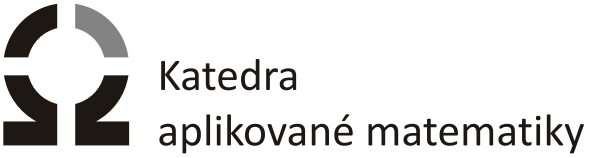
Katedra
aplikované matematiky

Počítačová cvičení
Škola matematického modelování
2016

Petr Beremlijski, Rajko Ćosić, Marie Sadowská, Matyáš Theuer

Počítačová cvičení
Škola matematického modelování

Petr Beremlijski, Rajko Ćosić, Marie Sadowská, Matyáš Theuer



Katedra aplikované matematiky
Fakulta elektrotechniky a informatiky
VŠB - Technická univerzita Ostrava
2016

Úvod

Tento text je určen pro účastníky semináře Škola matematického modelování (http://skomam.vsb.cz) a slouží jako pomůcka k úlohám, které řeší studenti v průběhu tohoto semináře. Tento seminář, pro který používáme zkratku ŠKOMAM, organizuje Katedra aplikované matematiky (http://am.vsb.cz) Fakulty elektrotechniky a informatiky Vysoké školy báňské – Technické univerzity Ostrava jednou ročně již od roku 2005. Tento rok probíhá již 12. ročník tohoto semináře.

Pro počítačové řešení úloh, kterými se budeme zabývat, jsme zvolili komerční systém Matlab. Matlab se skládá z několika součástí. Jde zejména o programovací jazyk zaměřený na numerické výpočty a vývoj numerických algoritmů. Matlab obsahuje také knihovnu funkcí pro řešení řady numerických úloh. A důležitou složkou Matlabu je grafické prostředí pro interaktivní zadávání příkazů. Podrobný popis tohoto jazyka je k dispozici v [1]. Pokud čtenář naši knihy nemá Matlab k dispozici, může použít systém Octave, který se značně podobá Matlabu a je volně k dispozici na webové adrese https://www.gnu.org/software/octave.

Tento seminář je pořádán s finanční podporou Fakulty elektrotechniky a informatiky (http://www.fei.vsb.cz). Nad akcí převzala záštitu Jednota českých matematiků a fyziků (http://jcmf.vsb.cz).

Cvičení 1: Matlab – nástroj pro matematické modelování

Abychom se mohli věnovat numerickému řešení matematických úloh, potřebujeme vhodné prostředí, které nám to umožní. A tak jako fyzici či chemikové mají své laboratoře, mají i numeričtí matematici svou Maticovou laboratoř[1] - Matlab. Podrobně se tomuto pracovnímu prostředí a jeho příkazům věnuje přiložený Matlabovský slabikář [2] nebo také úvod textu [1]. My si v tomto textu uvedeme pouze stručný přehled matlabovských proměnných a příkazů, které budeme potřebovat.

Prostředí

- *help, demos, intro, who, whos, clear, size, length*

Proměnné

- Skaláry
- Vektory
- Matice

Příkazy

- Skalární funkce - *sin, cos, tan, cot, exp, log, abs, sqrt, round*
- Vektorové funkce a generování vektorů - *max, min, sort*
- Maticové funkce a generování matic - *det, rand, ones, zeros, eye*
- Skalární operace - $+,\ -,\ *,\ /, \hat{}$
- Maticové a vektorové operace - $+,\ -,\ *,\ '$(transponování), $\backslash$ ($A\backslash v = x \Leftrightarrow Ax = v$)
  Operace „po prvcích“ - $.*,\ .\hat{},\ ./$
- 2D grafika (vykreslení grafů funkcí jedné proměnné) - *plot, hold on, hold off, figure*
- 3D grafika (vykreslení grafů funkcí dvou proměnných) - *meshgrid, mesh, contour, hold on, hold off, figure*
- Řídící příkazy - *if* (podmíněný příkaz), *for, while* (příkazy cyklu se známým počtem opakování a podmínkou na začátku)
- Relace a logické operace - $<,\ >,\ <=,\ >=,\ ==,\ \sim=,\ \&,\ |,\ \sim$
- Skripty a funkce - *function*

[1] MATrix LABoratory

Vše si nyní vyzkoušíme při řešení následujících úloh.

Úkol 1.1 Kolik členů harmonické řady[2] musíme nejméně sečíst, aby tento částečný součet řady měl hodnotu alespoň 10 (15, 20)? ▲

Úkol 1.2 Zkuste odhadnout s využitím Matlabu součet řady

$$\sum_{n=1}^{\infty} \frac{1}{n\,(n+1)}.$$

▲

Úkol 1.3 Sestrojte grafy následujících funkcí:

- $f(x) = x^2$
- $f(x) = \sqrt{1-x^2}$
- $f(x) = x^2 \sin\left(\frac{1}{x^2}\right)$
- $f(x) = |x|$ ▲

Úkol 1.4 Sestrojte grafy následujících funkcí:

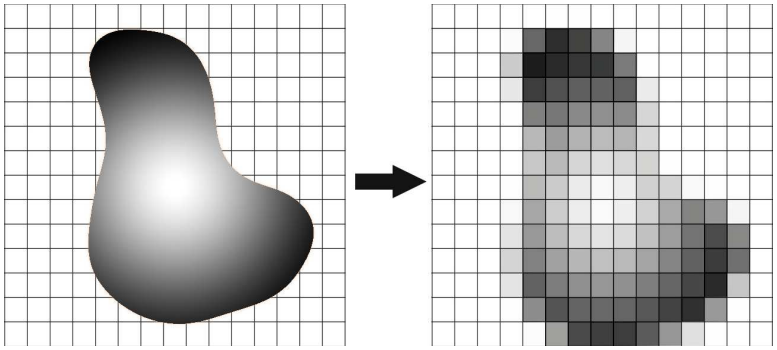
- $f(x,y) = x^2 + y^2$
- $f(x,y) = \sqrt{x^2+y^2}$
- $f(x,y) = (x^2+y^2)\sin\left(\frac{1}{x^2+y^2}\right)$
- $f(x,y) = \sqrt{|xy|}$ ▲

[2] Řadou (reálných čísel) rozumíme výraz $a_1 + a_2 + \cdots + a_n + \cdots = \sum_{n=1}^{\infty} a_n$, kde pro každé $n \in \mathbb{N}$ je $a_n \in \mathbb{R}$. Harmonickou nazýváme řadu $\sum_{n=1}^{\infty} \frac{1}{n}$.

V dobách analogových fotoaparátů se vyfocený obraz v každém okamžiku uchovával skutečně jako obraz. Ať už na negativu filmu, nebo po vyvolání na fotografickém papíře či diapozitivu. Dnes je tomu jinak. Při zmáčknutí spouště digitálního fotoaparátu se aktuální scéna před objektivem zachytí pomocí snímacího čipu a uloží jako soubor čísel na paměťovou kartu. I kdybychom tuto kartu rozebrali, obrázky na ní neuvidíme. K jejich zobrazení potřebujeme opět nějaké digitální zařízení, které umí obraz uložený v jedničkách a nulách převést do viditelné formy.

Matice obrazu

Pro jednoduchost se nejprve zabývejme černobílými obrázky - přesněji obrázky ve stupních šedé. V okamžiku exponování dojde k zachycení snímané scény na čip, který je tvořen soustavou miniaturních fotodiod uspořádaných do řádků a sloupců. V závislosti na osvětlení příslušné části čipu každá z diod vyprodukuje určitý náboj, který je změřen, a jeho hodnota je zaznamenána ve formě čísla. V případě JPEG obrázku se jedná o celé číslo od 0 do 255,[3] přičemž hodnota 0 odpovídá černé a hodnota 255 bílé. Proces rozdělení spojitého obrazu na diskrétní hodnoty v jednotlivých oddělených bodech se nazývá kvantování a vzorkování.



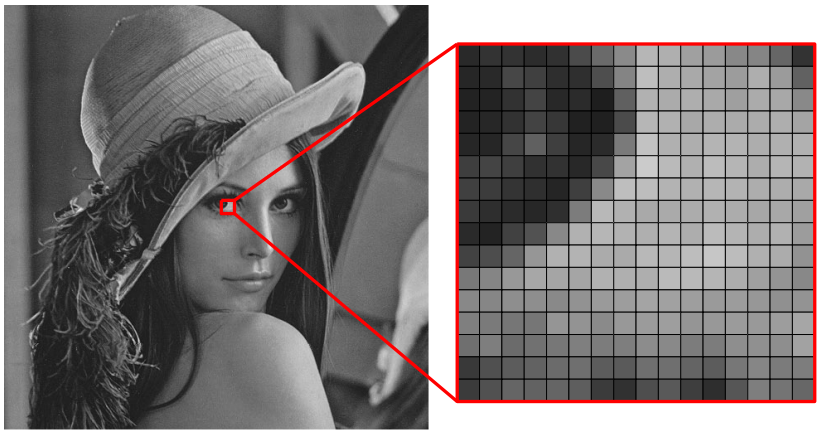
Obrázek 1: Příklad kvantování a vzorkování

Pomineme-li ve skutečnosti binární uložení obrazu, můžeme si jeho digitální podobu představit jako obecně obdélníkovou tabulku čísel, ve které každá hodnota reprezentuje jas určité malé plošky (pixelu) v zachyceném obraze. Tuto tabulku budeme označovat pojmem matice obrazu:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}, \text{kde } a_{i,j} \in \{0, 1, 2, ..., 255\}.$$

Příklad části matice obrazu můžeme vidět na obr. 2.

[3] Tyto hodnoty odpovídají osmibitové reprezentaci čísla.

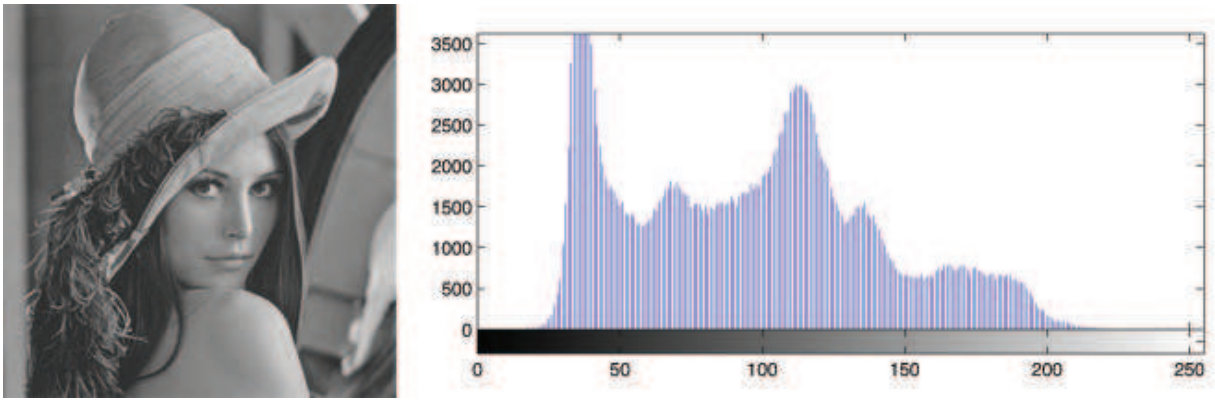


41	51	51	44	49	75	106	139	182	175	160	155	137	134	102	51
38	41	71	64	46	47	78	133	190	174	169	163	156	174	155	97
34	35	50	68	40	43	30	96	177	171	174	169	159	171	168	137
35	39	50	67	55	33	32	75	182	177	168	173	171	172	174	164
38	38	70	96	63	38	43	122	194	176	174	180	179	173	176	177
62	68	63	49	50	40	72	163	203	186	176	188	181	173	181	178
64	60	52	47	37	63	138	190	189	179	177	182	179	173	173	169
58	46	43	39	47	124	184	177	169	170	172	177	172	167	168	160
42	35	65	82	136	177	182	178	175	180	173	187	180	173	177	156
66	77	101	150	177	179	171	170	184	181	189	198	188	176	172	146
120	133	152	167	165	165	177	175	174	186	185	181	173	160	148	136
137	134	142	142	137	146	154	157	164	161	158	154	156	151	145	147
127	122	118	118	149	148	138	152	155	158	149	152	155	153	157	166
111	126	107	113	119	127	124	126	140	137	124	124	143	144	142	162
57	80	92	98	107	111	92	90	111	111	100	92	119	134	123	125
59	81	101	98	101	94	59	49	78	89	62	46	86	126	116	103

Obrázek 2: Část matice obrazu

Histogram

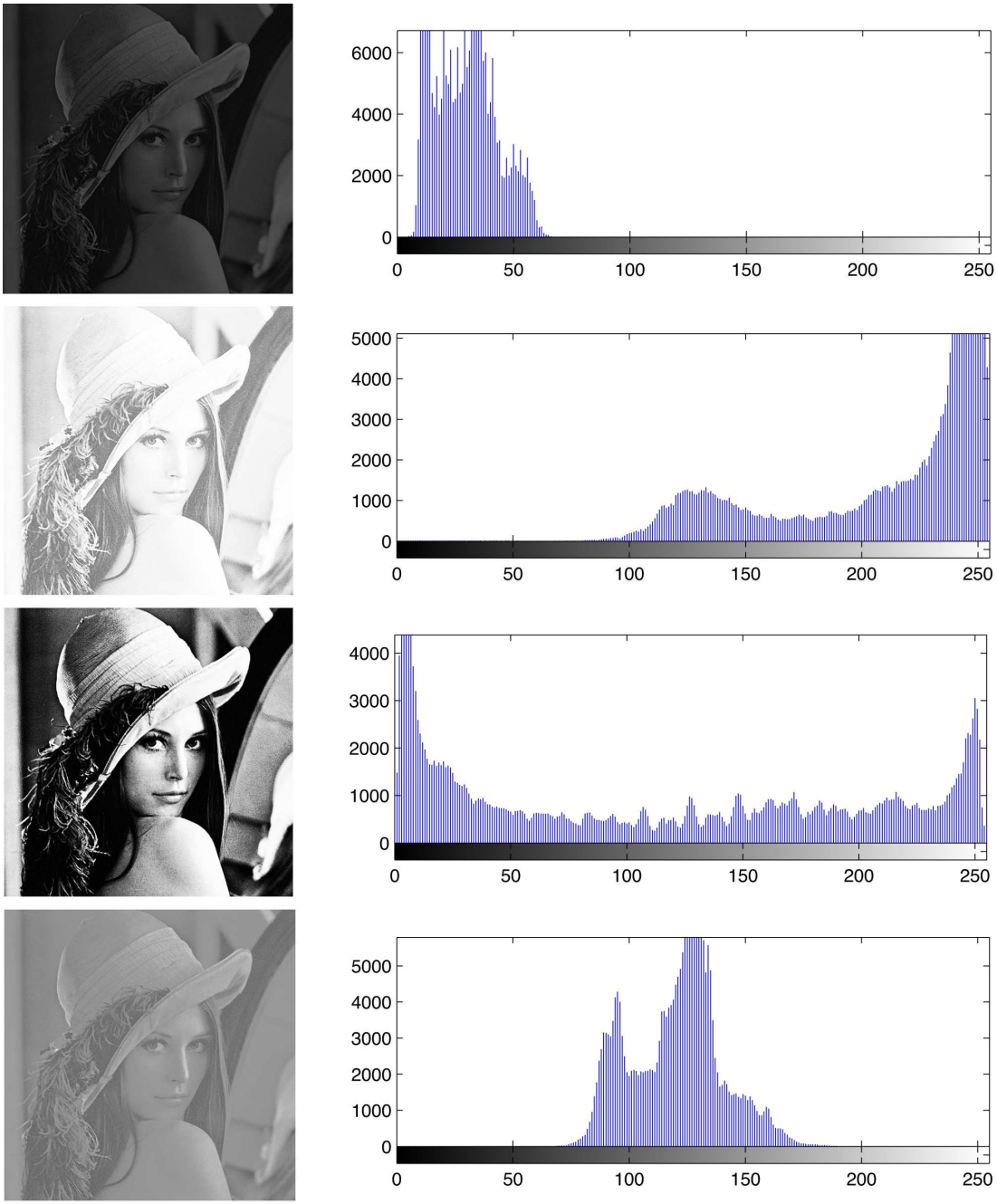
Máme-li obrázek ve stupních šedé (číslech od 0 do 255), může nás zajímat, kolikrát se v obrázku jednotlivé stupně (čísla) vyskytují. Graf četnosti výskytu jednotlivých hodnot v obrázku se nazývá histogram.



Obrázek 3: Obrázek a jeho histogram

Přestože obrázek není svým histogramem jednoznačně definován, můžeme z histogramu

o původním obrázku hodně vyčíst. Tmavé obrázky mají v histogramu velké hodnoty nahromaděné v levé části grafu. Naopak světlé obrázky mají v histogramu velké hodnoty v pravé části grafu. Histogram obrázků s nízkým kontrastem vypadá jako jeden relativně úzký „kopec“, zatímco obrázky s vysokým kontrastem mají většinou dva „kopce“, každý na opačné straně grafu.



Obrázek 4: Příklad různých obrázků a jejich histogramů

Jednoduché úpravy

Jak jsme si řekli, obrázky jsou v počítači uloženy jako matice čísel, takže místo manipulace s barevnými ploškami nám stačí provádět operace s čísly. Nyní si v jednoduchosti popíšeme některé základní operace s maticemi obrázků. Předpokládejme, že máme obrázek ve stupních šedé, který má 512×512 pixelů. Matice takového obrazu, kterou označíme I_{orig}, má 512 řádků a 512 sloupců. Dohromady je to 262144 čísel, takže je jasné, že nebudeme počítat s každou hodnotou ručně, ale nastavíme nějaké pravidlo, které počítači řekne, co má s jednotlivými hodnotami udělat.

- Ořez obrázku provedeme jednoduše tak, že vybereme jen omezený rozsah řádků a sloupců původní matice. Například pro výřez pravé horní čtvrtiny obrázku I_{orig} vezmeme pouze 1. až 256. řádek a 257. až 512. sloupec. Dostaneme tam matici I_{crop}, která má 256 řádků a 256 sloupců.

- Zesvětlení obrázku provedeme například tak, že k hodnotě každého pixelu v matici I_{orig} přičteme nějaké kladné číslo, například 100. Touto operací se však může stát, že se některé hodnoty ocitnou mimo rozsah 0 až 255. Opravíme to například tak, že všechny hodnoty, které vyjdou větší než 255, zmenšíme právě na 255. Ve všech bodech, kde došlo k takovém ořezu hodnot ovšem ztrácíme obrazovou informaci.

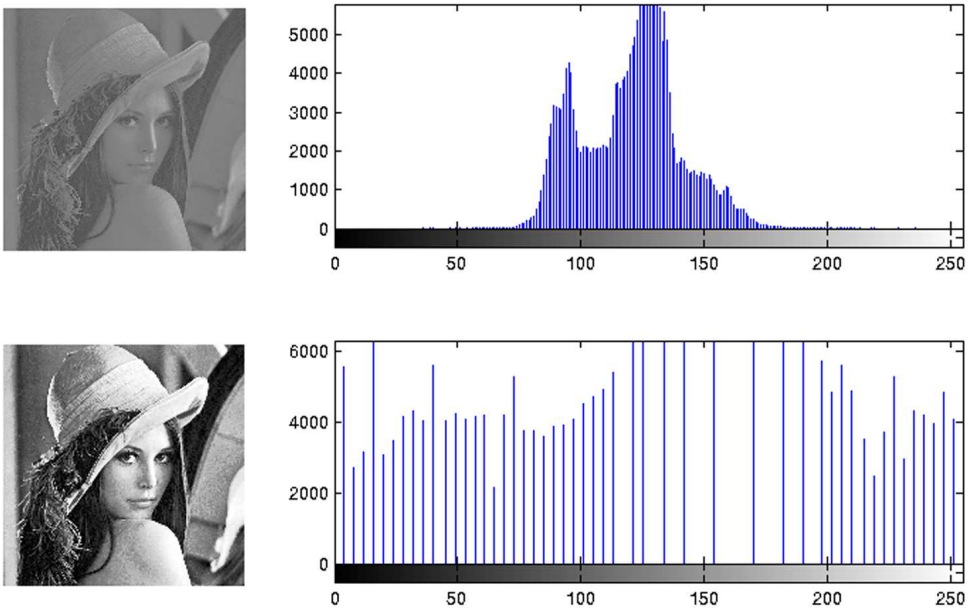
- Ztmavení obrázku můžeme provést obdobně jako zesvětlení, ale musíme zkontrolovat, zda některé hodnoty nevyšly záporné. Pokud ano, nastavíme je na 0. Další možností je vynásobit všechny hodnoty matice obrazu nějakým číslem z intervalu $(0, 1)$. Zde nehrozí, že bychom se dostali mimo rozsah 0 až 255, ale může se stát, že výsledkem nebudou jen celá čísla, a tak je potřeba každou výslednou hodnotu zaokrouhlit na nejbližší celé číslo.

- Negativ obrázku je obraz, který má obrácenou reprezentaci čísel. V klasickém obrázku odpovídá 0 černé a 255 bílé hodnoty mezi těmito čísly odpovídají různým stupňům šedé od tmavé po světlou. V negativním zobrazení je to naopak. Abychom nemuseli přeprogramovat zobrazovací zařízení k obrácení stupnice, můžeme obrátit stupnici přímo v našem obrazu. Hodnoty 0 změníme na 255 a naopak. Všechny hodnoty tedy vypočítáme tak, že původní hodnotu odečteme od čísla 255 a dostaneme právě negativ původního obrázku, který již zobrazíme klasicky.

- Prahování je jednoduchá operace, kde zvolíme $t \in (0, 255)$ (jednoduchý práh). Jednotlivé prvky matice I_{prah} nastavíme na 0, pokud příslušná hodnota matice I_{orig} je menší než t, a na 255, pokud je příslušná hodnota matice I_{orig} je větší nebo rovna t.

- Vyrovnání histogramu je metoda, která slouží k automatickému upravení kontrastu tak, aby byla četnost jednotlivých hodnot rozdělena přibližně rovnoměrně. Tato metoda je automatická a nezohledňuje, o jaký obrázek se jedná, takže výsledek může působit poněkud uměle.

Obrázek 5: Negativ obrázku (vlevo), obrázek po prahovaní s parametrem $t = 50$ (uprostřed) a $t = 150$ (vpravo)



Obrázek 6: Příklad vyrovnání histogramu obrázku s nízkým kontrastem

Barevné obrázky

Pro reprezentaci barevného obrazu se používají matice rovnou tři - jedna pro každou barevnou složku RGB.[4] Výsledný obraz pak vznikne složením těchto tří obrazů. Hodnota každého pixelu obrazu se tedy skládá ze tří čísel, a tak se tato ploška zobrazuje v jedné z 256^3 možných barev.

Pro jiný formát souboru může matice obrazu obsahovat i jiná čísla než celá čísla od 0 do 255. Můžeme například použít reálná čísla z intervalu $(0, 1)$, kdy řekneme, že černá je 0

[4]RGB je systém rozložení barevného obrazu na červený (R), zelený (G) a modrý (B) kanál.

a bílá odpovídá hodnotě 1. Pokud uložíme hodnoty každého pixelu ve formátu 14-bitových čísel, dostaneme až 16384^3 barev. Běžné zobrazovací zařízení však neumí takový rozsah barev zobrazit, a proto ve většině případů stačí ukládat obrázky do osmi bitů.

Obrázek 7: JPEG obrázek složený z RGB kanálů

Úkol 2.1 Načtěte svůj obrázek do Matlabu pomocí příkazu `imread`. Zobrazte tento obrázek pomocí funkce `imshow` a zobrazte jeho jednotlivé složky R, G a B. Převeďte tento obrázek do stupňů šedé pomocí funkce `rgb2gray` a uložte pomocí funkce `imwrite`. ▲

Úkol 2.2 Proveďte libovolný výřez vašeho černobílého obrázku obsahující 512×512 pixelů a uložte tento oříznutý obrázek. Zobrazte histogram tohoto obrázku pomocí funkce `imhist`. ▲

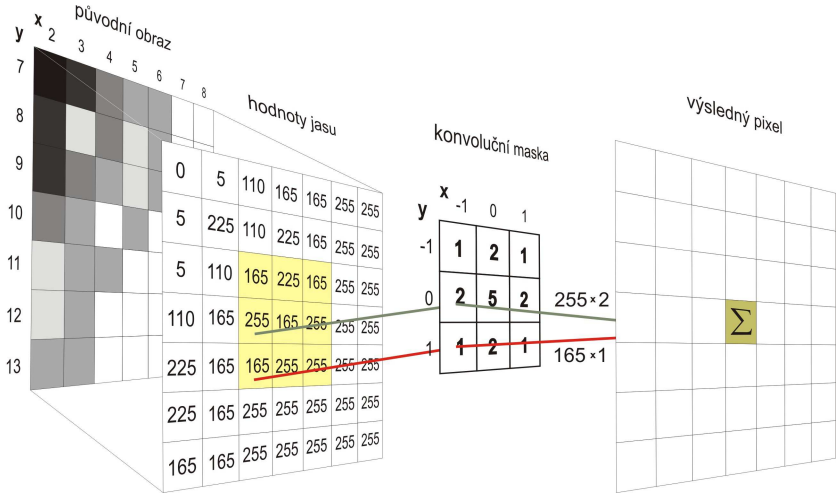
Úkol 2.3 Proveďte zesvětlení a ztmavení vašeho obrázku a vytvořte jeho negativ. Proveďte prahování s prahem $10, 100, 150, 200$. ▲

Úkol 2.4 Proveďte zvýšení kontrastu vašeho obrázku pomocí vyrovnání histogramu. ▲

Na závěr poznamenejme, že mnohem více lze o problematice zpracování obrazu nalézt v textu [3].

Cvičení 3: Konvoluce

V tomto cvičení si zkusíme aplikovat na obrázky diskrétní konvoluci. Diskrétní konvoluce je zobrazení, do kterého vstupuje obrázek a takzvaná konvoluční maska. Konvoluční maska specifikuje, co konkrétně konvoluce s obrázkem provede. Protože se v počítačích obrázky vyskytují nejčastěji ve formě matice, jejíž každá hodnota odpovídá jasu v daném pixelu, nepřekvapí nás, když i konvoluční maska bude ze stejného světa. Diskrétní konvoluce tedy vezme matici obrázku a každému bodu přiřadí novou hodnotu podle následujícího schématu:



Obrázek 8: Schéma diskrétní konvoluce

Jednoduše řečeno: položíme masku na obrázek tak, aby střed masky byl v bodě, kde konvoluci počítáme, přenásobíme každou hodnotou v masce příslušnou hodnotu obrázku a potom vše sečteme.[5]

Pomocí konvoluce jsme schopni odstranit z obrázků šum nebo zvýraznit hrany, což je hojně využíváno, chceme-li, aby počítač dokázal sám rozpoznat objekty na obrázku (třeba při zpracování dat z průmyslových kamer).

Čím je maska větší, tím větší okolí bodu nám může zasáhnout do výpočtu. Názorně je to vidět v následujícím příkladu, kdy byla nejdříve použita maska

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

která vlastně novou hodnotu spočítá jako průměr hodnot v okolních bodech. Poté na stejný obrázek aplikujeme masku větší, ale opět s hodnotami, které zprůměrují hodnoty obrázku:

[5]Konvoluci vypočteme pomocí předpisu $(f * h)(x, y) = \sum_{r=-k}^{k} \sum_{s=-k}^{k} f(x + r, y + s)\, h(r, s)$, kde funkce f popisuje obrázek a funkce h masku.

$$\frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Obrázek 9: Lena před aplikací konvoluce (vlevo), po konvoluci s maskou 3×3 (uprostřed) a po konvoluci s maskou 5×5 (vpravo)

Jak si můžeme všimnout, naše maska zprůměrováním hodnot na okolí obraz rozmazala, a to tím více, čím větší okolí bodu se do průměru započítalo. Kdyby byla maska stejně veliká jako obrázek, tak by celý výsledný obrázek byl jen v jedné barvě.

V minulém cvičení jsme si ukázali, jak se v Matlabu pracuje s maticemi obrazu. Nyní si zkusíme naprogramovat diskrétní konvoluci podle následujícího algoritmu.

Algoritmus 1: Diskrétní konvoluce

```
obrazek = nacti_obrazek()
N,M = rozmery(obrazek)
maska = matice(n,n)
for i=1..N
  for j=1..M
    restrikce = obrazek((i-n/2):(i+n/2),(j-n/2):(j+n/2))
    novy_obrazek(i,j) = suma_prvku(maska .* restrikce)
  end
end
uloz_obrazek(novy_obrazek)
```

Abychom se vyhnuli přečnívání masky „do prázdna“ při počítání hodnot u krajních bodů, nebudeme konvoluci počítat v bodech, ve kterých by maska přečnívala přes okraj obrázku. Vytvoříme si vlastně takový rámeček, na kterém výpočet konvoluce prostě vynecháme.

Úkol 3.1 Naprogramujte diskrétní konvoluci podle návodu výše a na obrázcích Lena.png a bricks.jpg (můžete použít i své obrázky z minulého cvičení) otestujte následující konvoluční masky. Obrázky načítejte v odstínech šedi.

- $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ ... odstraní šum
- $\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ ... Gaussova maska
- $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ ... Laplaceova maska
- $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ ... zvýrazní hrany
- $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ ... zvýrazní svislé hrany
- $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$ ... zvýrazní vodorovné hrany

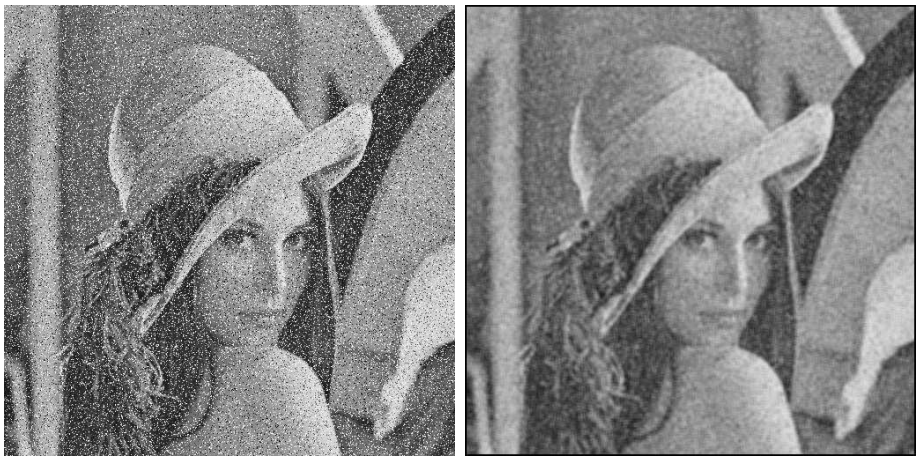
U posledních čtyř masek přičtěte ke každému bodu ještě hodnotu 128, ať obrázky nejsou moc tmavé. ▲

Pro ilustraci praktického využití diskrétní konvoluce použijeme na zašuměný obrázek konvoluční masku

$$\frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

viz obr. 10.

Jak je vidět, použili jsme stejnou masku, na které jsme si ukazovali rozmazání obrázku. Při odstraňování šumu pomocí diskrétní konvoluce se rozmazání obrázku pravděpodobně nevyhneme. Čím větší šum potřebujeme z obrázku odfiltrovat, tím více budeme muset obrázek rozmazat.



Obrázek 10: Lena před odstraněním šumu (vlevo) a Lena po odstranění šumu (vpravo)

Velmi podobný výsledek bychom obdrželi i při použití Gaussovy masky

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix},$$

viz obr. 11.

Obrázek 11: Lena před odstraněním šumu (vlevo) a Lena po odstranění šumu pomocí Gaussovy masky (vpravo)

Další informace o využití konvoluce pro zpracování obrazu může zvídavý čtenář získat v textu [3].

Reference

[1] T. Kozubek, T. Brzobohatý, V. Hapla, M. Jarošová, A. Markopoulos: *Lineární algebra s Matlabem*. Text vytvořený při realizaci projektu „Matematika pro inženýry 21. století“, Vysoká škola báňská - Technická univerzita Ostrava (2012). (http://mi21.vsb.cz/modul/linearni-algebra-s-matlabem)

[2] K. Sigmon: *Matlab Primer*. University of Florida (1993).

[3] E. Sojka, J. Gaura, M. Krumnikl: *Matematické základy digitálního zpracování obrazu*. Text vytvořený při realizaci projektu „Matematika pro inženýry 21. století“, Vysoká škola báňská - Technická univerzita Ostrava (2012). (http://mi21.vsb.cz/modul/matematicke-zaklady-digitalniho-zpracovani-obrazu)

Apendix A: Matice

Definice A.1 Nechť jsou dány prvky $a_{1,1}, a_{1,2}, \ldots, a_{m,n}$ z dané množiny $\mathcal{F}$. Matice typu (m, n) (stručně $m \times n$ matice) je obdélníková tabulka

$$A = \begin{bmatrix} a_{1,1} & \ldots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \ldots & a_{m,n} \end{bmatrix},$$

která má $m \cdot n$ prvků $a_{i,j}$ uspořádaných do m řádků r_i^A a n sloupců s_j^A, takže

$$A = \begin{bmatrix} r_1^A \\ \vdots \\ r_m^A \end{bmatrix} = \left[s_1^A, \ldots, s_n^A\right],$$
$$r_i^A = \left[a_{i,1}, \ldots, a_{i,n}\right], \quad s_j^A = \begin{bmatrix} a_{1,j} \\ \vdots \\ a_{m,j} \end{bmatrix}.$$

Stručně píšeme též $A = [a_{i,j}]$.