Výtvarné zpracování rastrových obrazů

Pro popis transformací je použita následující notace:

÷, div **- operace celočíselného dělení**
%, mod **- operace zbytek po celočíselném dělení**
W **- šířka obrazu**
H **- výška obrazu**
R **- maximální poloměr polárních souřadnic obrázku**

$$R = \sqrt{(W \div 2)^2 + (H \div 2)^2}$$

[x, y] **- kartézské souřadnice bodu obrazu**

$$x \in <0; W-1>, y \in <0; H-1>$$

[r,a] **- polární souřadnice bodu**
Old[x,y] **- barva původního obrazu v bodě o souřadnicích** *[x,y]*
New[x,y] **- barva nového obrazu v bodě o souřadnicích** *[x,y]*
xxx[x,y].a **- hodnota barevné složky** *a* **obrazu** *xxx* **v bodě o souřadnicích** *[x,y]*
- složka a může být *R* **– červená,** *G* **– zelená,** *B* **– modrá**
Rand(x) **- celé náhodné číslo z intervalu** *<0;x)*
New[x,y, w,h] **- okolí bodu** *[x,y]* **o šířce** *w* **a výšce** *h* **v původním obrázku**
Old[x,y, w,h] **- okolí bodu** *[x,y]* **o šířce** *w* **a výšce** *h* **ve výsledném obrázku**
cond ? a : b **- platí-li** *cond*, **má výraz hodnotu** *a*, **jinak** *b*

1 Jednoduché geometrické transformace

Geometrické transformace jsou operace, jež v podstatě nahlížejí na rastrový obraz jako na množinu bodů v rovině (v ortogonálním souřadném systému) a nad nimi realizují geometrické operace: posunutí, otočení a změnu měřítka. Tyto operace jsou prováděny buďto s konstantními koeficienty nebo s koeficienty proměnnými dle polohy bodu. U těchto operací nezáleží na barevné hloubce pixelu zpracovávaného obrázku a často se liší rozměry zdrojového a cílového obrazu.

1.1 Překlopení obrazu (prohození horizontální nebo vertikální)

Provedení operace spočívá v prohození dvou stejně vzdálených pixelů od osy vedené středem obrázku. Podle polohy osy pak dostaneme prohození horizontální nebo vertikální.

Horizontální a vertikální prohození

1.2 Otočení obrazu

Transformační rovnice $New[x, y] = Old[x.\cos(\alpha) - y.\sin(\alpha); x.\sin(\alpha) - y.\cos(\alpha)]$

Podle zadaného úhlu otočení je třeba vypočítat souřadnice rohů obrázku a z nich určit novou šířku a výšku. Díky tomu, že pracujeme v rastru a výsledné hodnoty po transformaci převádíme na celá čísla, došlo k nechtěnému vytvoření vzorku z bodů, do nichž během transformace otočení nebyly žádné původní pixely převedeny. Proto je nutná tzv. inverzní transformace.

nežádoucí vzorek

Otočení obrázku o 30° podle transformační rovnice a s použitím zpětné rotace

1.3 Zrcadlení

Po umístění osy a zvolení části obrázku, která se má zrcadlit, je vytvořen nový obrázek dvojnásobné velikosti zvolené části podle směru osy. Procházíme vybranou částí obrázku a nové pixely umísťujeme souměrně podle osy.

horizontální	**vertikální**
$x \in < 0; W-1 > \rightarrow New[x,y] = Old[x,y]$	$y \in < 0; H-1 > \rightarrow New[x,y] = Old[x,y]$
$x \in < W, 2W-1 > \rightarrow New[x,y] = Old[2W-x,y]$	$y \in < H, 2H-1 > \rightarrow New[x,y] = Old[x,2H-y]$

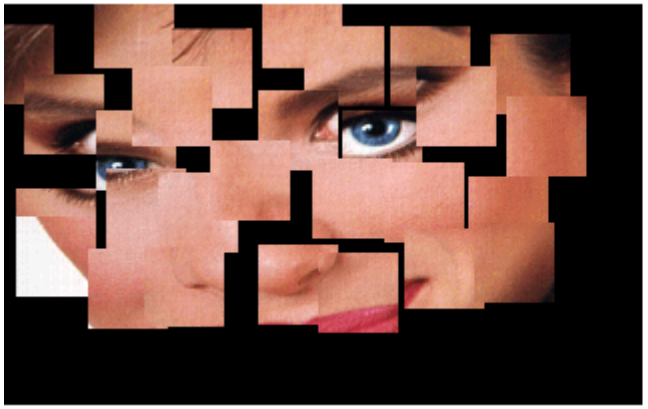
Původní tvář a její složenina z levé a pravé poloviny obličeje

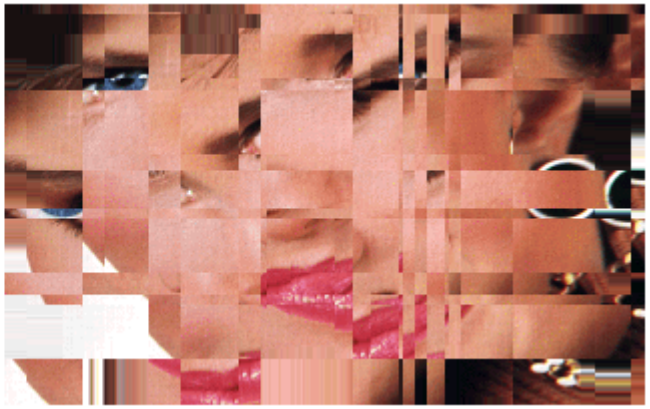
1.4 Náhodné posouvání stejných částí obrazu

Zajímavý vzhled rozpadajícího se obrazu můžeme dostat následující operací: obraz rozdělíme na stejné části – bloky a ty posuneme v libovolném směru o náhodnou vzdálenost.

$$New[x+dx, y+dy] = Old[x, y]$$
$$dx = Rand(a); dy = Rand(a)$$

Hodnoty *dx* **a** *dy* **jsou konstantní pro každou zvolenou pravoúhlou oblast. Pro zachování dobré rozpoznatelnosti obrazu je dobré, aby délka posunutí** *a* **byla menší než velikost segmentu. Podobné je náhodné rozhození obrazu. Transformační rovnice mají stejný tvar jako minulé rovnice, ale tentokrát je velikost bloku náhodná a pozadí je bílé.**

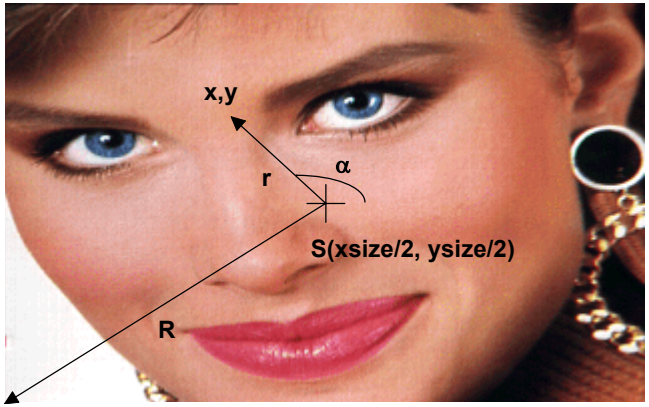




Náhodné posouvání a rozhození bloků

1.5 Vír

Zvolíme úhel otočení α, který je mírou víru. Poté procházíme jednotlivé souřadnice *x* a *y* nového obrázku a provádíme jejich převod do polárních souřadnic s počátkem ve středu obrázku.



Zavedení polárních souřadnic v rastrovém obrázku

S takto získanými polárními souřadnicemi r **a** α **provedeme transformaci otočení o úhel, který závisí na velikosti poloměru** r**. Poté se provede přepočet zpátky do souřadnic** (x_n, y_n)**. Pixelu na souřadnicích** (x, y) **je pak přiřazena barva bodu** (x_n, y_n)**. Nejjednodušší možností je lineární závislost úhlu na velikosti *r*. Z toho vyplývají dvě varianty provedení efektu. Úhel otočení se s rostoucím *r* zvětšuje nebo zmenšuje.**

V prvním případě je stočení obrázku největší v jeho rozích, ve středu obrázku přitom k pohybu nedochází, ve druhém je tomu naopak.

Efekt víru se zmenšujícím se úhlem otočení s rostoucím *r* (otočení o 60° a 360°)

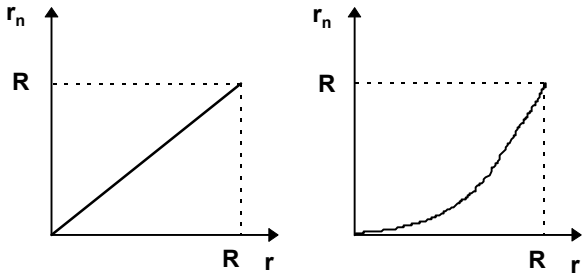
Efekt víru se zvětšujícím se úhlem otočení s rostoucím *r* (otočení o 60° a 360°)

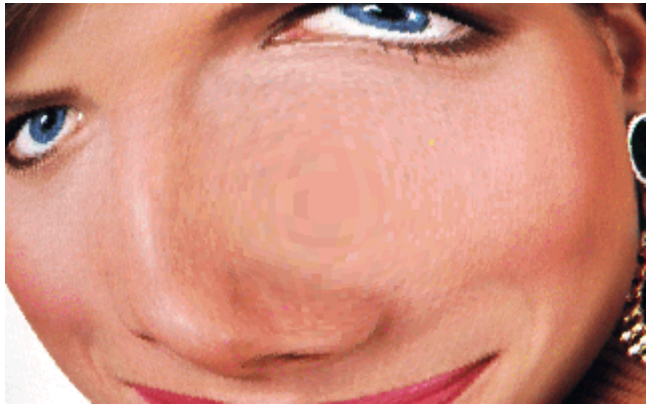
1.6 Stlačení obrazu ven (vyboulení)

Použijeme opět polárních souřadnic, kdy při následné zpětné transformaci na souřadnice kartézské se hodnota *r* mění podle zadané nelineární funkce:

$$r_n = f(r)$$

Př: Kvadratická funkce

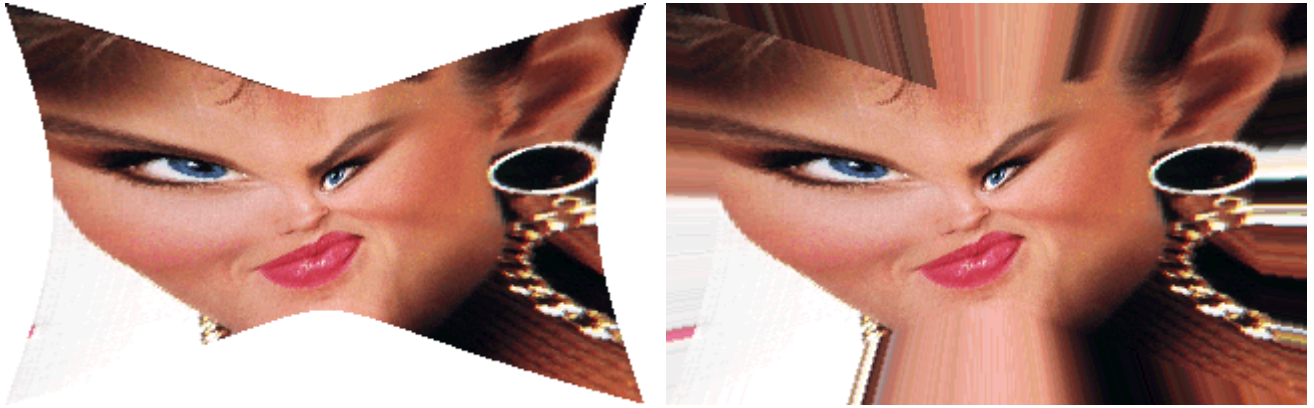




Efekt stlačení ven s použitím kvadratické funkce

1.7 Stlačení obrazu dovnitř (vmáčknutí)

Pro analogii efektů použijeme druhou odmocninu. Při zpětné transformaci mohou vyjít získané kartézské souřadnice mimo původní obrázek. Tyto pixely nastavíme na barvu pozadí, nebo jim přiřadíme barvu bodů na nejbližším okraji obrázku.



Efekt stlačení dovnitř s použitím druhé odmocniny

1.8 Zvlnění obrazu

Efekt využívá generátoru náhodných čísel, který rozhoduje o posunu zpracovávaného pixelu ve svislém nebo vodorovném směru. Nejprve se vypočítá pole svislých posuvů pro jeden řádek: Nastaví se startovací hodnota odchylky D_y a při průchodu polem se podle hodnoty náhodného čísla k této odchylce přičte nebo odečte jednička. Zároveň se takto upravená odchylka uloží do pole. S připraveným polem hodnot se pak provede posuv ve svislém směru pro každý řádek obrázku a zároveň se ve zpracovávaném řádku podle náhodného čísla realizuje horizontální posuv. V případě, že nové souřadnice pixelu padnou mimo meze obrazu, je barva pixelu nastavena na zvolenou hodnotu.

Zvlnění obrázku

1.9 Efekt koupelnového skla

Transformace vytváří dojem pohledu přes tvarované sklo. Parametr q určuje „jemnost“ efektu

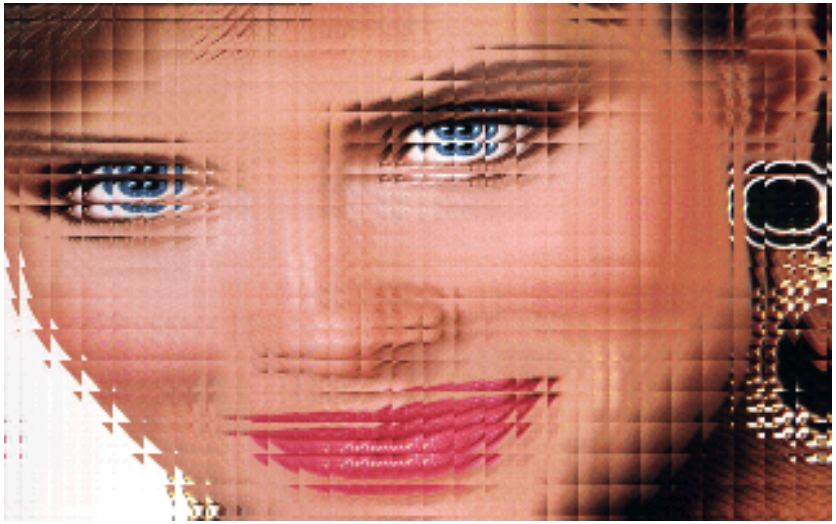
$$New[x, y] = Old[x + (x\%q) - (q \div 2), y]$$

Efekt koupelnového skla

1.10 Efekt fazetového skla

Tento efekt je parametrizován velikostí „oka sítě“ *N*.

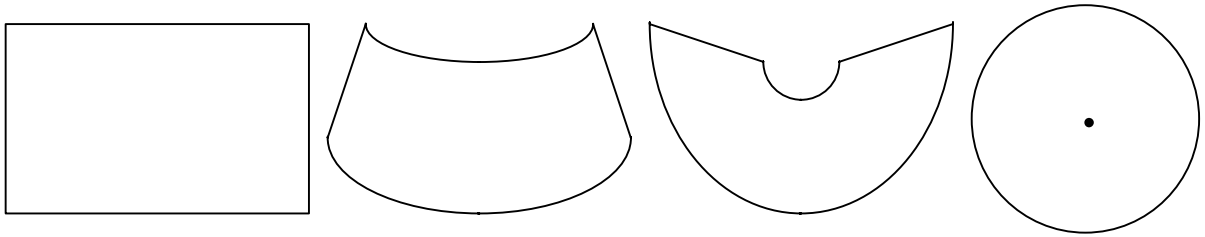
$$New[x, y] = Old[x - (N \div 2) + (x\%N), y - (N \div 2) + (y\%N)]$$



Efekt fazetového skla

1.11 Stočení obrazu do kruhu

Př. Mapovací funkce pro různé úhly výseče



Totální deformace rastrového obrázku do kruhu směrem nahoru a dolů

1.12 Sprej

Nejprve se definují rozměry pracovní obdélníkové oblasti, a potom následuje průchod celým obrázkem pixel po pixelu. Z obdélníkové oblasti, jejíž levý horní roh je určen aktuálním pixelem, se pomocí generátoru náhodných čísel vybere další pixel a barvy těchto dvou pixelů se prohodí.

Efekt spreje s nastavenou oblastí 5x5 a 20x20

1.13 Vepsání obrazu do kruhu

Střed kruhu je ve středu původního obrázku a jednotlivé strany přejdou v obvod kruhu.

Roztažení obrázku do kruhu

2 Barevné transformace obrazu

U těchto operací záleží na barevné hloubce pixelu zpracovávaného obrazu.

2.1 Olejomalba

Operace vytváří efekt obrazu malovaného olejovými barvami. Ve zvoleném okolí *N* zpracovávaného pixelu sestavíme histogram výskytu jednotlivých barevných odstínů a jako novou barvu tohoto pixelu pak dosadíme barvu sestavenou z nejčastěji se vyskytujících barevných složek v prozkoumaném okolí *MaxN*.

$$New[x,y,w,h].R = MaxN(Old[x,y,w,h].R)$$
$$New[x,y,w,h].G = MaxN(Old[x,y,w,h].G)$$
$$New[x,y,w,h].B = MaxN(Old[x,y,w,h].B)$$

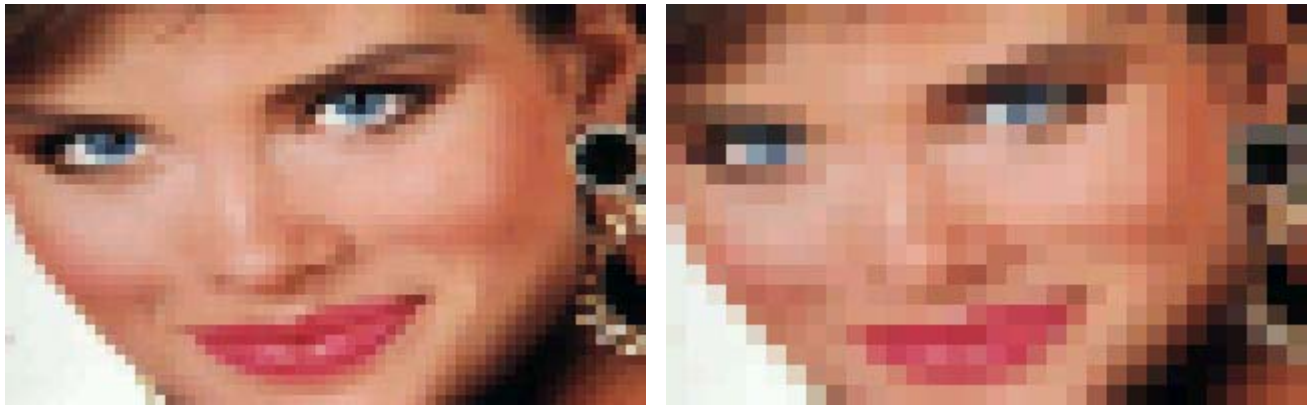
Olejová malba pro *N* = *3*

2.2 Pixelizace obrazu

Pixelizace obrazu spočívá ve výpočtu průměrného barevného odstínu na definované pravoúhlé oblasti původního obrázku. V nově vytvářeném obrázku je pak tato oblast vyplněna takto získaným barevným odstínem.

Operaci je možné modifikovat tím, že se místo průměrné hodnoty použije hodnota maximální (*Max*) nebo minimální (*Min*) hodnota z daného okolí.

$$New[x, y, w, h] = Avg(Old[x, y, w, h])$$
$$New[x, y, w, h] = Max(Old[x, y, w, h])$$
$$New[x, yw, h] = Min(Old[x, y, w, h])$$



Pixelizace obrazu na oblasti 5x5 a 10x10 pixelů

2.3 Plakátový efekt - posterizace

U plakátů je často omezen počet barevných odstínů – tzn. volíme určitý počet intenzit každé barevné složky a pro všechny body obrazu vyhledáme nejbližší hodnotu (pro každou složku zvlášť). Zvolit barvy je možné tak, že celý interval rozdělíme na žádaný počet (*n*) stejných intervalů (*d*) a z něj zvolíme nějakou hodnotu (například minimum).

$$d = 255 \div n$$
$$New[x, y].R = n.Old[x, y].R \div n$$
$$New[x, y].G = n.Old[x, y].G \div n$$
$$New[x, y].B = n.Old[x, y].B \div n$$

Posterizace

2.4 Negativ, solarizace a prahování

Hodnotu příslušné barevné složky nahradíme jejím doplňkem do maximální hodnoty. Uvažovaná maximální hodnota složek *R, G, B* je rovna *255*. Negaci je např. možné provést jen u některých barevných složek, tak docílíme dalších efektů.

Operace ***solarizace* je modifikací negace obrazu. Pokud intenzita šedi pixelu přesáhne určitou hranici závislou na poloze pixelu v obrázku, provedeme negaci tohoto pixelu.**

Efekt ***prahování* je založen na rozdělení barev na dvě skupiny podle toho, zda je jejich hodnota větší nebo menší než daná hodnota *(práh – pR, pG, pB)*. Hodnota výsledného pixelu se nastaví na hodnotu extrému (*0* nebo *255*), který do intervalu patří. Tato operace se provádí pro každou barevnou složku zvlášť.**

$$New[x,y].R = Old[x,y].R \geq pR\,?\,255:0$$
$$New[x,y].G = Old[x,y].G \geq pG\,?\,255:0$$
$$New[x,y].B = Old[x,y].B \geq pB\,?\,255:0$$

Negativ, prahování a solarizace

2.5 Rozpouštění barev

Od náhodně vygenerované pozice uvnitř obrázku se postupuje svisle dolů a testují se vždy dva po sobě následující pixely. Pokud je odstín šedi horního pixelu tmavší, barvy obou pixelů se prohodí a provádí se porovnání následujících dvou bodů. V případě, že je odstín vrchního pixelu světlejší, vygeneruje se nová náhodná pozice a postup se opakuje. Počet opakování je většinou dán počtem pixelů v obrázku. Efekt je možné umocnit malou změnou algoritmu. Pokud nastane situace, že odstín šedi horního pixelu je tmavší, nedojde k prohození barvy obou pixelů, ale provede se rotace barev ve svislém směru od dané pozice až k dolnímu okraji obrázku.

Normální a silnější rozpouštění obrázku

3 Homogenní barevné souřadnice

Řady zajímavých efektů můžeme docílit aplikací barevných homogenních souřadnic. Ty získáme tak, že barvě pixelu určené barevnými složkami *(R, G, B)* **přiřadíme čtveřici** *(Rw, Gw, Bw, w)***, kde** $w \neq 0$**. Nejčastěji bude** $w = 1$**. Podobně jako u geometrických transformací lze takto definované operace nad obrazy skládat pomocí standardního násobení matic. V příkladech jsou uvažovány rozsahy barevných složek** *R, G* **a** *B* **v intervalu** *<0, 255>***. Obecně se pak dá efekt z této kategorie popsat následující rovnicí:**

$$\begin{bmatrix} New[x,y].R \\ New[x,y].G \\ New[x,y].B \\ x \end{bmatrix} = \begin{bmatrix} a_{11} & . & . & a_{14} \\ . & . & . & . \\ . & . & . & . \\ a_{41} & . & . & a_{44} \end{bmatrix} \times \begin{bmatrix} Old[x,y[.R \\ Old[x,y[.G \\ Old[x,y[.B \\ 1 \end{bmatrix}$$

3.1 Matice pro změnu jasu

Změnu jasu v obrazu lze vyjádřit maticí

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ R_{posun} & G_{posun} & B_{posun} & 1 \end{pmatrix}$$

Ke skutečné změně jasu dojde pouze v případě, že všechny uvedené konstanty mají stejnou hodnotu. Pokud se hodnoty navzájem liší, dochází k různému ovlivnění barevných složek, čímž lze dosáhnout dalších efektů.

Př.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 120 & 120 & 120 & 1 \end{pmatrix}$$

Změna jasu obrázku podle zvolené matice

3.2 Matice pro změnu měřítka barev

Podobně jako při prostorových transformacích lze provádět i změnu velikosti jednotlivých barevných složek pomocí známé měřítkové matice:

$$\begin{pmatrix} R_{rozsah} & 0 & 0 & 0 \\ 0 & G_{rozsah} & 0 & 0 \\ 0 & 0 & B_{rozsah} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

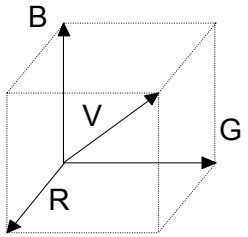
Př.

$$\begin{pmatrix} 1.2 & 0 & 0 & 0 \\ 0 & 0.6 & 0 & 0 \\ 0 & 0 & 1.1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Změna měřítka barev obrázku podle zvolené matice

3.3 Matice pro otočení barev

Analogicky k transformaci otáčení lze provádět rotaci barev podle některé z barevných os *R, G* **nebo** *B*. **Složením základních transformací a jejich inverzí můžeme rotovat kolem libovolného barevného vektoru. Jako příklad uveďme otočení kolem vektoru šedé** *V* **o** *120°*, **tj. barevný přechod** *(R, G, B)* $\rightarrow$ *(B, R, G)*.



$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Otočení barev kolem vektoru šedé o 120°

3.4 Matice pro převod barev na stupně šedi

Matice pro převod barevného obrazu na odstíny šedi:

$$\begin{pmatrix} R_{váha} & R_{váha} & R_{váha} & 0 \\ G_{váha} & G_{váha} & G_{váha} & 0 \\ B_{váha} & B_{váha} & B_{váha} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Hodnoty konstant podle standardu NTSC:

$R_{váha} = 0.299, \qquad G_{váha} = 0.587, \qquad B_{váha} = 0.114$

Hodnoty konstant podle standardu CIE (lepší):

$R_{váha} = 0.3086, \quad G_{váha} = 0.6094, \quad B_{váha} = 0.0820$

Převod obrázku na stupně šedi pomocí matice

3.5 Matice pro změnu sytosti barev

Sytost vlastně určuje množství příměsi jiných barev. Složitější cesta, jak změnit hodnotu sytosti vede přes převody barevných modelů HSV ⇔ RGB nebo HLS ⇔ RGB.

Matice pro změnu sytosti barev:

$$\begin{pmatrix} A & B & C & 0 \\ D & E & F & 0 \\ G & H & I & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Použité konstanty jsou definovány takto:

$A = (1 - s) * R_{váha} + s$ $B = (1 - s) * R_{váha}$ $C = (1 - s) * R_{váha}$

$D = (1 - s) * G_{váha}$ $E = (1 - s) * G_{váha} + s$ $F = (1 - s) * G_{váha}$

$G = (1 - s) * B_{váha}$ $H = (1 - s) * B_{váha}$ $I = (1 - s) * B_{váha} + s$

Hodnota sytosti je v intervalu $s = \langle 0, 1 \rangle$**,** $R_{váha}$, $G_{váha}$ **a** $B_{váha}$ **jsou výše definované konstanty. Dosadíme-li za** s **hodnotu** 0**, je barevný vektor převeden na odstín šedi, naopak po dosazení hodnoty** 1 **se tento vektor nezmění (jednotková matice). Tyto dva případy zvolených hodnot odpovídají umístění barvy ve středu a na obvodu jehlanu či kužele uvedených barevných modelů.**

Změna sytosti barev obrázku na hodnotu s = 0.6

4 Průsvity

4.1 Konstantní průsvit

Tento průsvit je nejjednodušší z možných průsvitových operací. Máme dva obrázky *A* a *B*, ze kterých průsvit vytváříme. Barevné hodnoty jejich pixelů označíme *(R_A, G_A, B_A)* a *(R_B, G_B, B_B)* a určíme, kolik procent *p* obrázku *A* bude vidět v obrázku *B*. Barevné hodnoty výsledného pixelu pak určíme podle vztahů

$$R = pR_A + (1-p)R_B$$
$$G = pG_A + (1-p)G_B$$
$$B = pB_A + (1-p)B_B$$

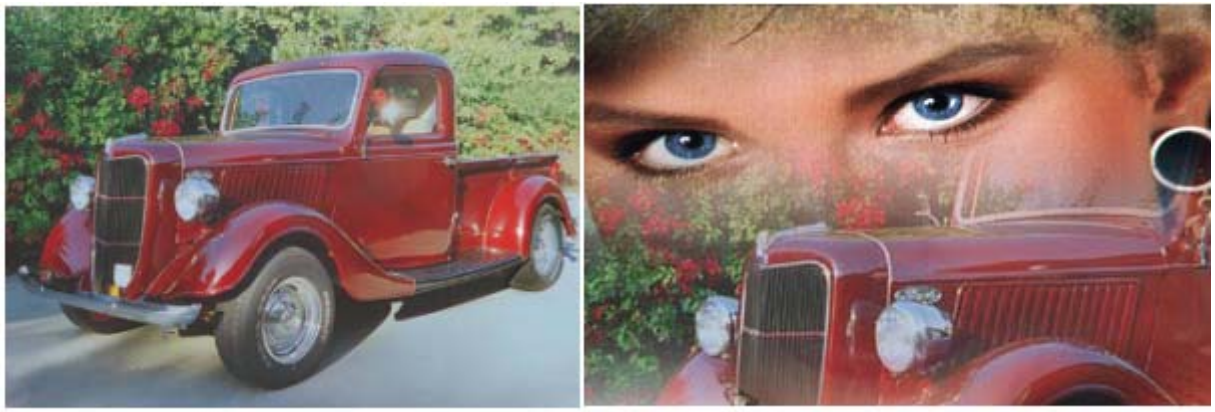
Průsvit obrázků na 50%

4.2 Průsvit řízený funkcí

Při vytváření těchto efektů se vychází z výše uvedených rovnic, procento určující míru průsvitu však není konstantní a zpravidla závisí nějakým způsobem na souřadnicích zpracovávaného pixelu.

$$p = f(x, y)$$

Lineární změna průsvitu obrázků zleva - doprava a shora - dolů

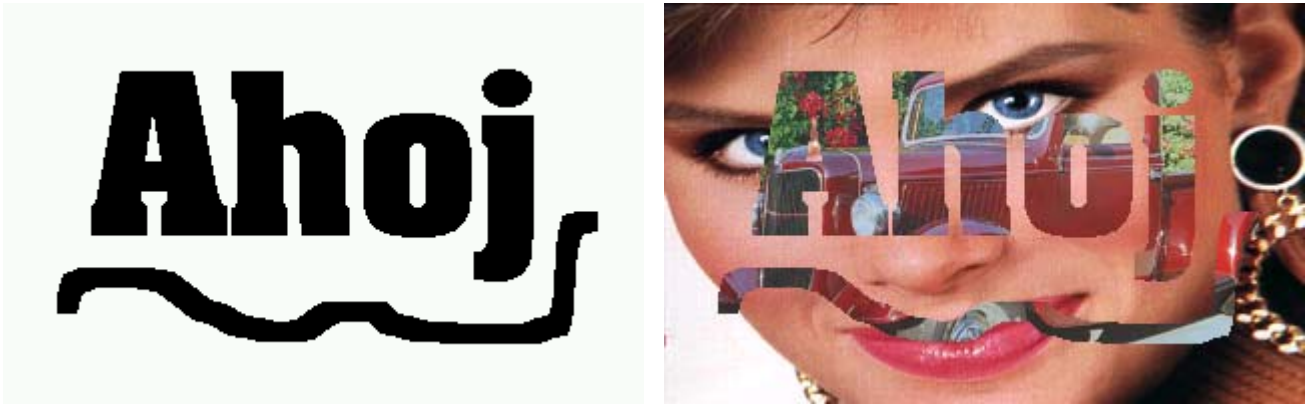


Nelineární řízení průsvitu vertikální vlnou sinusovky

4.3 Průsvit řízený maskou

Pro zcela obecné řízení průsvitu při skládání obrázků se nabízí možnost zadat funkci průsvitu pomocí tzv. masky. Masku tvoří další obrázek v odstínech šedi. Úrovně intenzit představují procenta průsvitnosti pixelu daného obrazu.

Př. Použití masky s binární průsvitností (0% nebo 100% průsvitu)



Maska a provedený průsvit obrázků

5 Filtrace obrazu

Při filtraci obrazu se používají většinou dva přístupy, prostorový a frekvenční. Uvedeme pouze prostorový přístup, jenž pracuje přímo s obrazem. Obraz transformujeme pomocí operátoru

$$g(x,y) = T\left[f(x,y)\right],$$ **kde *f(x,y)* je vstupní funkce a *g(x,y)* je výstupní funkce**

Operátor T je nejčastěji vyjádřen pomocí funkce hodnot okolí bodu - tzv. masky okolí.

Příklad masky okolí o rozměru 3x3:

	x-1	**x**	**x+1**
y-1	$h_{-1,-1}$	$h_{0,-1}$	$h_{1,-1}$
y	$h_{-1,0}$	$h_{0,0}$	$h_{1,0}$
y+1	$h_{-1,1}$	$h_{0,1}$	$h_{1,1}$

$$\begin{aligned} T\left[f(x,y)\right] = {} & h_{-1,-1}f(x-1,y-1) + h_{0,-1}f(x,y-1) + h_{1,-1}f(x+1,y-1) + \\ & + h_{-1,0}f(x-1,y) + h_{0,0}f(x,y) + h_{1,0}f(x+1,y) + \\ & + h_{-1,1}f(x-1,y+1) + h_{0,1}f(x,y+1) + h_{1,1}f(x+1,y+1) \end{aligned}$$

Jedná se vlastně o konvoluci

$$g(x,y) = \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} h(i,j)f(x+i,y+j)$$

Funkce *h* bývá označovaná jako jádro.

5.1 Rozostření

K vytvoření dojmu rozostřeného obrazu nám stačí použít obyčejné průměrování okolí bodu. Konvoluční maska 3x3 může vypadat následovně.

$$h = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Rozostření obrazu

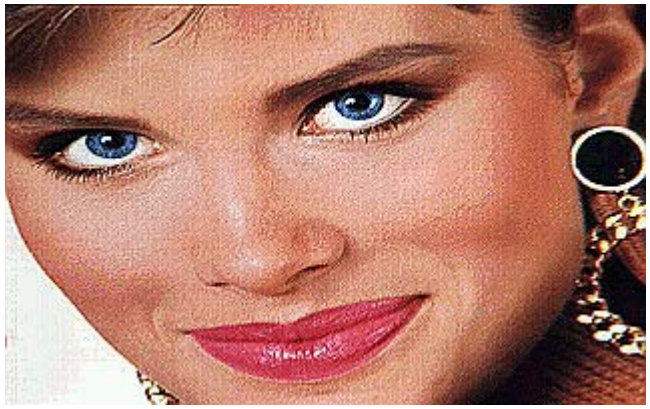
Někdy se zvětšuje váha středového bodu masky případně jeho čtyř sousedů pro přiblížení k vlastnostem Gaussovského rozložení.

$$h = \frac{1}{10}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad h = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

5.2 Zaostření

Pro zaostření obrazu se osvědčila tato konvoluční maska.

$$h = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$



Zaostření obrazu

5.3 Reliéf

Pro zobrazení reliéfu obrázku lze použít konvoluční masku

$$h = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Reliéf obrazu

Je možné použít Sobelova operátoru, který aproximuje první parciální derivaci. Ten je možné vytvořit pro různé velikosti a směry. Pro rozměr 3x3 existuje osm konvolučních masek, které se získají rotací masky první.

$$h_1 = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \quad h_2 = \begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix} \quad h_3 = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \quad h_4 = \ldots$$

Masky h_1 a h_3 mají důležitý význam. Maska h_1 přibližuje parciální derivaci ve směru x a h_3 ve směru y.

První parciální derivace ve směru x a y

5.4 Detekce hran

Pro detekci hran je vhodné použít Laplaceův gradientní operátor (Laplacián). Udává velikost hrany a nikoliv její směr, proto není citlivý na otočení. Podle typu sousednosti je možné použít dvě konvoluční masky.

$$h^4 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad h^8 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

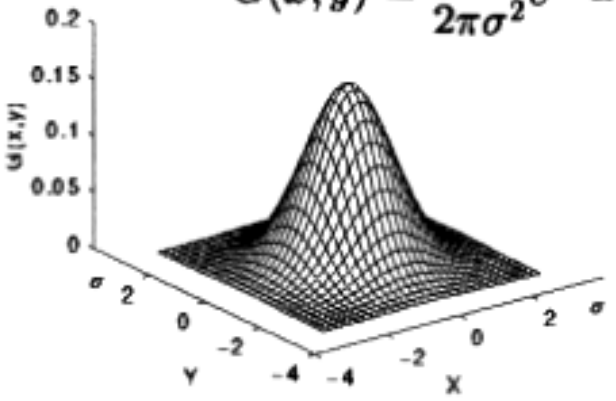
Detekce hran pomocí Laplaciánu

5.5 Kresba

Velmi zajímavé výsledky dává konvoluční maska diskrétní aproximace Laplacián Gaussiánu.

Gaussian (vyhlazovací filtr)

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



Gaussian je ve 2D izotropní a tak můžeme výpočet zjednodušit:

1D Gassian

$$\frac{1}{10.7}$$

1.3	3.2	3.8	3.2	1.3

aplikujeme ve směru x a y

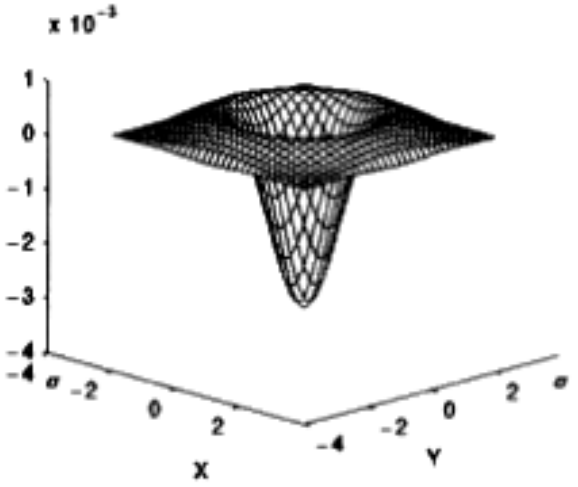
LaplaceGassian

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Diskrétní aproximace

$$\frac{1}{115}$$

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2



Př. Matice funkce LoG se $\sigma = 1.4$

$$h = \begin{pmatrix} 0 & 0 & 3 & 2 & 2 & 2 & 3 & 0 & 0 \\ 0 & 2 & 3 & 5 & 5 & 5 & 3 & 2 & 0 \\ 3 & 3 & 5 & 3 & 0 & 3 & 5 & 3 & 3 \\ 2 & 5 & 3 & -12 & -23 & -12 & 3 & 5 & 2 \\ 2 & 6 & 0 & -23 & -40 & -23 & 0 & 6 & 2 \\ 2 & 5 & 3 & -12 & -23 & -12 & 3 & 5 & 2 \\ 3 & 3 & 5 & 3 & 0 & 3 & 5 & 3 & 3 \\ 0 & 2 & 3 & 5 & 5 & 5 & 3 & 2 & 0 \\ 0 & 0 & 3 & 2 & 2 & 2 & 3 & 0 & 0 \end{pmatrix}$$

Diskrétní aproximace funkce LoG

6 Nestandardní transformace

Uvedený výčet představuje jen základní výtvarné efekty, které jsou převážně výpočetně nenáročné. Lze nalézt řadu dalších výpočetně náročných transformací. Pro příklad uveďme techniku LIC.

6.1 Lineární integrální konvoluce

LIC je technika určená pro vizualizaci vektorových polí. Nicméně se dá použít i při zpracování obrazu. Její podstatou je lokální „rozmazání“ textury podle směru vektorového pole. LIC vznikla z lineárních algoritmů, jejichž nevýhodou je, že nerespektují malá lokální zakřivení pole. Naproti tomu LIC aproximuje lokální chování pole pomocí čáry, která prochází středem počítaného pixelu a vine se podél „vlákna“ vektorového pole symetricky na obě strany. Metodu lze různě modifikovat pro dosažení dalších efektů.

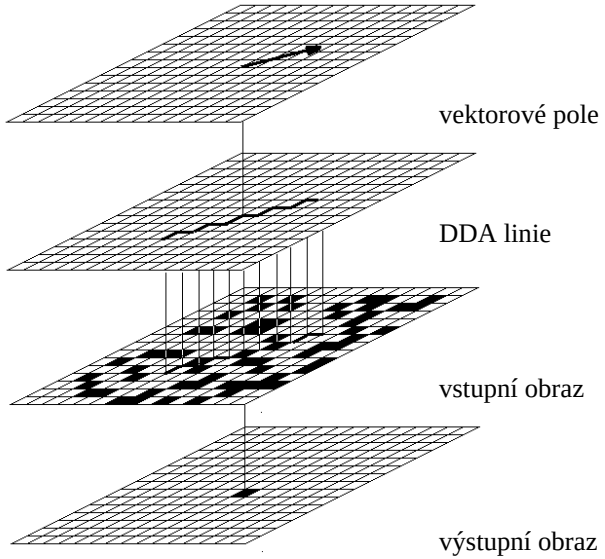
Obraz zpracovaný technikou LIC

LIC – Lineární integrální konvoluce

LIC je technika určená pro vizualizaci vektorových polí. Její podstatou je lokální „rozmazání“ textury ve směru vektorového pole.

LIC vznikla z lineárních algoritmů, jejichž nevýhodou je, že nerespektují malá lokální zakřivení pole.

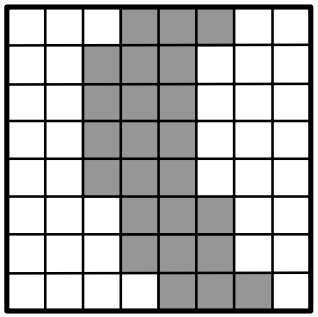
LIC aproximuje lokální chování pole podél myšlené čáry, která vychází z počítaného pixelu a vine se podle vektorového pole symetricky na obě strany.



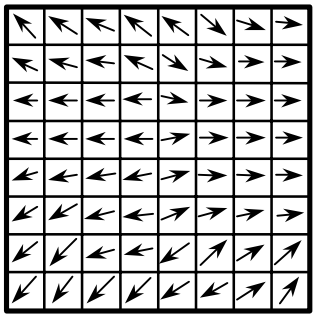
Postup: Pomocí konvolučních masek určíme v obrazu gradientní vektory. Operace se provádí na 256 úrovňové mapě (odstíny šedé).

Protože vektory představují gradienty, je třeba je otočit o 90 stupňů, aby směřovaly po „liniích“ obrázku.

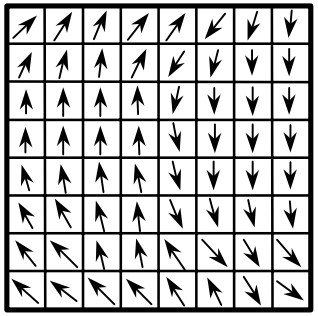
Originál



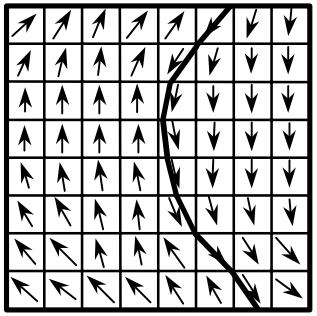
Gradient



Gradient otočený o 90°



Konečný stav



Vlastní LIC.

V každém bodě obrazu se počítá suma hodnot na pomyslné čáře délky *2L*

Proces lze zapsat rovnicí:

$$Output[x,y] = \frac{\sum_{i=0}^{l} Input(\lfloor P_i \rfloor) + \sum_{i=0}^{l'} Input(\lfloor P_i' \rfloor) h_i'}{\sum_{i=0}^{l} h_i + \sum_{i=0}^{l'} h_i'},$$

kde $Input(\lfloor P_i \rfloor)$ je pixel odpovídající $Input(\lfloor {}_x P_i \rfloor, \lfloor {}_y P_i \rfloor)$

Metoda je výpočetně náročná, proto se užívá zjednodušení:

První výpočetně náročná část je samotná detekce gradientů, proto se používá aproximace pomocí omezeného počtu směrů (například 8 –nebo 24).

Počet potřebných konvolučních masek je k počtu směrů poloviční.

Konvoluční masky:

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

(8 směrů)

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & -1 & 0 & 1 \\ -1 & -1 & -1 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 & -1 & 0 \\ -1 & -1 & -1 & 0 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ -1 & 0 & 0 & 0 & 1 \\ -1 & -1 & -1 & -1 & 0 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & -1 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & -1 & 0 & 1 \\ -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 0 & 1 & 1 & 1 \\ -1 & 0 & 1 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & -1 & 0 & 1 \\ -1 & -1 & -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 & 0 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -1 & -1 & 0 & 1 \\ -1 & -1 & -1 & 0 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & 0 & 1 & 1 & 1 \\ -1 & 0 & 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 & -1 & 0 \\ -1 & -1 & -1 & 0 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 & 0 \\ -1 & -1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 0 \\ -1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

(24 směrů)

Pokud je použit jednotkový integrační krok, jmenovatel zlomku udává počet „prošlých“ pixelů, což zjednodušuje výpočet jmenovatele zlomku. Nutnost počítat pohyb ve vektorovém poli v reálných číslech (změny souřadnic ve směru os jsou v rozsahu 0..1), přináší oproti celočíselné aritmetice zpomalení výpočtu.

LIC celočíselná

LIC v reálných číslech

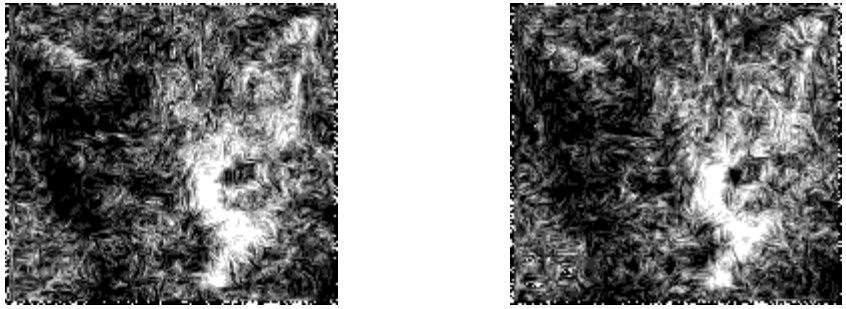
Pro urychlení je možné použít aproximaci, kdy se pohyb počítá tak, že se určí pixel, do nějž linie míří. Délka je pak daná vzdáleností tohoto pixelu od výchozího.

Poznámka: **Řada modifikací přináší další efekty.**

***Efekt hrubé zrno*: Před vlastním zpracováváním můžeme obraz zpracovat filtrem typu dolní propust.**

***Efekt směrové rozmazání*: Délku integračního kroku (L) je možné dynamicky měnit – například podle velikosti vektoru (pokud vektory představují rychlost, pak rozlišíme místa rychlá (větší L) a pomalá (menší L – méně „rozmazané“). Je možné řídit délku kroku intenzitou šedi (jasem)…**

***Efekt zašumění*: Použijeme vektorové pole vygenerované z obrázku a aplikujeme ho na bílý šum.Ve vektorovém poli se často vyskytují *singulární body*, které mohou ve výsledném obrázku působit rušivě. Toto je možné zaretušovat náhradou nulových vektorů náhodnými vektory.**



LIC v reálných číslech na bílém šumu (s a bez singularit)

***Efekt aplikace vektorového pole*: LIC lze využít pro zpracování obrazu i způsobem, pro nějž byla původně navržena – pro zobrazení vektorových polí. Vygenerujeme vektorové pole zajímavého tvaru (např. soustředné kružnice, čárové rastry atd.) a s tímto polem provedeme LIC nad obrázkem. Pokud však používáme zjednodušení na 24 nebo dokonce 8 směrů, vznikají problémy s generováním vektorového pole. Vektorové pole lze vytvořit i „ručně“ …**

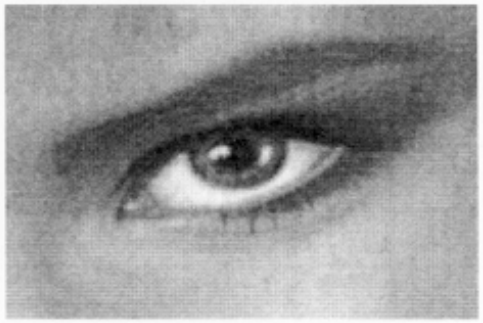
Aplikace metody LIC

Efekt „čtverečkového“ vektorového pole

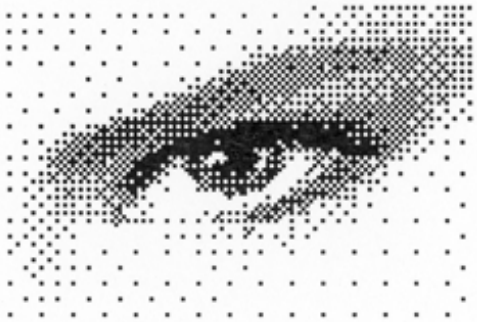
Vincent van Gogh:

(LIC neznal)

VZORKY ROZPTYLOVACÍCH ALGORITMŮ



originál



Bayer

Burkes

Floyd-Steinberg

Stucky

prahování (0,5)

Doplňme: I pouhé zvětšené vzorky rozptylovacích algoritmů (dithering) mohou býti výtvarně zajímavé.