

# Elektronická dokumentácia k diplomovej práci

## Algoritmy transformace 2D obrazu

Bc. PETRÁŠ Rastislav

# Obsah dokumentácie

- Úvod
- Teoretická časť
  - Reprezentácia obrazových dát
  - Geometrické transformácie obrazu
  - Transformácie farieb
- Praktická časť
  - Možnosti úpravy v jazyku C#
  - Algoritmy pre spracovanie farieb
  - Algoritmy pre grafické filtre

# Úvod

- V dnešnej modernej dobe sa stretávame so spracovaním obrazu na každom kroku. Či už v novinách, časopisoch, letákoch, reklamných tabulách, internetových stránkach. Často aj s upravenými fotkami, ktoré nezachytávajú skutočnú situáciu. Všetky tieto úpravy sú robené pomocou programov pre editáciu grafických dát.

# Úvod

- Medzi najbežnejšie používané úpravy medzi širokou verejnosťou patrí hlavne úprava farieb a vyhladzovanie. Transformácie obrazu majú však v počítačovej grafike oveľa väčší pojem napr. geometrické transformácie, transformácie farieb, spracovanie 2D a 3D obrazu. Všetky transformácie sú náročné na výpočtový výkon. To vedie vývojárov k optimalizovaniu softvéru a vyvíjaniu nového a rýchlejšieho hardvéru.

# Reprezentácia obrazových dát

## Vektorové dáta

- Obrazové dáta sú reprezentované vektormi. Dáta obsahujú informácie o objektoch zložených z kružníc, kriviek a jednoduchých telies, ktoré umožňujú ich geometrickú konštrukciu. Pokiaľ je takto uložená napríklad kružnica, súbor neobsahuje informácie o všetkých jednotlivých bodoch, na ktorej leží. Obsahuje informáciu, že sa jedná o kružnicu, potom informácie o jej strede, jedného bodu, ktorý na nej leží a posledný bod určuje rovinu jej konštrukcie. Ďalej sú to informácie o farbe objektu a hrúbke čiary. Vektorové dáta sa používajú hlavne pre technické výkresy v CAD systémoch, kde zmena mierky, alebo prípadná editácia objektov a tvarov nespôsobí skreslenie. Nie je však vhodná pre reprezentáciu náhodných obrazových dát ako sú fotografie.

# Reprezentácia obrazových dát

## Rastrové dáta

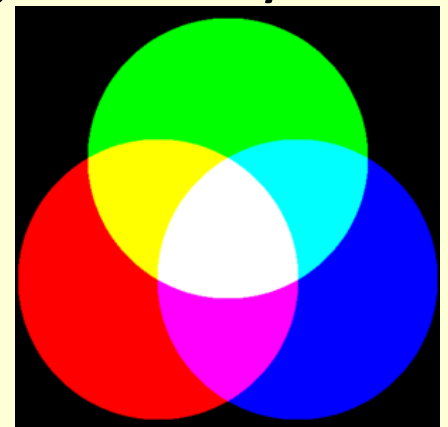
- Obraz je pri rastrovom formáte uložený v matici, kde každý bod matice reprezentuje jeden bod v obraze. Výhodou takéhoto formátu je jednoduchá úprava každého bodu zvlášť. Na tomto princípe pracuje väčšina zobrazovacích zariadení ako sú monitory, tlačiarne, televízory, atď.. Kvalita obrázku je daná predovšetkým počtom bodov a farebnej hĺbke.

# Farby

- Oko je vo svojej činnosti obmedzené nielen do veľkosti vnímaných objektov, ale aj do počtu farieb. Veľkosť bodu na obrazovke je závislá na veľkosti a rozlíšení monitora a pohybuje sa rádovo v desatinách milimetrov. Pokiaľ je vedľa seba umiestnených niekoľko takýchto bodov a každý má inú farbu, za normálnych podmienok oko nie je schopné body od seba odlíšiť. Zrková informácia je v mozgu integrovaná a my môžeme vnímať farbu, ktorá na monitore nie je zobrazená. Dôležité pri vnímaní farieb je spôsob, akým je farba vytvorená. Je všeobecne známe, že farby môžeme „miešať“. V farebných modeloch sú preto farby realizované miešaním základných farieb.
- Farebné modely sa delia na dve hlavné skupiny: aditívne a subtraktívne.

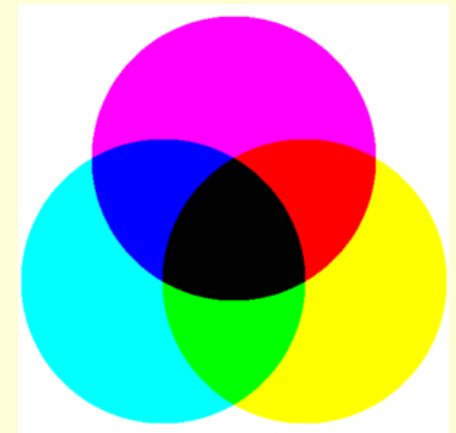
# Farby

- Aditívne miešanie farieb
  - Pri aditívnych systémov je podklad čierny a farby vznikajú pridávaním troch základných farieb a to červenej, zelenej a modrej. V prípade prekrytia všetkých základných farieb pri plnej intenzite vzniká farba biela. V prípade nulovej intenzity vzniká farba podkladu, čiže čierna.
  - Príkladom aditívneho miešania farieb sú televízory a monitory.



# Farby

- Subtraktívne miešanie farieb
  - Subtraktívne systémy majú farbu podkladu bielu a farby vznikajú odpočítavaním od bielej. Základné tri farby sú: azúrová, purpurová, žltá. Azúrová farba pohlcuje červenú zložku svetla, purpurová zelenú a žltá modrú. V prípade prekrytia všetkých základných farieb pri plnej intenzite vzniká farba čierna. V prípade nulovej intenzity vzniká farba podkladu, čiže biela.
  - Príkladom subtraktívneho miešania farieb sú tlačiarne.



# Farebné modely

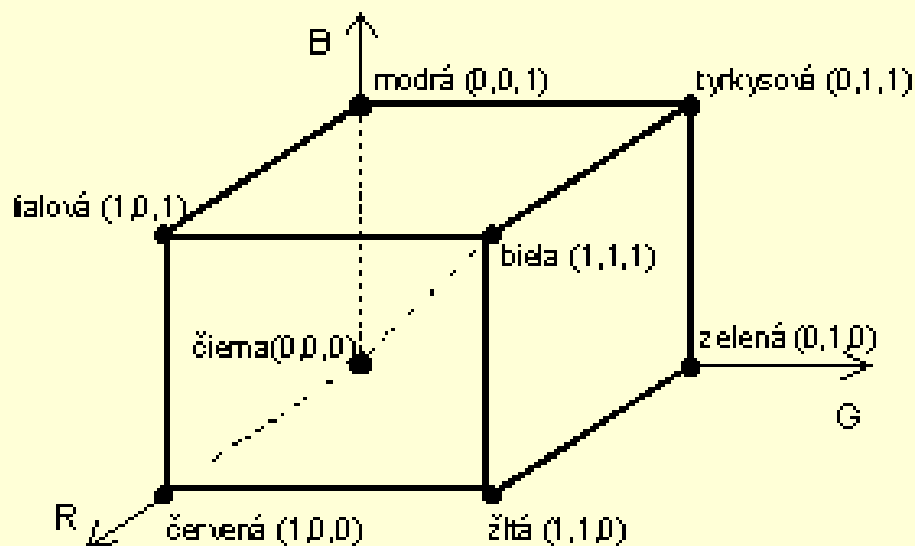
## RGB 1

- Ide o najpoužívanejší a najznámejší farebný model. Bol štandardizovaný v roku 1931. Farebný gamut je celý vyprodukovaný aditívnym spôsobom, a to miešaním troch základných farieb, červená (red), zelená (green) a modrá (blue).
- Medzi najbežnejšiu implementáciu modelu RGB patrí 24-bitová. Vtedy pripadá na každú farebnú zložku 8 bitov, čo dáva dohromady 256 úrovní každej zložky. Celkový možný počet farieb nám dáva  $2^{24}$  čo je približne 16,7 milióna. Platí, že čím väčšie číslo je, tým je farba svetlejšia. Celý model sa dá jednoducho zobrazíť ako jednotková kocka umiestnená v osách r,g,b. Počiatok súradnicového systému [0,0,0,] odpovedá čiernej farbe. Vrchol súradnicového systému v bode [1,1,1,] odpovedá farbe bielej. V protiľahlých rohoch ležia farby RGB.

# Farebné modely

## RGB 2

- V praxi sa môžeme stretnúť s modelom RGBA, kde  $A$  je informácia o priehľadnosti. Môže naberať hodnôt od 0 do 1, pričom 0,0 znamená že farebný bod je nepriehľadný a 1,0 že bod je úplne priehľadný.



# Farebné modely

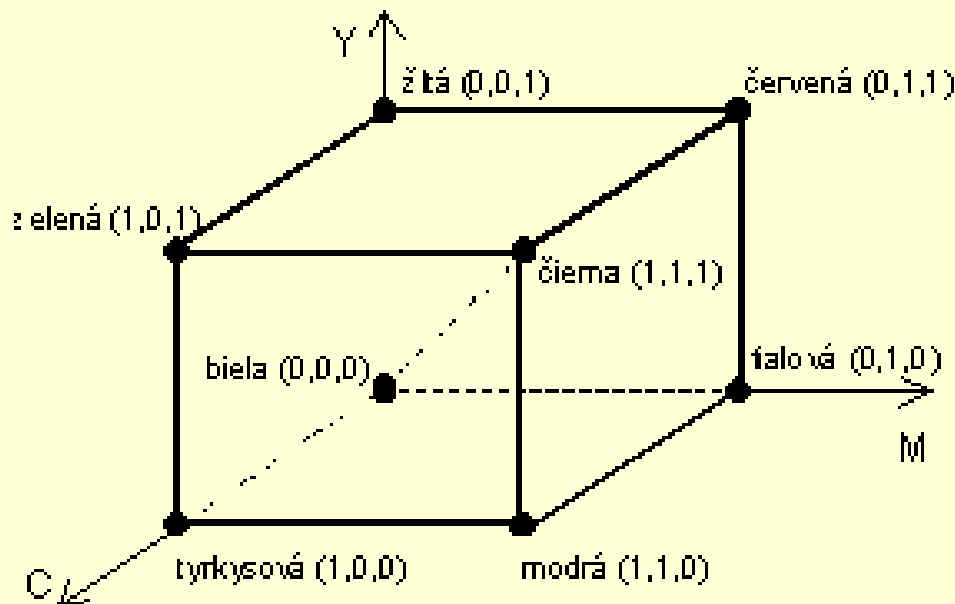
## CMY a CMYK 1

- Farebný model sa skladá z farieb Cian (modrozelená), Magenta (purpurová) a Yellow (žltá). Nakoľko z týchto farieb sa nedá namiešať dokonale čierna farba, bola zavedená ďalšia farebná zložka K (black). Farby v tomto modeli vznikajú subtraktívnym spôsobom.

# Farebné modely

## CMY a CMYK 2

- Najčastejšie využitie tohto modelu je v tlači. Zobrazenie v jednotkovej kocke je v počiatočných súradniciach  $[0,0,0,]$  farba biela a na vrchole  $[1,1,1,]$  farba čierna.



# Farebné modely

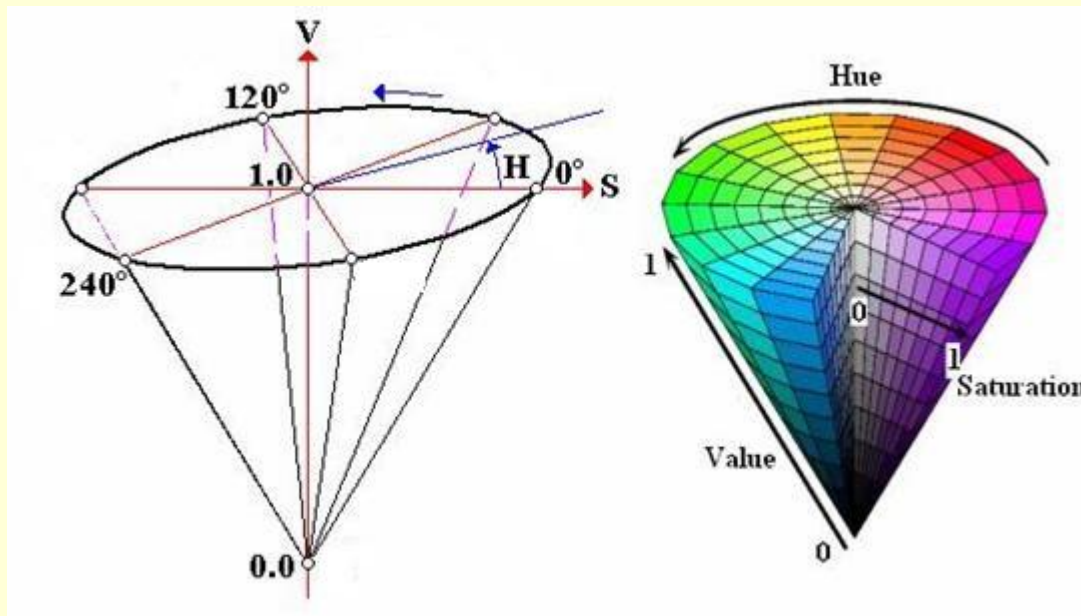
## HSV a HLS 1

- Patrí medzi systémy, kde farby nevznikajú miešaním. Skladá sa z farebného tónu – *Hue*, sýtosti – *Saturation* a jasovej hodnoty – *Value*. Farebný tón označuje prevládajúcu spektrálnu farbu, sýtosť určuje prímies iných farieb a jas je daný množstvom bieleho svetla. Používa sa napríklad v štandardnom farebnom editore Windows. Na zobrazenie priestoru sa používa šesťboký ihlan. Jeho vrchol leží v počiatku sústavy súradníc HSV. Hodnoty súradníc S a V sa menia od 0 do 1, hodnoty uhlu H nadobúdajú hodnôt z intervalu od  $0^\circ$  PO  $360^\circ$ . Vrchol ihlanu reprezentuje čiernu farbu. Ako rastie smerom k podstave, stred podstavy reprezentuje farbu bielu. Sýtosť odpovedá relatívnej vzdialenosti bodu od osy ihlanu. Dominantné farby ležia na plášti, čisté sú na obvode podstavy. Pri pohybe po obvode v rovnakej výške od základne sa postupne mení farebný tón, sýtosť a jas zostávajú bez zmeny.

# Farebné modely

## HSV a HLS 2

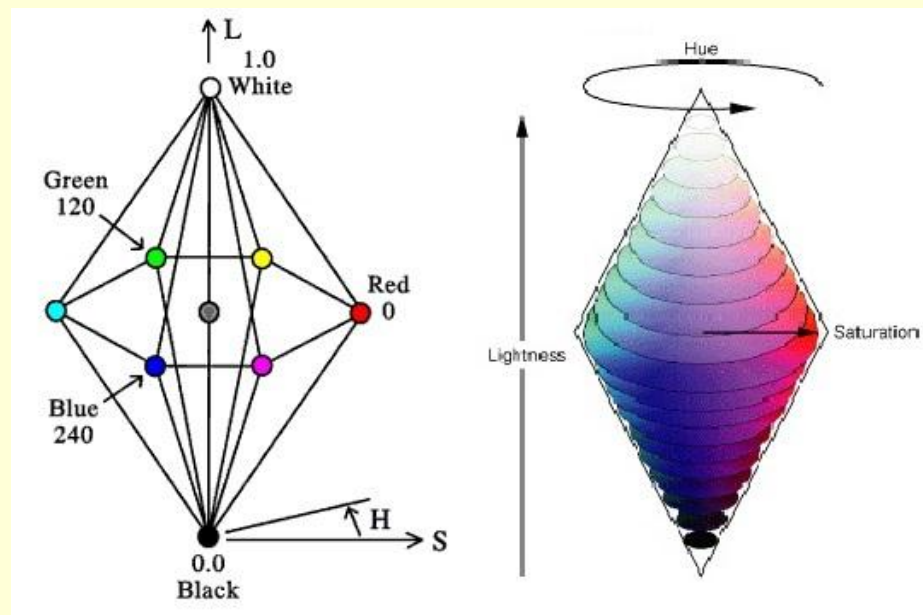
- Systém HSV má aj nedostatky, ktoré môžu sťažovať prácu s presným určením farby. Jedným z nich je ihlanovitý tvar, ktorý spôsobuje, že vo vodorovnom reze sa musí bod o konštantnej hodnote S pohybovať pri zmene H po dráhe v tvare šesťuholníka, čiže nie po kružnici. Medzi ďalšie nedostatky patrí nesymetria priestoru z hľadiska jas.



# Farebné modely

## HSV a HLS 3

- Všetky spomenuté nedostatky rieši farebný priestor HLS. Skladá sa z farebného tónu – Hue, svetlosti – Lightness a zo sýtosti – Saturation. Priestor HLS je obdobou priestoru HSV, kde bol ihlan nahradený dvojicou kužeľov.



# Formáty grafických súborov

- Rôzne formáty rastrových súborov sa vo všeobecnosti delia do dvoch kategórií a to, bez stratovej kompresie ako napríklad BMP, RAW, TIFF, TGA, PNG, MNG a so stratovou kompresiou ako JPEG, GIF.
- Výhody bezstratovej kompresie sú hlavne pri ďalších úpravách obrázku, kde nedochádza ku strate kvality. Ich nevýhodou je až niekoľkonásobná veľkosť oproti komprimovaným obrázkom.
- Výhoda stratovej kompresie je najmä v malej veľkosti súboru. Ich nevýhodou je nemožnosť obnovenia pôvodného obrázka. Využívajú nedokonalosť ľudského oka, takže bežný človek nerozozná rozdiely medzi stratovou a bezstratovou kompresiou.

# Formáty grafických súborov

## BMP

- Ide o grafický formát, ktorý je používaný Microsoftom. Je to bezstratový formát, ktorého najväčšou nevýhodou, prečo sa vo veľkej miere nepoužíva, je veľkosť. Maximálna farebná hĺbka je 24-bit. Existujú aj ďalšie modifikácie tohto formátu, ktoré pridávajú väčšiu farebnú hĺbku, alfa kanál, prípadne čiastočnú kompresiu. V praxi je používaný hlavne spoločnosťou Microsoft.

# Formáty grafických súborov

## TIFF

- Bol definovaný firmou Aldus v roku 1986 ako štandardná metóda pre ukladanie čiernobielych predlôh vytvorenými skenermi. V súčasnosti je TIFF najpoužívanejším štandardným bezstratovým formátom súborov vo väčšine kresliacich programov. Svoje využitie našiel hlavne v skenovaní. Jeho schopnosť rozšírenia a podpora veľkého množstva dátových kompresných schém dovoľuje vývojárom prispôbiť si TIFF formát svojim potrebám. Využíva sa hlavne u obrázkov, ktoré sa majú s istotou ďalej upravovať. Ide taktiež o výhradný formát pre Microsoft Windows GUI. Jeho schopnosť rozšírenia, kedy je schopný ukladať viacnásobné bitmapové predlohy, umožňuje splniť aj tie najnáročnejšie požiadavky pre ukladanie dát. Podporuje farebnú hĺbku až 24 bitov a veľa druhov kompresíí. Univerzálne vlastnosti mu umožňujú jeho použitie v akomkoľvek operačnom prostredí. Najväčšie nevýhody formátu TIFF sú sťažnosti na jeho údajnú neprenositelnosť medzi jednotlivými aplikáciami. Tieto problémy sa väčšinou zvädzajú hlavne na vlastné súbory TIFF. Medzi ďalšie nevýhody je ich veľkosť, čo je ale daňou za kvalitu.

# Formáty grafických súborov

## JPEG 1

- Je v súčasnosti najpoužívanejší a najznámejší formát obrázkových súborov. Jeho najväčšie rozšírenie spôsobilo rozširovanie internetu. Ide o stratový formát. Jeho hlavnou výhodou je malá veľkosť. Skupina JPEG vznikla v roku 1986. Princíp kompresného algoritmu je v použití DCT (diskrétna kosínusová transformácia) a kvantizifikácií získaných koeficientov. Celá kompresia a strata údajov je vytvorená tak, aby sa strácali len informácie, ktoré ľudské oko nevie dobre rozlíšiť. To je totiž oveľa citlivejšie na zmenu intenzity farby ako na malú zmenu samotnej farby. Napríklad pri kompresnom pomere 15:1 až 25:1 nedôjde k výraznej strate kvality fotografie.

# Formáty grafických súborov

## JPEG 2

- V oblasti kompresie obrazu sa snažia do popredia dostať algoritmy založené na vlnkovej transformácii (Wavelet Transformation). DWT (Diskrétna vlnková transformácia) dokáže reprezentovať obraz efektívnejšie ako napríklad DCT. DWT rozkladá obraz do tzv. bázových funkcií, ktoré sú obecné výhodnejšie pre reprezentáciu digitálnych signálov než sínusové alebo kosínusové funkcie. Na tomto základe bol vytvorený štandard JPEG2000 . Diskrétnu kosínusovú transformáciu nahradila DWT, ktorá umožňuje dosiahnuť vyššieho stupňa kompresie.

# Formáty grafických súborov

## RAW

- Formát RAW sa používa pri profesionálnej fotografii. Ide o dáta, ktoré sú zobrazené priamo zo snímača zariadenia. Ich hlavnou výhodou je oveľa širšia možnosť úprav, nedochádza ku stratám – všetky algoritmy na spracovanie a prevod obrazu sú ponechané až na úpravu v počítači, dáta sú ukladané s takou farebnou hĺbkou, s akou sú zo snímané. Oproti formátu TIFF nezaberajú toľko miesta. Nevýhoda je, že každý výrobca používa iný čip, a preto nie každý program je schopný načítať akýkoľvek súbor. Preto je potrebné v niektorých prípadoch použiť softvér originál od výrobcu.

# Formáty grafických súborov

## GIF

- Je výtvorom CompuServe Inc. a slúži k ukladaniu viacerých bitmapových obrázkov v jednom súbore pre výmenu medzi rôznymi platformami a systémami. Pokiaľ to berieme podľa počtu súborov za dobu jeho existencie, ide pravdepodobne o najrozšírenejší formát pre ukladanie multibitovej grafiky a obrazových dát.
- Prevažná väčšina súborov GIF obsahuje 16 alebo 256 farebné obrázky takmer vo fotografickej kvalite. Často sa však do neho ukladajú obrázky v škále šedej.
- Najväčšou nevýhodou formátu GIF je, že využíva komprimačný algoritmus LZW, ktorý nie je zadarmo. Medzi ďalšie nevýhody patrí, že podporuje maximálne 8bit farby.

# Formáty grafických súborov

## PNG

- Vznikol ako alternatívna náhrada formátu GIF, ktorý je patentovo chránený. Vznikol pod záštitou konzorcia W3C a nie je patentovo chránený. Bol navrhnutý s cieľom vytvoriť jednoduchý formát, ktorý sa dá ľahko implementovať, ktorý by bol plne prenositeľný a ktorý dosahuje alebo prekračuje všetky schopnosti formátu GIF. Používa sa hlavne pre jeho bezstratovú kompresiu a zobrazovanie obrázkov na internete. Používa kompresný algoritmus deflate/inflate s posuvným oknom o veľkosti 32 768 bytov. Kompresia Deflate je 100% bezstratová, ktorej autorom je Phil Katz a používa sa aj v utilite pkzip pre archiváciu súborov hlavne pre jej rýchlosť, dobrú zdokumentovateľnosť, je voľne dostupná a podporovaná na veľkom počte operačných platforiem. Jeho hlavné výhody oproti GIF sú možnosť ukladať obrázky v 48 bitovej farebnej hĺbke, alfa kanál pre transparentnú masku, detekcia poškodenia súboru a obrazová gama informácia podporuje automatické jasovo/kontrastové prispôbenie.

# Geometrické transformácie obrazu

- Geometrické transformácie sa používajú k úprave alebo korekcii obrázkov. Patrí sem posunutie, otočenie, zmena veľkosti, skosenie a v špecifických prípadoch aj zložitejšia metóda nazývaná “warping”. Základné transformácie sú v podstate jednoduché matematické operácie. Avšak pokiaľ potrebujeme napríklad otočiť obrázok s rozlíšením 100x100 okolo iného bodu ako je počiatok súradníc, musíme najskôr posunúť bod okolo ktorého sa bude obrázok otáčať (počítať sto tisíc operácií), otočiť obrázok (počítať sto tisíc operácií) a posunúť bod naspäť (počítať sto tisíc operácií). Keby sme to mali všetko robiť postupne trvalo by to veľmi dlho. Preto sa transformácie skladajú. To prebieha v homogénnych súradniciach, pomocou matic. V našom prípade budeme preto celú operáciu robiť len raz, čiže spravíme sto tisíc operácií.

# Geometrické transformácie obrazu

## Posunutie

- Posunutie znamená premiestnenie objektu z pôvodnej pozície do novej.
- Algebraický zápis pre posunutie 2D obrazu je

$$\begin{aligned}x' &= x + t_x \\ y' &= y + t_y\end{aligned}$$

- S použitím matíc

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

# Geometrické transformácie obrazu

## Otočenie

- Otočenie je transformácia bodu okolo pevného bodu po kruhovej dráhe. Určuje ju uhol otočenia a stred otočenia.
- Algebraický zápis pre otočenie 2D obrazu je

$$x' = x \cos(\beta) - y \sin(\beta)$$

$$y' = x \sin(\beta) + y \cos(\beta)$$

- S použitím matíc

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\beta) & -\sin(\beta) & 0 \\ \sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

# Geometrické transformácie obrazu

## Zmena mierky

- Pri zmene mierky dochádza k zmene veľkosti objektu a jeho polohy v smere súradnicových osí. Ak je koeficient pomeru novej dĺžky ku starej väčší ako 1, objekt sa predĺži, ak je koeficient menší ako 1, tak sa výsledný objekt skrúti.
- Algebraický zápis pre zmenu mierky 2D obrazu je

$$\begin{aligned}x' &= s_x x \\ y' &= s_y y\end{aligned}$$

- S použitím matíc

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

# Geometrické transformácie obrazu

## Skosenie

- Pri použití skosenia dochádza k zmene jednej súradnice, pričom druhá ostáva zachovaná.
- Algebraický zápis pre skosenie 2D obrazu pre os x je

$$\begin{aligned}x' &= x - y \tan(\beta) \\ y' &= y\end{aligned}$$

- S použitím matíc

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \tan(\beta) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

# Geometrické transformácie obrazu

## Odraz 1

- Zrkadlový odraz, inak nazývaný súmernosť, je v podstate zmenou mierky a to tak, že os podľa ktorej chceme odraz previesť, prehodíme znamienko súradníc.
- Algebraický zápis pre odraz 2D obrazu podľa osi x je

$$\begin{aligned}x' &= -x \\ y' &= y\end{aligned}$$

- S použitím matíc

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

# Geometrické transformácie obrazu

## Odraz 2

- Algebraický zápis pre odraz 2D obrazu podľa osi  $y$  je

$$\begin{aligned}x' &= x \\y' &= -y\end{aligned}$$

- S použitím matic

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

# Geometrické transformácie obrazu

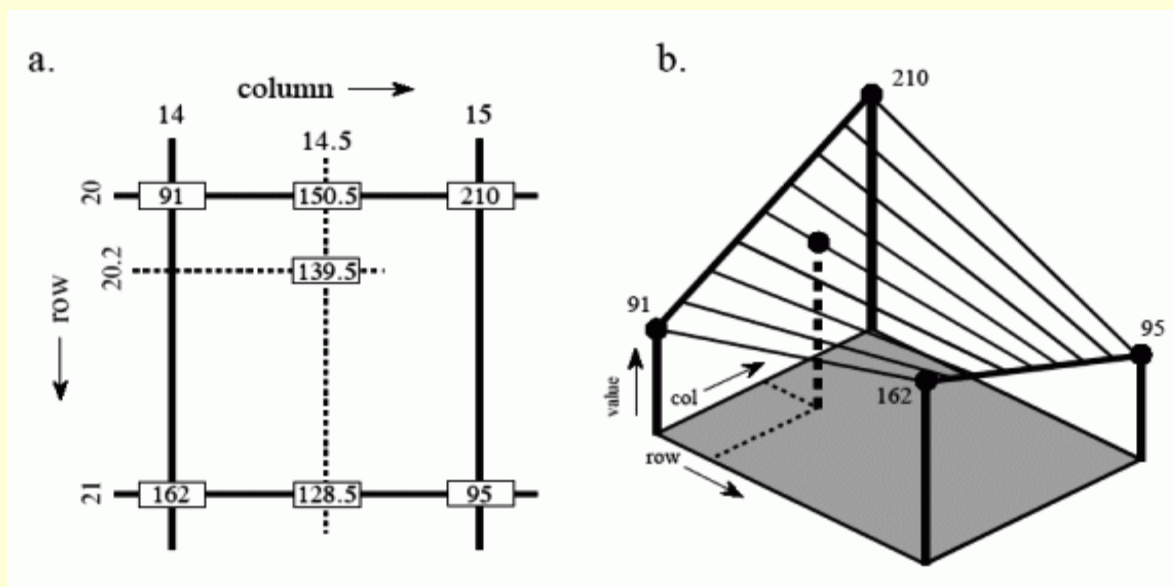
## Warping 1

- Warping patrí medzi zložitejšie deformácie obrazu. Ide o takzvané pokrútenie alebo zvlnenie objektu, kde máme zadanú množinu referenčných bodov medzi pôvodným a výstupným obrazom. Algoritmy warpingu môžeme rozdeliť do dvoch kategórií. Jednouúčelové algoritmy, ktoré sa zadávajú jednoducho niekoľkými parametrami a majú jediný výstup. Druhá skupina algoritmov zahrňuje obecné metódy, ktoré umožňujú prakticky ľubovoľnú transformáciu.

# Geometrické transformácie obrazu

## Warping 2

- Prvá z nich je založená na položení virtuálnej siete na obraz. Tá svojou zmenou určuje warping obrazu. Sieťový warping je vhodný pre globálne zmeny v obraze a nie je s ním jednoduché robiť malé lokálne operácie. Druhá metóda je založená na vkladanie takzvaných magnetov do obrazu. Zmena ich polohy určuje transformáciu. Sú vhodné pre malé, lokálne transformácie v obraze.



# Transformácie farieb obrazu

- Transformácie farieb majú kľúčovú úlohu pri tvorbe tieňovaných vizualizácií priestorových dát a to tak, že určujú akými spôsobmi je pôvodná farba modifikovaná. Zlepšujú sa tak vizuálne vlastnosti upravovaného obrázku, ktorý je vhodný pre ďalšie použitie, alebo tlač. Pri transformácií by mal ostať zachovaný odtieň pôvodných farieb. To znamená, že meniť by sa mala len ich intenzita.

# Transformácie farieb obrazu

## Odtieň šedej

- Najjednoduchším spôsobom redukcie farieb obrazu je prepočet na odtiene šedej. Farebné odtiene môžu byť na odtiene šedej prevedené veľmi jednoducho podľa vzorca :

$$I = 0,266 * R + 0,587 * G + 0,114 * B$$

- $I$  je výsledná intenzita (úroveň šedej) a  $R, G, B$  sú základné farebné zložky pôvodnej farby. Prevod obrazu na úrovne šedej spočíva v postupnom prepočítaní všetkých bodov obrazu podľa vyššie uvedeného vzorca.

# Transformácie farieb obrazu

## Negatív 1

- Jedná sa o najjednoduchšiu transformáciu obrazu. Mení sa intenzita jasu každého bodu v obraze podľa vzorca:

$$g(x, y) = 1 - f(x, y)$$

- pre obraz v stupňoch šedi a

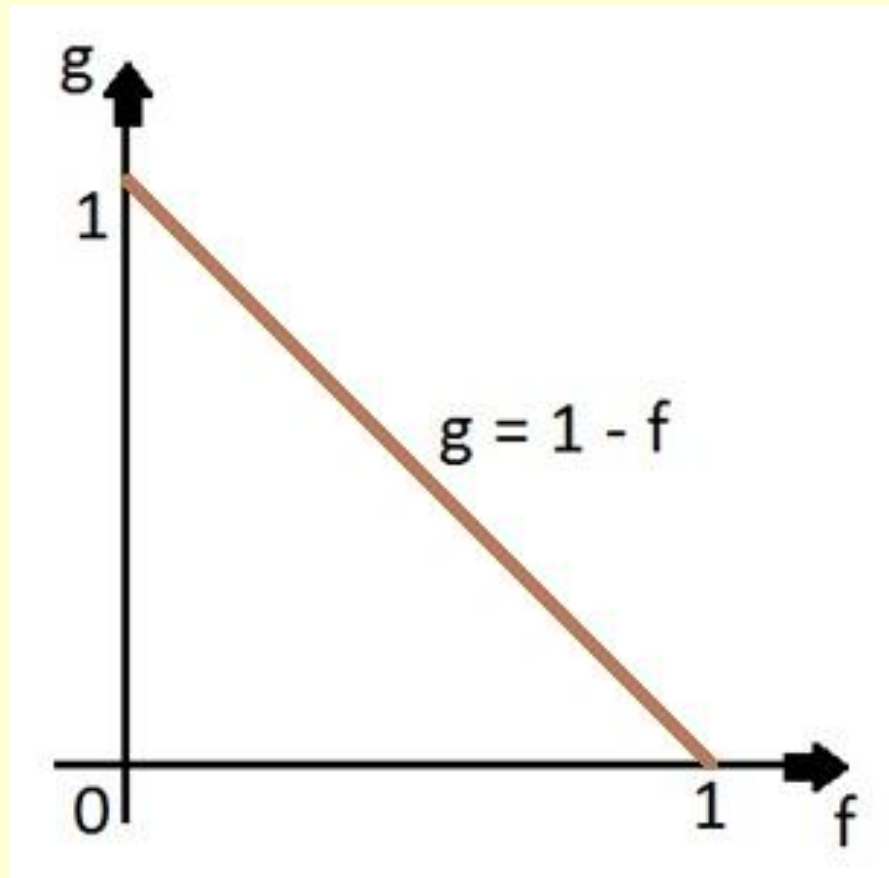
$$g(x, y) = L - 1 - f(x, y)$$

- pre farebný obraz, kde L je počet jasových úrovní. Napríklad u 8 bitového farebného modelu je L=256.

# Transformácie farieb obrazu

## Negatív 2

- Transformačná krivka hodnôt jasu pre negatív:



# Transformácie farieb obrazu

## Gama 1

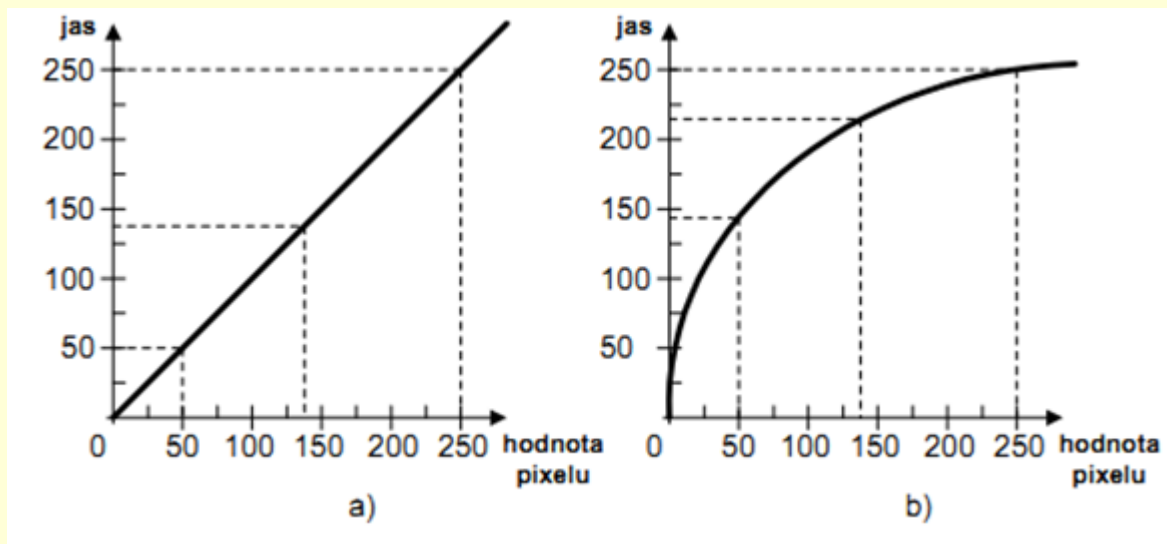
- Korekcia gamy je najčastejšie používanou nelineárnou funkciou. V súčasnosti už takmer nepoužívané CRT obrazovky fungujú na princípe toku elektrónov, ktoré vybudia fosfor, ktorý začne žiariť určitou farbou a vysvieti na obrazovke bod. Závislosť medzi jasom (intenzitou) a hodnotou pixelu nie je lineárna. To je spôsobené nelineárnym chovaním fosforu pri vybudení elektrónovým lúčom.
- Gama faktor: odchýlka hodnoty pixelu od reálneho jas pixelu
- Gama korekcia: slúži na opravu (odstránenie) nelinearit zobrazovacieho systému, čiže úprava jas na obrazovke. Využíva sa tiež aj na opravu zlej expozície, kde niektoré miesta sú príliš tmavé alebo naopak príliš svetlé.

# Transformácie farieb obrazu

## Gama 2

- Korekcia gamy sa odstraňuje pomocou vzorca:

$$i_n = i^{\frac{1}{\gamma}}$$



Závislosť hodnoty bodu a jeho jasu:

- Ideálna
- Reálna

# Transformácie farieb obrazu

## Gama 3

- Hodnota  $\gamma$  závisí na type obrazovky. Väčšina televízorov v súčasnej dobe má túto konštantu už predom nastavenú, a nemusíme sa teda o nič starať. Kvalitnejšie monitory majú zabudované čidlo, vďaka ktorému sa vedia sami kalibrovať, nakoľko hodnota konštanty závisí aj na teplote. Hodnota  $\gamma$  sa vo väčšine prípadov pohybuje v rozmedzí 1,8 až 2,5  $\pm$  = 0,3.

# Transformácie farieb obrazu

## Jas

- Patrí medzi najjednoduchší spôsob zlepšenia vzhľadu zle exponovaného obrázku. Vo väčšine prípadov stačí práve len úprava jasů. Zmeny sa docielia transformáciou pôvodných jasových úrovní na iné. Používajú sa štandardné aritmetické operácie.
- Základný vzorec pre výpočet je
$$g(x, y) = T[f(x, y)],$$
- kde  $g(x, y)$  je výstupný obraz,  $T$  je transformácia a  $f(x, y)$  je vstupný obraz. Pri zložitejších transformáciách sa výsledný obraz počíta aj z okolia vstupného obrazu ako napríklad ostrenie obrazu pomocou nelineárnej metódy alebo pri vyhladzovaní.

# Transformácie farieb obrazu

## Kontrast 1

- Používa sa v prípade, pokiaľ je rozdiel medzi rôznymi úrovňami jasu v obraze príliš malý.
- Obrázok je v tomto prípade nekontrastný a pre nás dôležité dáta sa nachádzajú v príliš úzkom pásme jasových úrovní. Predpokladajme, že minimálna hodnota jasu vyskytujúca sa v obraze je  $f_1$  a maximálna hodnota je  $f_2$ . Hodnoty z ich intervalu je nutné previesť na celý zobraziteľný rozsah hodnôt. Transformáciu realizujeme tak, že od danej hodnoty jasu  $f$  odpočítame minimálnu hodnotu nachádzajúcu sa v obraze a vynásobíme koeficientom rozťahnutia  $1/(f_2 - f_1)$ . Výsledný vzorec je:

$$g = \frac{f - f_1}{f_2 - f_1}$$

# Transformácie farieb obrazu

## Kontrast 2

- Alternatívne, ale menej často používaný je nelineárny vzťah:

$$g = \frac{1}{1 + \left(\frac{m}{f + \varepsilon}\right)^E}$$

- kde  $f$  sú vstupné hodnoty, hodnota  $m$  určuje stred rozťahnutia, hodnota  $E$  určuje strmosť krivky a konštanta  $\varepsilon$  zabraňuje deleniu nulou.

# Transformácie farieb obrazu

## Interpolácia 1

- Medzi časté požiadavky pri úprave obrázkov je zmena rozlíšení. Obrázok môžeme buď zmenšiť, alebo zväčšiť. Pre každý bod nového obrázku je nutné vypočítať jeho polohu a hodnotu podľa transformačnej funkcie. Existuje viacero metód pre výpočet. Najjednoduchšou je interpolácia najbližším susedom. Hodnota bodu je daná hodnotou bodu v pôvodnom obraze, ktorý je k nemu najbližšie. Táto metóda nie je vždy uspokojivá a spôsobuje stratu detailov. Najčastejšie používanou metódou je bilineárna interpolácia.

# Transformácie farieb obrazu

## Interpolácia 2

- Pri bilineárnej interpolácii je výsledná hodnota jasú daná hodnotami jasú a polohy štyroch susedných bodov:

$$f(i, j), f(i + 1, j), f(i, j + 1), f(i + 1, j + 1),$$

- kde  $(i, j)$  sú súradnice bodu a  $f$  má význam hodnoty jasú. Táto metóda je výpočtovo jednoduchá a vo väčšine prípadov postačuje.
- V prípade potreby na oveľa väčšiu kvalitu obrazu, je možné použiť zložitejšiu bikubickú interpoláciu. V tomto prípade sa pre výpočet hodnoty výsledného bodu používa viacej susedných bodov.

# Transformácie farieb obrazu

## Histogram 1

- Histogram patrí medzi najefektívnejšie nástroje kontroly kvality a úpravu snímky. Zachytáva rozdelenie stupňov svetlosti obrazových dát v grafickej forme. Matematicky sa dá histogram vyjadriť ako vektor absolútnej početnosti hodnôt zastúpených v obraze. Stĺpce, ktoré sa nachádzajú nad každou hodnotou, hovoria formou diagramu o tom, ako často sa určitá intenzita pixelu nachádza v obrázku. Keď sa pruhy dotýkajú, histogram naberá tvar pohoria. Čím viacej je niektorý odtieň v obrázku zastúpený, tým vyšší je jeho pruh. Pokiaľ niektorý odtieň chýba, je toto miesto prázdne.

# Transformácie farieb obrazu

## Histogram 2

- Pokiaľ je pôvodný obrázok zložený z jasových zložiek, je histogram jednorozmerný vektor a reprezentuje ho jeden histogram. Pri farebných obrázkoch odpovedá každej farbe vlastný histogram. Tie sa môžu navzájom výrazne líšiť.

# Transformácie farieb obrazu

## Histogram 3

- Na obrázku môžeme vidieť názornú ukážku rozloženia histogramu, kde v ľavej časti sa nachádzajú tmavé miesta a v pravej časti svetlé miesta.



# Transformácie farieb obrazu

## Histogram 4

- Histogram je neoceniteľným pomocníkom pri analyzovaní fotografií. Môžeme z neho okamžite zistiť, či je obrázok správne exponovaný, informácie o kontraste, o dynamickom rozsahu obrázku a mnoho ďalšieho. Nie je to však vždy ľahké. Pomocou histogramu dokážeme taktiež určiť, či bol obrázok niekedy upravovaný a či bol na uloženie zvolený vhodný formát. Ak použijeme na uloženie obrázku kompresný formát, zmení to aj výsledný histogram.

# Transformácie farieb obrazu

## Histogram 5

- Základne charakteristiky histogramu
- Pokiaľ je graf šedej stupnice väčšinou na ľavej strane, obrázok je príliš tmavý
- Pokiaľ je graf šedej stupnice väčšinou na pravej strane, obrázok je príliš svetlý
- Pokiaľ graf šedej stupnice nie je dostatočne rozložený, je pravdepodobne potrebné zvýšiť kontrast
- Pokiaľ je graf v úzkej oblasti, fotografia pravdepodobne nemá dostatok detailov
- Pokiaľ je graf rovnomerné rozložený, fotografia má pravdepodobne vyrovnanú kompozíciu a dostatok detailov

# Transformácie farieb obrazu

## Vyrovnanie histogramu 1

- Pokiaľ máme nevhodne exponovaný obraz, nerovnomerne rozložené zložky histogramu, snažíme sa ho upraviť tak, aby bolo toto rozloženie rovnomerné (úrovne jasú by sa vyskytovali pravidelne). Metóda, ktorou môžeme tento výsledok docieľiť, sa nazýva metóda vyrovnania histogramu (ekvalizácia histogramu).

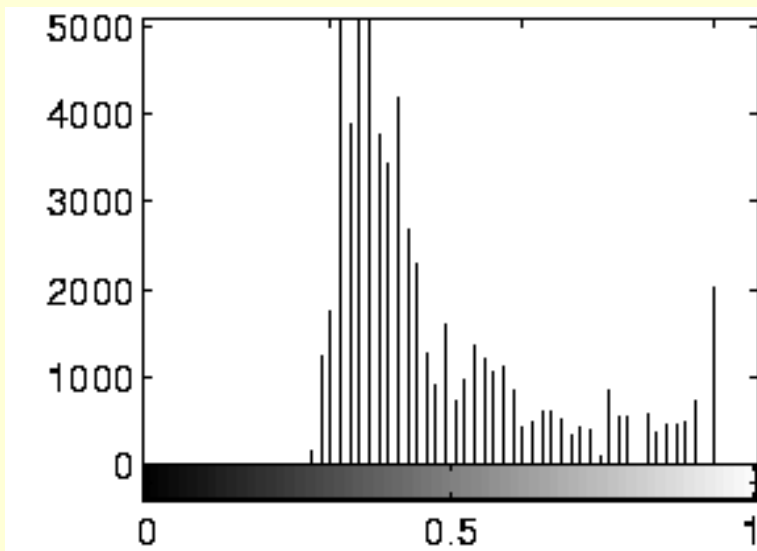
# Transformácie farieb obrazu

## Vyrovnanie histogramu 2

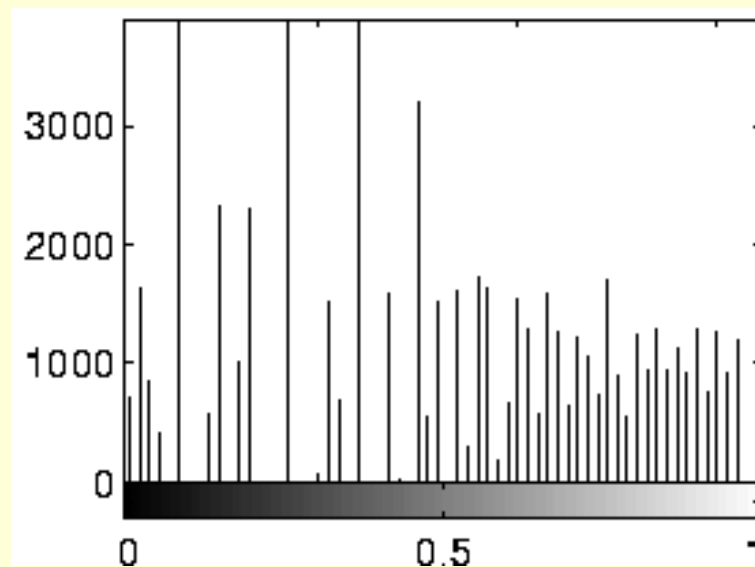
- Tmavé aj svetlé odtiene majú veľmi nízky kontrast. Avšak bežným zvyšovaním kontrastu ho paradoxne ešte viacej znížime. Obraz ako celok má totiž kontrast veľmi vysoký a jeho zvyšovaním posúvame tmavé tóny do ešte tmavších a svetlé do ešte svetlejších. Nepomôže ani zmena jasú, lebo pri znižovaní orezávame tmavé farby a pri zvyšovaní orezávame svetlé farby. Jediná možnosť ako vyriešiť tento problém je už v spomínanej metóde vyrovnaní histogramu, čo zabezpečí rovnomerne zastúpenie tmavých, stredných a svetlých farebných odtieňov. Týmto dosiahneme kontrastnejší obraz a zvýraznenie detailov.

# Transformácie farieb obrazu

## Vyrovnanie histogramu 3



Príklad na nevyrovnaný histogram



Príklad na vyrovnaný histogram

$$g(k) = \frac{q_k - q_0}{x * y} * \sum_{j=k_i}^{i=k} L(i)$$

$L(i)$  predstavuje histogram početnosti jasov  $i$  vo vstupnom obrázku (počet bodov s rovnakou intenzitou),  $x * y$  je celkový počet bodov v obrázku,  $\langle q_0, q_k \rangle$  interval výstupných jasov, a  $k_i$  je najnižšia intenzita vstupného obrázku.

# Filtrácia obrazu

- Pod pojmom filtrácia obrazu sa rozumie práca s digitálnym obrazom, pri ktorej dochádza k zvýrazneniu alebo potlačeniu určitých informácií. Vo filtrácií sa rozlišuje celá rada metód pre úpravu. Pri úprave obrazu sa využívajú dva hlavné spôsoby a to priestorové a frekvenčné. Priestorové metódy pracujú priamo s obrazom  $f(x,y)$ , alebo jeho časťou, ktorú transformujú na výstupný obraz  $g(x,y)$  podľa vzorca:

$$g(x, y) = T[f(x, y)].$$

- Pri frekvenčných metódach sa využíva konvolúcia obrazu.

# Filtrácia obrazu

## Odstraňovanie šumu

- Patrí medzi často používanú a užitočnú funkciu pri úpravách obrazu. Šum je definovaný ako nová informácia, ktorá bola k pôvodnej informácii pridaná snímacím zariadením alebo pri transfere dát. Vo väčšine prípadov sa prejavuje v obraze ako zrnenie. Šum rozdeľujeme do dvoch kategórií a to podľa spôsobu, ako bol pridaný do pôvodného obrazu a to na aditívny a multiplikatívny. V praxi existuje viac druhov šumu a vždy je potrebné zvážiť, ktorá metóda sa použije na jeho odstránenie, nakoľko výsledky môžu byť veľmi rozdielne. Hlavným problémom je identifikovať, či ide o šum alebo nie. Každý filter nejakým spôsobom vyhodnotí na základe okolia a veľkosti zmeny hodnoty bodu, či ide o šum. Filtre pracujú buď na princípe konvolúcie alebo lokálnej štatistiky okolia.

# Filtrácia obrazu

## Odstraňovanie šumu

### Lineárne metódy - Spriemerovanie

- Filtrácia spriemerovaním je jednoduchým príkladom vyhadzovania obrazu pomocou konvolúcie, kde sa na obraz použije konvolučná matica. Ide o najjednoduchšiu metódu vyhladzovania šumu v obraze, pri ktorej sa spoliehame na značnú redundanciu dát v obraze. Vďaka tomu môžeme predpokladať, že susedné body majú rovnaký, prípadne podobný jas. Každému bodu obrazu sa jeho jas nahradí aritmetickým priemerom jasom susedných bodov. To spôsobí, že ostré hrany alebo šum sa v obraze rozmazú, čo je najväčším nedostatkom tejto metódy. Opakovaným použitím filtru dochádza k čoraz väčšiemu rozmazaniu, až nakoniec výsledný obraz dosiahne jednu farbu, ktorá je priemerom všetkých hodnôt. Vo väčšine prípadov sa za susedov považujú body zo štvorcového nepárneho okolia bodu (3x3, 5x5, 7x7,.... atď.).

$$g(x, y) = \frac{1}{M} \sum_{(i,j) \in \Omega} f(i, j)$$

- Konvolučná maska 3x3 vyzerá nasledovne

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

# Filtrácia obrazu

## Odstraňovanie šumu

Lineárne metódy – Gaussové vyhladzovanie <sup>1</sup>

- Ide o podobný princíp ako vyhladzovanie pomocou spriemerovania. Rozdiel je v konvolučnej matici. Tá ma nastavené koeficienty tak, že tie ktoré sú bližšie k strede majú vyššiu váhu, aby odpovedali gaussovej krivke.
- Príklad matice 3x3

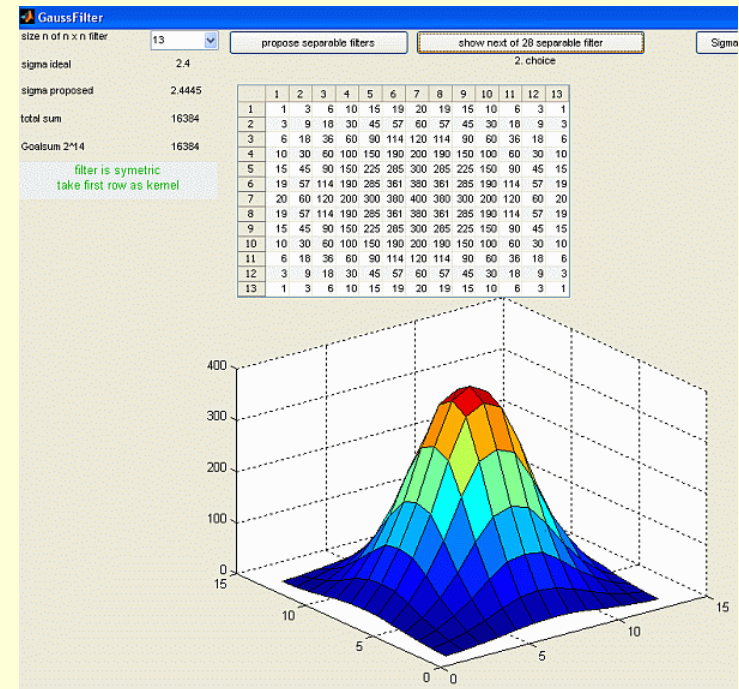
$$h = \frac{1}{10} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

# Filtrácia obrazu

## Odstraňovanie šumu

Lineárne metódy – Gaussové vyhladzovanie 2

- Na obrázku vidíme tvar hustoty pravdepodobnosti  $h(x,y)$  pre dvojrozmerný náhodný vektor s Gaussovým rozdelením.



Príklad na gaussovo rozdelenie

# Filtrácia obrazu

## Odstraňovanie šumu

Lineárne metódy – Gaussové vyhladzovanie 3

- Dvojrozmerné Gaussovo rozdelenie je dané vzťahom:

$$h_i = \frac{\exp \frac{-(G_j - m)^2}{2\sigma^2}}{\sigma\sqrt{2\pi}} \quad -\infty < G_j < \infty$$

- Funkcia hustoty  $G_j$  je definovaná pre všetky body v rovine od  $-\infty$  do  $+\infty$ . Od parametru  $\sigma$  závisí veľkosť masky. Od určitej dostatočne malej hodnoty nie sú ostatné hodnoty považované za významné.

# Filtrácia obrazu

## Odstraňovanie šumu

### Nelineárne metódy

- Nelineárne metódy odstraňovania šumu pracujú na princípe lokálnej štatistiky v okolí bodu, kde sa snažia nájsť tú časť, do ktorej reprezentovaný bod patrí. Tú potom dosadia na miesto upravovaného bodu. Ich výhoda oproti lineárnym filtrom je, že výsledný obraz nie je tak rozmazaný.

# Filtrácia obrazu

## Odstraňovanie šumu

Nelineárne metódy - Medián

- Je najčastejšie používanou metódou. Medián je prostredný prvok v usporiadanej množine hodnôt. Hodnoty získame z okolia bodu. Tie zoradíme a hodnotou v strede nahradíme jas filtrovaného bodu. Táto metóda veľmi dobre odstraňuje šum impulzného (bodového) charakteru. Nevýhodou je narušovanie tenkých čiar. Okolie bodu, ktoré sa používa, nemusí byť len štvorcové. Niekedy je vhodné použiť aj okolie v tvare kríža, ktoré neporuší diagonálne čiary, prípadne v tvare písmena X, ktoré neporuší vodorovné čiary.

# Filtrácia obrazu

## Odstraňovanie šumu

Nelineárne metódy – Rotujúca maska

- Vychádza sa z toho, že sa bude spriemerovať iba tá časť okolia bodu, ku ktorej bod pravdepodobne prináleží. Z toho vyplýva, že body ležiace na hrane sa nebudú počítať do spriemerovania. Okolie musí byť homogénne. Top sa dá vyšetriť na základe rozptylu bodov.

# Filtrácia obrazu

## Detekcia hrán 1

- Keď chceme nájsť hranu v obraze, musíme hľadať zmenu jasů. Detekcia hrán v obraze je postavená na nájdení miesta, kde je táto zmena najvýraznejšia. Možnosťou, ako sa dajú jednoducho tieto zmeny nájsť, je prvá a druhá derivácia intenzity jasů v obraze.

# Filtrácia obrazu

## Detekcia hrán 2

- Pri aplikovaní prvej derivácie obrazu v smere x a y dostávame gradient, ktorý je definovaný ako vektor

$$\nabla f = \begin{bmatrix} Gx \\ Gy \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- Veľkosť tohto vektoru je

$$|\nabla f| = \sqrt{Gx^2 + Gy^2} = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

# Filtrácia obrazu

## Detekcia hrán 3

- Pre zjednodušenie výpočtu sa táto hodnota niekedy nahradzovala vynechaním odmocniny hodnotou  $G_x^2 + G_y^2$ , alebo pomocou súčtu absolútnej hodnoty  $|G_x| + |G_y|$ . Tieto náhrady sa chovajú ako derivácie, čiže sú nulové v oblastiach s konštantnou intenzitou jasu a ich hodnoty závisia na zmene. Tento gradient môžeme využiť ako informáciu pri hľadaní hrany. [10]
- Taktiež je možné ho využiť pri ostrení obrazu a to tak, že obraz upravíme, aby v ňom boli strmšie hrany.
- V praxi existuje niekoľko operátorov na hľadanie hrán založených na princípe gradientu.

# Filtrácia obrazu

## Detekcia hrán – pomocou prvej derivácie

Robertsov operátor <sup>1</sup>

- Najjednoduchší a najstarší operátor. Používa okolie len 2x2. Nevýhodou Robertsovho operátora je veľká citlivosť na šum, nakoľko okolie použité pre aproximáciu je veľmi malé.
- Veľkosť gradientu aproximuje podľa vzťahu:
- $|\nabla f(x, y)| \approx |f(x + 1, y + 1) - f(x, y)| + |f(x, y + 1) - f(x + 1, y)|$

# Filtrácia obrazu

## Detekcia hrán – pomocou prvej derivácie

Robertsov operátor 2

- Robertsov operátor v smere x :

-1	1
-1	1

$$h_1 = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \quad h_2 = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix},$$

# Filtrácia obrazu

## Detekcia hrán – pomocou prvej derivácie

Prewittovej operátor

- Gradient sa odhaduje v okolí masky 3x3 pre osem smerov. Vybraná je tá, ktorá má najväčší modul gradientu. Robí sa rozdiel hodnôt
- $f(x + 1, y) - f(x - 1, y)$ ,
- alebo pre y je to
- $f(x, y + 1) - f(x, y - 1)$ .
- Výsledná maska má tvar:

-1	0	1
-1	0	1
-1	0	1

-1	-1	-1
0	0	0
1	1	1

# Filtrácia obrazu

## Detekcia hrán – pomocou prvej derivácie

### Sobelov operátor

- Jeho koeficienty sú volené tak, aby zdôrazňovali hodnoty bližšie k strede masky. Maky môžu rotovať po  $45^\circ$  a počítať smerové derivácie nie len v smere x a y. V praxi sa však často používa pri detekcii vodorovných a zvislých čiar. Pre toto použitie postačujú nasledovné masky:

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

0	-1	-2
1	0	-1
2	1	0

# Filtrácia obrazu

## Detekcia hrán – pomocou prvej derivácie

Robinsonov operátor

- Prvé tri masky sú dané vzťahom:

-1	1	1
-1	2	1
-1	1	1

1	1	1
1	-2	1
-1	-1	-1

1	1	1
-1	-2	1
-1	-1	1

Samotný operátor je podobný ako Prewittovej operátor.

# Filtrácia obrazu

## Detekcia hrán – pomocou prvej derivácie

Robinsonov operátor

- Jeho konvolučné masky majú tvar

3	3	3
-5	0	3
-5	-5	3

3	3	3
3	0	3
-5	-5	-5

-5	3	3
-5	0	3
-5	3	3

# Filtrácia obrazu

## Detekcia hrán – pomocou druhej derivácie

- Pomocou druhej derivácie dostaneme rýchlosť zmeny hodnoty jasů. Prejavuje sa hlavne na strmých a izolovaných hranách. Môžeme ju taktiež použiť aj na detekciu izolovaných bodov. Nevýhodou je, že pri tejto operácii sa do veľkej miery zvyrazňuje aj šum.

# Filtrácia obrazu

## Detekcia hrán – pomocou druhej derivácie

Laplaceov operátor <sub>1</sub>

- Je vhodný k detekcii izolovaných bodov. Jeho operátor sa označuje  $\nabla^2$ . Operátor sa nemení v závislosti od pootočenia o násobky  $45^\circ$ , ale udáva len veľkosť hrany, nie jej smer. Výsledok aplikácie Laplaceovho operátora na funkciu  $f(x,y)$  je:

$$\nabla^2 f(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2}$$

Nevýhodou tohto operátora je, že je veľmi citlivý na šum.

# Filtrácia obrazu

## Detekcia hrán – pomocou druhej derivácie

Laplaceov operátor 2

0	-1	0
-1	4	-1
0	-1	0

Maska Laplaceovho operátora

0	1	0
1	4	1
0	1	0

Maska Laplacian of Gaussian

-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	24	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

# Možnosti úpravy v jazyku C#

## Trieda Bitmap a Image

- Prostredie .NET poskytuje viacej možností pre prácu s grafikou. Obsahuje metódy pre základné úpravy obrázkov. Výhodou je jednoduchá aplikácia algoritmu. Medzi nevýhody patrí nemožnosť pozmeniť hodnoty výpočtu a funkciu a rýchlosť. Pre načítavanie a vkladanie bodov slúžia metódy *GetPixel()* a *SetPixel()*. Výhoda je jednoduché použitie, ale najväčšia nevýhoda je rýchlosť. V porovnaní s inými možnosťami sa jedná až o niekoľkonásobne pomalšiu metódu manipulácie s obrazovými dátami.

# Možnosti úpravy v jazyku C#

## Trieda ColorMatrix 1

- Trieda ColorMatrix patrí do menného priestoru System.Drawing.Imaging. Pomocou nej môžeme veľmi pružne meniť informácie o farbách. Inštancia triedy ColorMatrix umožňuje transformáciu hodnôt ARGB každého pixelu obrázku jednotlivo. Nevýhodou tejto triedy je na prvý pohľad vysoká zložitosť a veľmi slabá dokumentácia platformy .NET. Pre transformáciu jednotlivých pixelov obrázku je nutné vytvoriť inštanciu triedy ColorMatrix. Takýto prvok obsahuje maticu o rozmere 5\*5 v tvare poľa float [],[]. Prvky 0 až 2 v oboch smeroch obsahujú transformačné hodnoty pre červenú, zelenú a modrú. Štvrtý prvok uchováva hodnotu transformácie alfa a pomocou piateho riadku môžeme pripočítať hodnoty k hodnotám farieb RGBA. Prvok v bunke 4,4 musí mať vždy hodnotu 1. Piaty stĺpec sa nepoužíva a musí obsahovať všetky hodnoty 0 (okrem spomenutej hodnoty na pozícii 4,4).

# Možnosti úpravy v jazyku C#

## Trieda ColorMatrix 2

- Príklad matice ktorá nezmení informácie o farbe:

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

# Možnosti úpravy v jazyku C#

## Trieda ColorMatrix 3

- Pri transformácií farieb sa zmenia hodnoty červenej, zelenej a modrej farby a hodnota alfa kanálu každého pixelu podľa matice farieb. Transformácia je jednoduchá, avšak je zložitejšie ju vysvetliť. Stĺpce sú určené pre výsledné hodnoty farieb. Prvý stĺpec udáva po transformácií hodnotu červenej, druhý hodnotu zelenej, tretí hodnotu modrej a štvrtý hodnotu alfa. Riadky sú určené pre farebné zložky transformovaného pixelu. Momentálne platná hodnota červenej, zelenej a modrej farby a hodnota alfa kanálu sa násobia činiteľom, zadaným v príslušnom riadku. Výsledky jednotlivých násobení sa zrátajú, pričom k výsledku sa ešte pripočítajú hodnoty posledného riadku, vynásobené číslom 255.

# Možnosti úpravy v jazyku C#

## Trieda ColorMatrix 4

- Vďaka tomu je možné ju použiť aj na nelineárne transformácie. Výsledné hodnoty RGBA z jednotlivých stĺpcov sa nakoniec zostavia do jednej výslednej farby, ktorá nám udáva výsledný pixel. Výsledný vzorec je nasledovný:
  - $\text{červená} = \text{červená} * (S_0, R_0) + \text{zelená} * (S_0, R_1) + \text{modrá} * (S_0, R_2) + \text{alfa} * (S_0, R_3) + (S_0, R_4) * 255$
  - $\text{modrá} = \text{červená} * (S_1, R_0) + \text{zelená} * (S_1, R_1) + \text{modrá} * (S_1, R_2) + \text{alfa} * (S_1, R_3) + (S_1, R_4) * 255$
  - $\text{zelená} = \text{červená} * (S_2, R_0) + \text{zelená} * (S_2, R_1) + \text{modrá} * (S_2, R_2) + \text{alfa} * (S_2, R_3) + (S_2, R_4) * 255$
  - $\text{alfa} = \text{červená} * (S_3, R_0) + \text{zelená} * (S_3, R_1) + \text{modrá} * (S_3, R_2) + \text{alfa} * (S_3, R_3) + (S_3, R_4) * 255$
- $R_x$  – x znamená číslo riadku,  $S_x$  – x znamená číslo stĺpca.

# Možnosti úpravy v jazyku C#

## Trieda ColorMatrix 5

- Výsledok pri výpočte je obmedzený na rozsah jedného bajtu, nikdy teda nie sú hodnoty menšie ako 0 a väčšie ako 255.
- Použitie spočíva v kreslení metódou DrawImage objektu Graphics. Využíva sa argument ImageAttributes. Pred týmto však musíme najskôr predať tomuto objektu pomocou metódy SetColorMatrix inštanciu typu ColorMatrix. Na príklade môžeme vidieť zosvetlenie obrázku.

# Možnosti úpravy v jazyku C#

## Použitie ukazateľov

- Medzi ďalšie možnosti ako pracovať rýchlo s bitmapou, je za pomocou použitia ukazateľov (pointrov). S nástupom programovacieho jazyka C# a platformy .NET došlo k zmenám v ich používaní. Boli vyradené, resp. prestali sa používať, a tým sa mal stať kód C# bezpečnejší. Mnoho užívateľov nabralo dojem, že boli odobrané úplne. To je však mylné. S ukazovateľmi je v C# možné pracovať v takzvanom „unsafe kóde“. Vo väčšine prípadov pri programovaní toto nie je potrebné a je doporučené sa tomu vyhýbať. Využíva sa hlavne v prípadoch, kedy potrebujeme každý kúsok výkonu navyše. Nevýhodou je, že pri ich použití nebude funkčný garbage collector a nebude sa používať typová bezpečnosť. To kladie oveľa väčšie nároky na programátora pri písaní kódu. Musí sa sám postarať o uvoľňovanie pamäte. Použitie unsafe kódu je nutné povoliť v nastavení projektu, nakoľko východiskové nastavenia ho zakazujú. Podľa oficiálnej dokumentácie modul CLR sa unsafe-nebezpečný kód nazýva neoverený kód. Preto sa nejedná nutne o nebezpečný kód, ale iba kód, ktorý nie je možnosť overiť podľa CLR. Preto je jeho použitie na vlastnú zodpovednosť a programátor musí zabezpečiť minimalizovať riziká a chyby.

# Možnosti úpravy v jazyku C#

## Použitie poľa

- Medzi ďalšie, avšak menej používané metódy je metóda pomocou poľa. Na začiatku sa obrázok načíta bod po bode do poľa. V tom sa nijako nelíši od predchádzajúcich spôsobov. Po tomto načítaní sa však všetky operácie robia iba v rámci poľa. Výhodou v tomto spôsobe použitia je rýchlosť. Operácie s poľom sú niekoľkonásobne rýchlejšie ako operácie s bitmapou. Najväčšie využitie tohto spôsobu úpravy je pri kombinácií viacerých transformácií. Pri jednej transformácii (napríklad chceme odstrániť len šum) a následného ukončenia nie je efektívny a dokonca najpomalší.

# Možnosti úpravy v jazyku C#

## Test rýchlosti algoritmov

- Pre testovanie boli vybrané 3 rôzne obrázky:
- Obrázok 1: Rozlíšenie 4288x2848, Veľkosť 1,52 MB
- Obrázok 2: Rozlíšenie 1940x1315, Veľkosť 237 kB
- Obrázok 3: Rozlíšenie 800x531, Veľkosť 86,8 kb

Metóda - číslo	Obrázok 1	Obrázok 2	Obrázok 3
GetPixel(), SetPixel()	35,08 s	7,57 s	1,23 s
Unsafe kód	6,15 s	1,32 s	0,23 s
ColorMatrix	1,52 s	0,38 s	0,07 s

# Algoritmy pre spracovanie farieb

## Odtieň šedej 1

- Konvertovanie obrázku do odtieňa šedej je pomerne ľahké. Z každého bodu v obraze zoberieme všetky farebné zložky a spojíme hodnoty. Bohužiaľ ľudské oko je však rôzne citlivé na rôzne frekvencie a nemôžeme spraviť z farieb len priemer. Musíme ich pridať do výslednej farby v určitých pomeroch. Pre červenú je to pomer 0.299, pre zelenú 0.587 a pre modrú 0.114. Výslednú hodnotu potom priradíme bodu.

# Algoritmy pre spracovanie farieb

## Odtieň šedej 2



Príklad zdrojového kódu za použitia metód  
GetPixel() a SetPixel():

```
1  int R, G, B, A;
2  Color pixelColor;
3  for (int y = 0; y < bitmapImage.Height; y++)
4  {
5      for (int x = 0; x < bitmapImage.Width; x++)
6          {
7              pixelColor = bitmapImage.GetPixel(x, y);
8              A = pixelColor.A;
9              R = (byte)((red * pixelColor.R) +
10 (green * pixelColor.G) + (blue * pixelColor.B));
11             G = B = R;
12             bitmapImage.SetPixel(x, y, Color.FromArgb(
13 (int)A, (int)R, (int)G, (int)B));
14         }
15 }
```

# Algoritmy pre spracovanie farieb

## Negatív 1

- Vytvorenie negatívu obrázku, čiže invertovanie farieb je pomerne jednoduchá úprava. Invertovanie farby nám vytvára negatív aký poznáme z analógových fotografických filmov. Úprava spočíva v tom, že musíme načítať každú farbu každého bodu v obrázku a odpočítať ju od hodnoty 255. Vzorec je:  
 $255 - \text{hodnota červená}, 255 - \text{hodnota zelená}, 255 - \text{modá}$  pre každý bod v obrázku.

# Algoritmy pre spracovanie farieb

## Negatív 2



Príklad zdrojového kódu za použitia metód  
GetPixel() a SetPixel():

```
1  int R, G, B, A;
2  Color pixelColor;
3  for (int y = 0; y < bitmapImage.Height; y++)
4  {
5      for (int x = 0; x < bitmapImage.Width; x++)
6      {
7          pixelColor = bitmapImage.GetPixel(x, y);
8          A = pixelColor.A;
9          R = (byte)(value - pixelColor.R);
10         G = (byte)(value - pixelColor.G);
11         B = (byte)(value - pixelColor.B);
12         bitmapImage.SetPixel(x, y,
13             Color.FromArgb((int)A, (int)R, (int)G, (int)B));
14     }
15 }
```

# Algoritmy pre spracovanie farieb

## Gama 1

- Ako parametre algoritmu pre zmenu gamy sú hodnoty gamy pre každú zložku RGB samostatne.
- Princíp fungovania gama filtru je vo vytvorení poľa o veľkosti 256 hodnôt pre červenú, modrú a zelenú farbu. Hodnota gamy by mala byť v rozmedzí 0,2 až 5,0.
- Základný vzorec pre výpočet je:

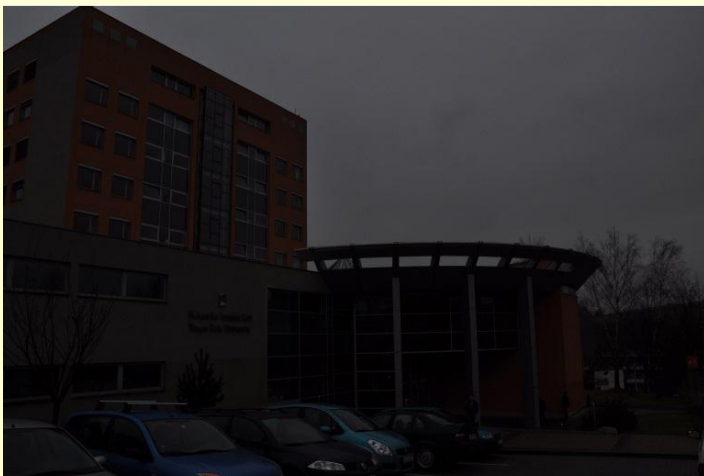
$$255 * \left( \frac{i}{255} \right)^{\frac{1}{gamma}} + 0,5$$

# Algoritmy pre spracovanie farieb

## Gama 2

Príklad zdrojového kódu za použitia metód GetPixel() a SetPixel():

```
1  int R, G, B, A;
2  Color pixelColor;
3  byte[] redGamma = new byte[256];
4  byte[] greenGamma = new byte[256];
5  byte[] blueGamma = new byte[256];
6  for (int i = 0; i < 256; ++i)
7  {
8      redGamma[i] = (byte)Math.Min(255, (int)((255.0 *
9      Math.Pow(i / 255.0, 1.0 / r)) + 0.5));
10     greenGamma[i] = (byte)Math.Min(255, (int)((255.0 *
11     Math.Pow(i / 255.0, 1.0 / g)) + 0.5));
12     blueGamma[i] = (byte)Math.Min(255, (int)((255.0 *
13     Math.Pow(i / 255.0, 1.0 / b)) + 0.5));
14 }
15 for (int y = 0; y < bitmapImage.Height; y++)
16 {
17     for (int x = 0; x < bitmapImage.Width; x++)
18     {
19         pixelColor = bitmapImage.GetPixel(x, y);
20         A = pixelColor.A;
21         R = redGamma[pixelColor.R];
22         G = greenGamma[pixelColor.G];
23         B = blueGamma[pixelColor.B];
24         bitmapImage.SetPixel(x, y, Color.FromArgb((int)A,
25         (int)R, (int)G, (int)B));
26     }
27 }
```



# Algoritmy pre spracovanie farieb

## Jas 1

- Úprava jasu patrí medzi jednoduchšie transformácie. Ak chceme zmeniť hodnotu jasu, musíme pridať, alebo ubrať určitú hodnotu z každej farebnej zložky pre každý bod obrázku. Nevýhodou takej úpravy je, že je často nenávratna. To znamená, že ak napríklad hodnota farby je blízka k hodnote 255 a my ju zvýšime nad túto hodnotu, bude automaticky orezaná na túto maximálnu hodnotu a my nenávratne stratíme informáciu o predošlej. To isté platí aj pri znížení hodnoty, kedy je hodnota menšia ako 0 automaticky nastavená na nulovú hodnotu.

# Algoritmy pre spracovanie farieb

## Jas 2



Príklad zdrojového kódu za použitia metód GetPixel() a SetPixel():

```
1         for (int y = 0; y < bitmapImage.Height; y++)
2         {
3             for (int x = 0; x < bitmapImage.Width; x++)
4             {
5                 pixelColor = bitmapImage.GetPixel(x, y);
6                 A = pixelColor.A;
7                 R = pixelColor.R + brightness;
8                 if (R > 255)
9                 {
10                    R = 255;
11                }
12                else if (R < 0)
13                {
14                    R = 0;
15                }
16                bitmapImage.SetPixel(x, y, Color.FromArgb(A, R, G, B));
17            }
18        }
```

.... pre hodnoty G a B je výpočet rovnaký ako pre R

# Algoritmy pre spracovanie farieb

## Kontrast 1

- Úprava kontrastu patrí medzi ďalšie často používané úpravy. Transformácia
- upravuje hodnoty medzi bodmi. Môžeme rozdiel buď zväčšiť, alebo zmenšiť. Pri použitej transformácii sa prednastavená hodnota pre rozdiel pohybuje v rozmedzí -100 až 100. Výsledná hodnota sa určuje zo vzťahu:

$$\left(\frac{100 + \textit{kontrast}}{100}\right)^2$$

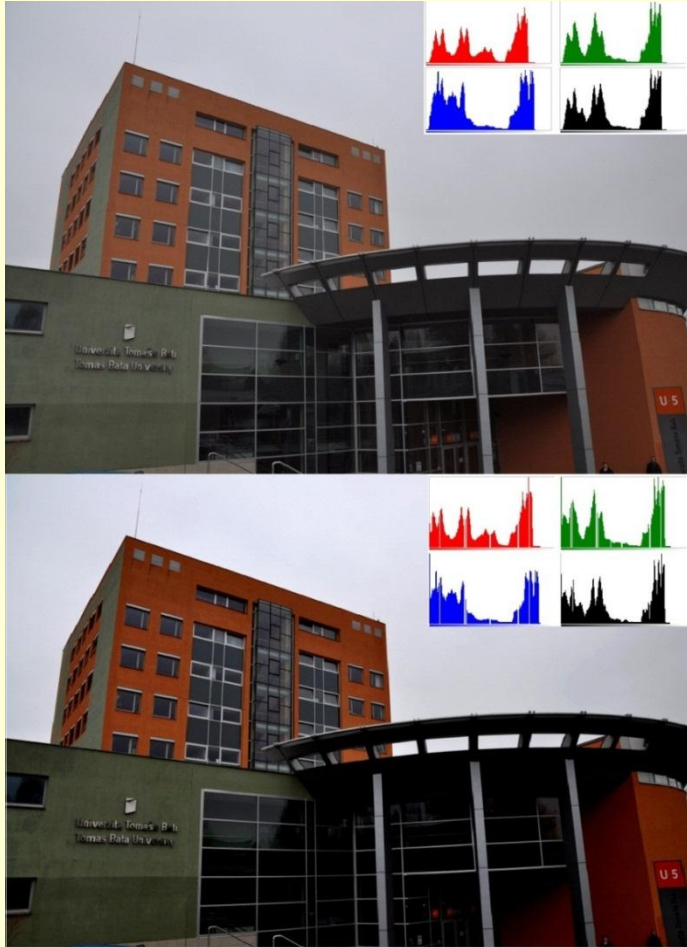
# Algoritmy pre spracovanie farieb

## Kontrast 2

- Postup výpočtu je taký, že zoberieme každý pixel v obraze a každú farebnú zložku vydělíme hodnotou 255, aby sme dostali hodnotu medzi 1 a 0. Potom odpočítame hodnotu 0,5. Následne vynásobíme hodnotu hodnotou kontrastu. Všetkým zložkám, ktoré budú mať hodnoty záporne sa kontrast zníži, zatiaľ čo kladným hodnotám sa kontrast zvýši. Potom opäť pripočítame hodnotu 0,5 a prevedieme ju na rozsah 0-255. Pokiaľ je výsledná hodnota väčšia ako 255 je hodnota nastavená na túto maximálnu hodnotu. Pokiaľ je menšia ako 0 je nastavená na 0.

# Algoritmy pre spracovanie farieb

## Kontrast 3



Príklad zdrojového kódu za použitia metód GetPixel() a SetPixel():

```
1  for (int y = 0; y < bitmapImage.Height; y++)
2  {
3      for (int x = 0; x < bitmapImage.Width; x++)
4      {
5          pixelColor = bitmapImage.GetPixel(x, y);
6          A = pixelColor.A;
7          R = pixelColor.R / 255.0;
8          R -= 0.5;
9          R *= kontrast;
10         R += 0.5;
11         R *= 255;
12         if (R > 255)
13         {
14             R = 255;
15         }
16         else if (R < 0)
17         {
18             R = 0;
19         }
20         .... pre hodnoty G a B je výpočet rovnaký ako pre R
21         bitmapImage.SetPixel(x, y,
22             Color.FromArgb((int)A, (int)R, (int)G, (int)B));
23     }
```

# Grafické filtre

- Grafické filtrácie sa používajú napríklad na vyhladzovanie obrazu, ostrenie obrazu, odstraňovanie šumu, hľadanie nespojitosti a hrán.
- Filtrácie zahrňujúce okolie fungujú väčšinou na princípe konvolúcie.

# Grafické filtre

- Konvolučná maska
  - Konvolučná maska je v podstate matica o určitých rozmeroch. Čím väčšia matica je, tým máme menej bodov, ktoré budú mať vplyv po okrajoch. Princíp funkcie je, že pixel v strede obklopuje osem ďalších s určitou váhou (pri maske 3x3). Celková hodnota matice sa delí faktorom a prípadne je možné k nemu pripočítať ľubovoľnú hodnotu. Zvyčajne hodnota faktoru je hodnota všetkých hodnôt v matici. Pri konvulúcií sa ešte musí riešiť úprava okrajových bodov. V praxi sa používajú dve metódy. Prvá zväčší maticu o polovičku a doplní nulami, prípadne hodnoty doplní zrkadlovým skopírovaním. Druhá a častejšie používaná možnosť je, že sa výpočet robí len v oblasti dosahu masky a nespracovaný okraj sa oreže, čoho výsledkom je zmenšený obrázok.

# Grafické filtre

## Vyhladzovanie obrazu

- Vyhladzovanie obrazu je vhodné pri veľmi veľkých detailoch, alebo ostrých
- prechodoch. Je vhodné aj na odstránenie šumu z obrazu.



V strede masky je predvolená hodnota 8 a po okrajoch hodnoty 1. Faktor je v tomto prípade nastavený na 16 a prírastok je 0.

# Grafické filtre

## Gaussove vyhladzovanie

- Gaussovo vyhladzovanie je veľmi podobné normálnemu vyhladzovaniu obrazu.
- Rozdiel je, že Gaussovo vyhladzovanie dáva prirodzenejšie a živšie rozostrenie. Rozloženie hodnoty v maske vytvára takzvaný kruhový efekt, kde pixely ďalej od okraja majú menšiu váhu.



# Grafické filtre

## Ostrenie obrazu

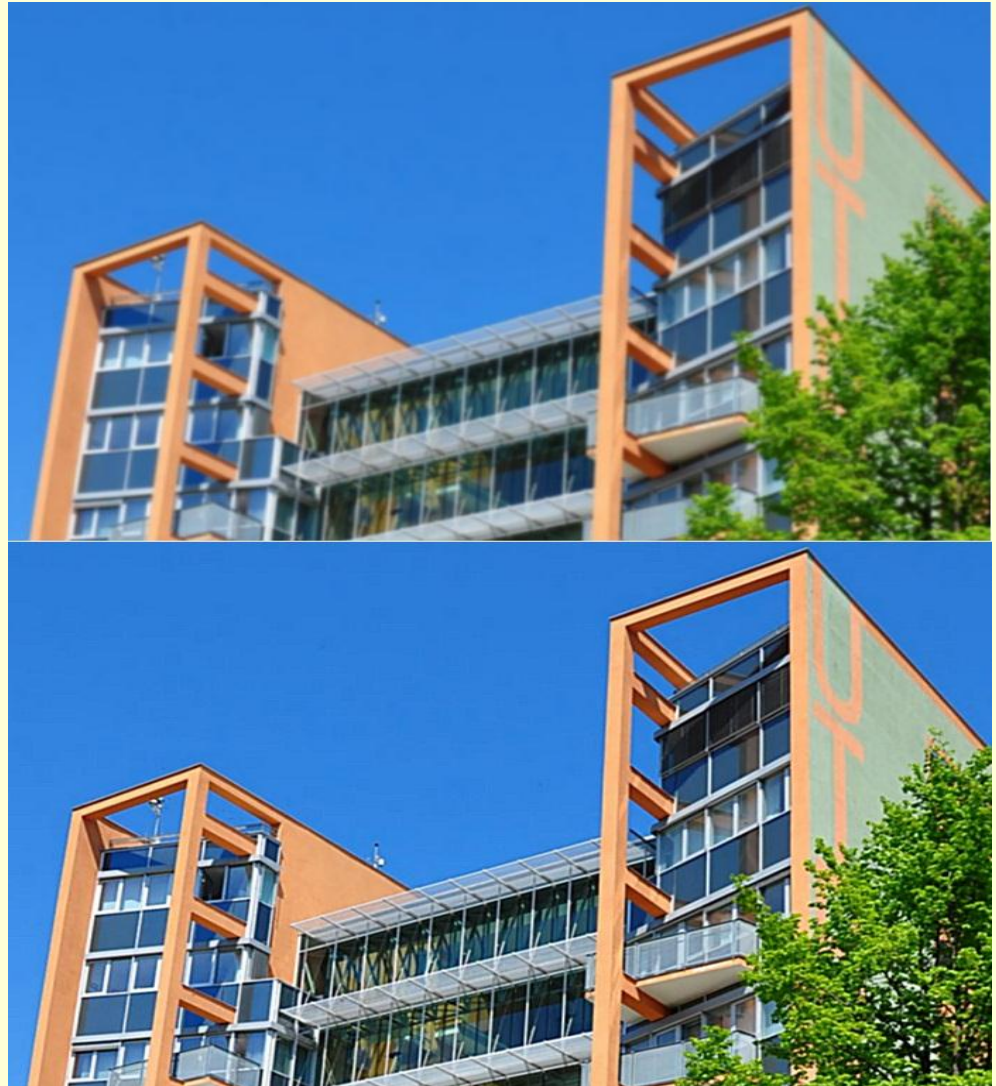
- Ostrenie obrazu je opačná funkcia, ako sme pri vyhladzovaní. V prípade aplikácie
- ostrenia na obrázok, kde sme použili gaussove vyhadzovanie zistíme, že sa jedná o takmer pravý opak. Princíp je vo zvyšovaní rozdielu medzi hodnotami susedných pixelov. Masky odoberá len vo vertikálnom a horizontálnom smere. Ak potrebuje zväčšiť stupeň ostrosti, zväčšíme hodnotu stredného bodu masky.



# Grafické filtre

## Stredové ostrenie obrazu

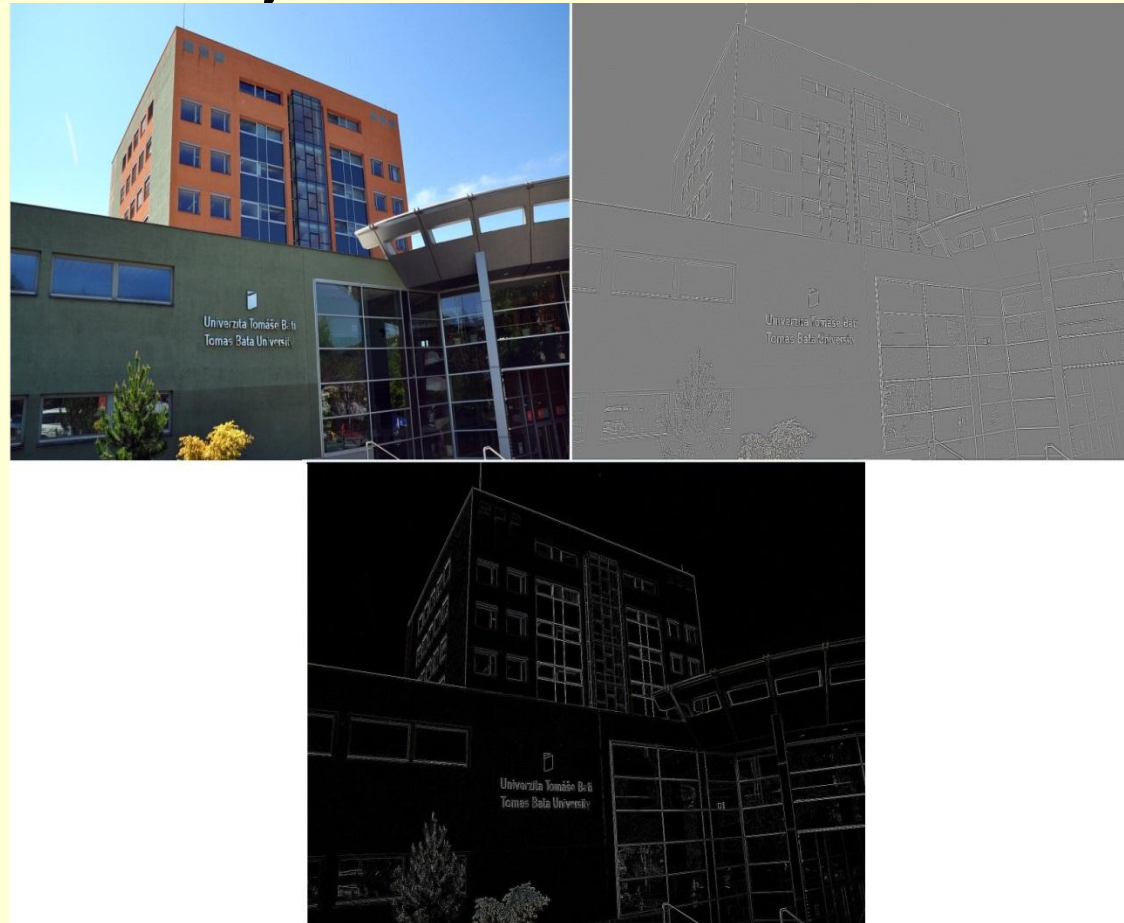
- Stredové ostrenie obrazu funguje na princípe klasického ostrenia, ale odoberá hodnoty zo všetkých susedných bodov v maske.



# Grafické filtre

## Vytlačený vzor

- Účelom tejto transformácie je poskytnúť dojem, že obraz je plastický a je akoby vytlačený do povrchu materiálu – reliéf obrázku. Princíp je vo zvýraznení hrán. Využíva sa Laplaceov operátor. Konvolučná maska má hodnotu prírastku k deliteľu až 127. To nám zaručí zosvetlenie obrazu, inak by bol výsledok čierny obrázok. Aplikácia filtra môže byť vo viacerých smeroch.



Príklad na použitie Laplacovho operátora  
(pozitívna/negatívna maska)

# Grafické filtre

## Detekcia hrán

- Spôsobov ako hľadať hrany v obrázku je veľa. Pomocou konvolučnej masky sa
- jedná o veľmi jednoduchý spôsob. Filtre na detekciu hrán nehýbu s hodnotou pixelu v strede, ale iba s pixelom, ktoré ho obklopujú. Funguje podobne ako filtre s Laplacovým operátorom, avšak tieto zvyšujú účinok. Podobne môže byť maska aplikovaná vo viacerých smerov.

