
  - pixel nestojí sám o sobě, převádí se malé oblasti
  - konkrétních rozptylovacích matic je mnoho



Transformace barev

- Maticové rozptýlení
  - pixel nestojí sám o sobě, převádí se malé oblasti
  - konkrétních rozptylovacích matic je mnoho



$$C_{in} = \{1,2,3,4,5\} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 3 \\ 2 & 1 \end{bmatrix}$$

A jak by fungoval převod do ASCII?

Transformace barev

- Maticové rozptýlení
  - pixel nestojí sám o sobě, převádí se malé oblasti
  - konkrétních rozptylovacích matic je mnoho

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

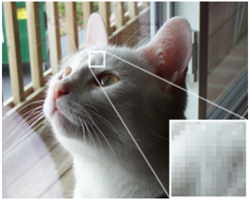
$$M_{tisk} = \begin{bmatrix} 1 & 5 & 9 & 2 \\ 8 & 12 & 13 & 6 \\ 4 & 15 & 14 & 10 \\ 0 & 11 & 7 & 3 \end{bmatrix}$$

Transformace barev

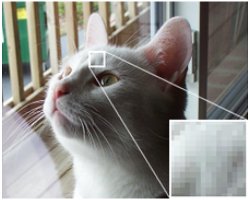
- Distribuce zaokrouhlovací chyby
  - pixel nestojí sám o sobě
  - při převodu do omezené palety vnikne chyba
  - ta se *rozdistribuuje* do ještě nepřevedených pixelů v okolí
- Floyd–Steinberg dithering - distribude chyby + optimalizované paleta

$$\begin{bmatrix} & & \star & \frac{7}{16} & \dots \\ \dots & \frac{3}{16} & \frac{5}{16} & \frac{1}{16} & \dots \end{bmatrix}$$

Transformace barev

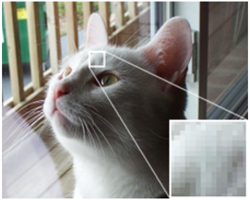
Originál	8bit prah.	8bit distr.
		

Transformace barev

Originál	8bit prah.	8bit distr.
		

Transformace barev

Originál

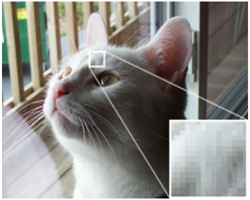


8bit prah.

8bit distr.

Transformace barev

Originál

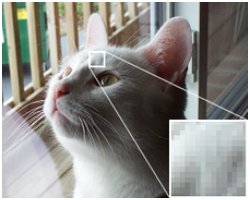


8bit prah.

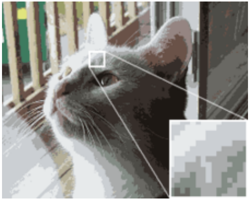
8bit distr.

Transformace barev

Originál 8bit prah. 8bit distr.

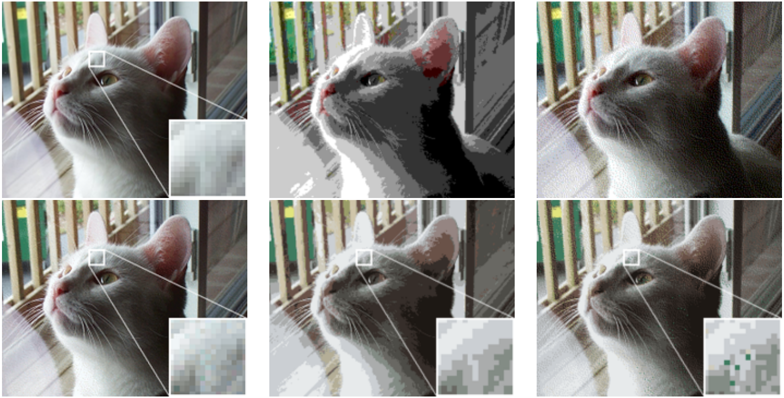






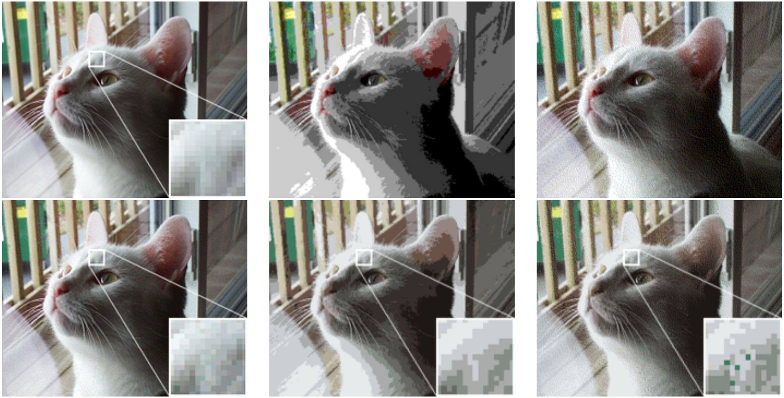
Transformace barev

Originál | 8bit prah. | 8bit distr.



Transformace barev

Originál 8bit prah. 8bit distr.



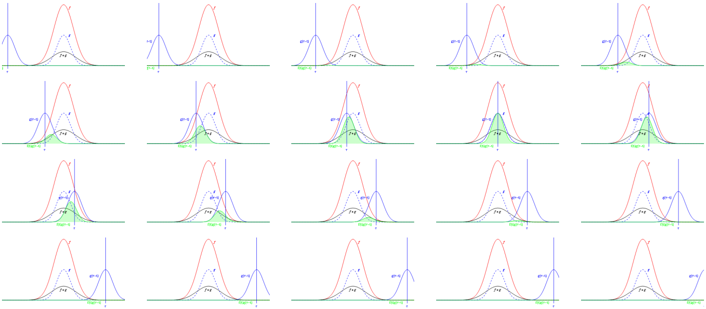
8bit opt + dith. 16b paleta 16b paleta + dith.

Konvoluce

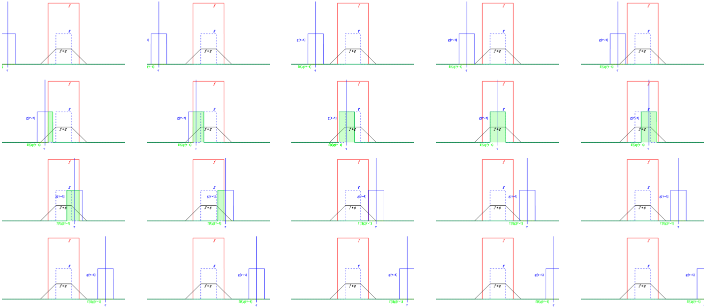
$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{1}$$

$$* : L_1 \times L_1 \longrightarrow L_1 \tag{2}$$

Spojitá konvoluce



Spojitá konvoluce



Konvoluce

Základní vlastnosti

$$f * g = g * f \tag{3}$$

$$f * (g * h) = (f * g) * h \tag{4}$$

$$a(f * g) = (af) * g = f * (ag) \tag{5}$$

$$f * (g + h) = (f * g) + (f * h) \tag{6}$$

$$f * \delta = \delta * f = f \tag{7}$$

Konvoluce

$$\delta(x) = \begin{cases} \infty, & \text{if } x = 0. \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

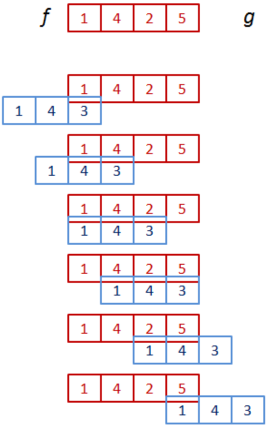
$$\int \delta(x)dx = 1 \tag{9}$$

$$\int_{-\infty}^{\infty} f(x)\delta(x-a)dx = f(a) \tag{10}$$

Diskrétní Konvoluce

- Diskrétní konvoluce (rastr)
- 1D

$$(f * g) = \sum_{m=-\infty}^{\infty} f[x] \cdot g[n-m] \tag{11}$$



$c = f * g$

C[0] = 1*3 = **3**

C[1] = 1*4 + 4*3 = **16**

C[2] = 1*1 + 4*4 + 2*3 = **23**

C[3] = 4*1 + 2*4 + 5*3 = **27**

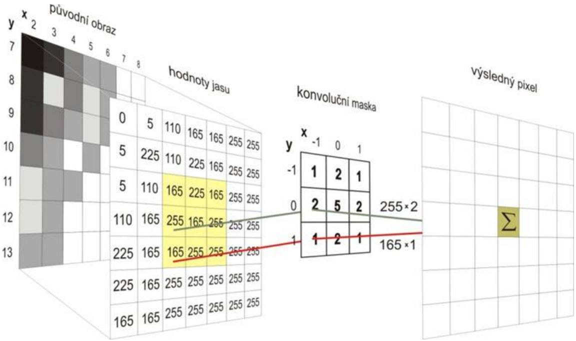
C[4] = 2*1 + 5*4 = **22**

C[5] = 5*1 = **5**

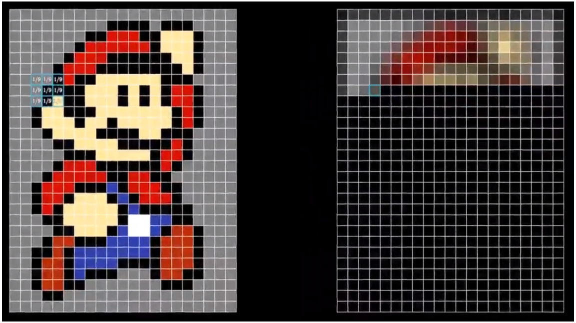
Diskrétní konvoluce

- Diskrétní konvoluce (rastr)
- 2D (obrázky)

$$(f * g) = \sum_{m=-\infty}^{\infty} f[x] \cdot g[n-m] \tag{12}$$



Diskrétní konvoluce



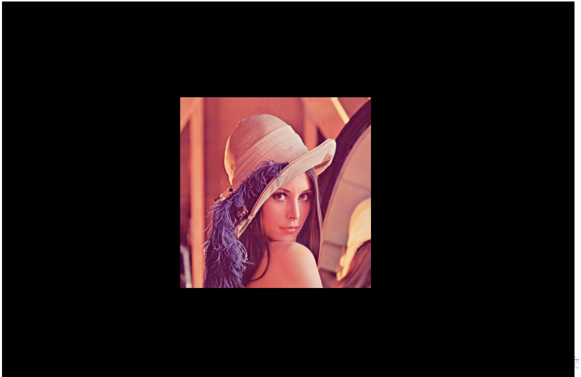
video

Okrajový jev

- Diskrétní konvoluce (rastr)
- 2D (obrázky)
- Okrajový jev: A co na kraji obrazu?
  - type valid
  - type same
  - type full

Okrajový jev

- Diskrétní konvoluce (rastr)
- 2D (obrázky)
- Okrajový jev: A co na kraji obrazu?
  - oblepení nulami (zero padding)
  - zrcadlové prodloužení (mirror extension)
  - periodické prodloužení (periodic/toroid extension)



Okrajový jev

- Diskrétní konvoluce (rastr)
- 2D (obrázky)
- Okrajový jev: A co na kraji obrazu?
  - oblepení nulami (zero padding)
  - zrcadlové prodloužení (mirror extension)
  - periodické prodloužení (periodic/toroid extension)

Okrajový jev

- Diskrétní konvoluce (rastr)
- 2D (obrázky)
- Okrajový jev: A co na kraji obrazu?
  - oblepení nulami (zero padding)
  - zrcadlové prodloužení (mirror extension)
  - periodické prodloužení (periodic/toroid extension)

Konvoluční filtry

- Prosté a vážené průměrování
- Odšumění
- Detekce hran
- Ostření
- Filtry ve frekvenční oblasti (příště)

Odšumění

- Prosté a vážené průměrování
- Odšumění
- Součet roven 1
  - proč?

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Odšumění

- Prosté a vážené růměrování
- Gaussův šum
- Součet roven 1
  - proč?

$$\frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Detekce hran

- Sobelův operátor
  - první derivace
  - směrově orientovaný
  - dvojice masek h a h
- Součet roven 0
  - Proč?

$$h = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \overline{h} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$|G| = \sqrt{h^2 + \overline{h}^2}$$

Detekce hran

- Sobelův operátor
  - první derivace
  - směrově orientovaný
  - dvojice masek h a $\overline{h}$

$$h = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \overline{h} = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$$

$$|G| = \sqrt{h^2 + \overline{h^2}}$$

Detekce hran

- Robinsonův operátor
- Kirschův operátor

$$h_{Robin} = \begin{bmatrix} -1 & -1 & -1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix} h_{Kirsch} = \begin{bmatrix} -5 & -5 & -5 \\ 3 & 0 & 3 \\ 3 & 3 & 3 \end{bmatrix}$$

$$|G| = \sqrt{h^2 + \overline{h^2}}$$

Detekce hran

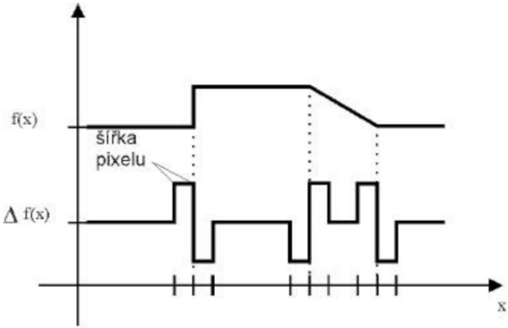
- Laplaceuv operátor
  - druhá derivace
  - dvojí reakce na hranu

$$\lambda = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \lambda = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

více: [Gonz89][Gonz97][Lewi90]

Detekce hran

- Laplaceuv operátor
  - druhá derivace
  - dvojí reakce na hranu



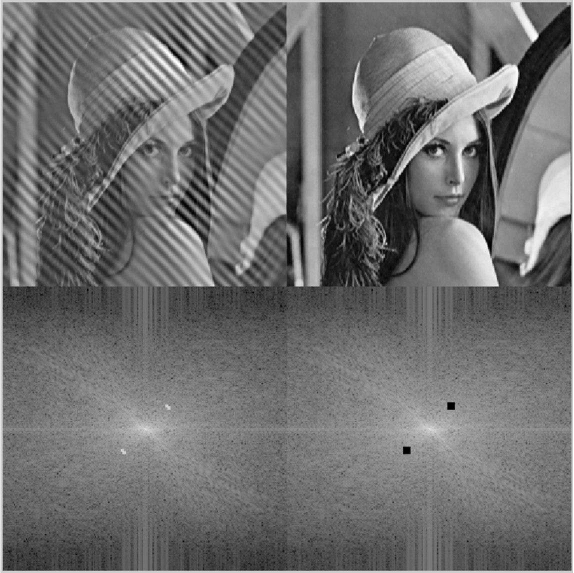
Ostření

- Zaostření pomocí detekce hran
- Vysoké frekvence
- Nízké frekvence

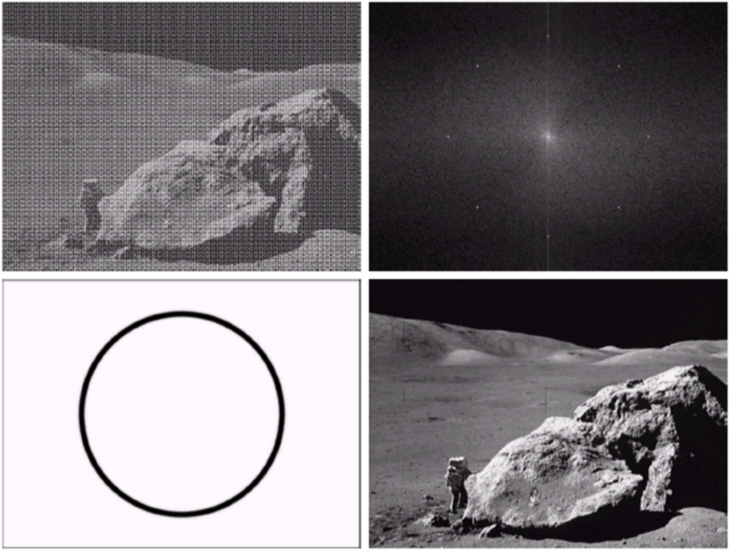
$$\lambda = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \lambda = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$|G| = \sqrt{h^2 + \overline{h^2}}$$

Příště



Příště



Zdroje

...