VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY | KATEDRA APLIKOVANÉ MATEMATIKY

Počítačová cvičení
Škola matematického modelování
2021

Petr Beremlijski, Michal Béreš

Rajko Ćosić, Marie Sadowská

Katedra aplikované matematiky
Fakulta elektrotechniky a informatiky
VŠB - Technická univerzita Ostrava
2021

Předmluva

Základní pravidlo matematického života zní: pokud vás svět postaví před obtížný problém, pokuste se nejdříve vyřešit jednodušší problém a doufejte, že jednoduchá verze bude natolik blízká tomu původnímu problému, že vám to svět odpustí.

Jordan Ellenberg

Milí studenti,
rád bych Vás jménem Katedry aplikované matematiky a Jednoty českých matematiků a fyziků přivítal na již 17. ročníku našeho semináře ŠKOMAM (ŠKOla MAtematického Modelování).

Žijeme teď v divné době a jinak, než jsme byli zvyklí. Jiný bude letos i náš seminář. Bude - jak se dnes říká - ONLINE. Přijdeme tím o jednu jeho podstatnou část, a to o možnost osobních setkání a rozhovorů a o možnost představit Vám naši krásnou budovu a inspirující prostředí, kde se dobře studuje a pracuje. O jednu zásadní věc se ale připravit nenecháme, a to o radost z naší krásné, inspirující a užitečné disciplíny - matematiky. Uvidíte řadu přednášek, kde se nám snad podaří Vás překvapit něčím pro Vás novým a zajímavým a ukázat Vám i na dálku, že se svou prací bavíme. V rámci cvičení si i sami vyzkoušíte něco z aplikované matematiky a snad Vás to (aspoň některé) přiměje k úvahám (a k následnému rozhodnutí) přihlásit se ke studiu našeho programu „Výpočetní a aplikovaná matematika“. Byli bychom moc rádi.

Milí studenti, přeju Vám (i nám), ať si oba dny trávené na ŠKOMAMu pořádně užijeme.

V Orlové 2.2.2021 Jirka Bouchala

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. V tomto roce probíhá 17. ročník tohoto semináře. Díky opatřením způsobeným celosvětovou pandemií Covid-19 proběhne tento rok náš seminář poprvé online.

Pro počítačové řešení úloh jsme v prvních 13 ročnících používali komerční systém Matlab. Ve čtrnáctém ročníku jsme se rozhodli udělat změnu a nově jsme zvolili programový balík R. Software R je zejména prostředí určené pro statistickou analýzu dat a jejich grafické zobrazení. Jazyk R ale umožňuje navíc i manipulaci s daty, numerické výpočty a grafické výstupy. Obsahuje také (podobně jako Matlab) řadu dalších knihoven s mnoha připravenými funkcemi. Jeho hlavní výhodou je jeho volná dostupnost pro výukové i vědecké účely.

Tento rok jsme se rozhodli opět pro změnu a v tomto ročníku budeme pro počítačová cvičení používat programovací jazyk Python, který navrhnul Guido van Rossum na konci 80. let. Python je podobně jako jazyk R volně přístupný (https://www.python.org/) a nyní patří mezi nejoblíbenější a nejpoužívanější programovací jazyky. Jeho další velkou výhodou je možnost instalace na většinu běžných platforem jako je MS Windows, Unix, macOS nebo Android. Jedním z důvodů, proč jsme se pro Python rozhodli, je jeho jednoduchost pro učení i pro začátečníky. Podrobný popis tohoto jazyka je k dispozici v [1]. Pro snadnou práci během online cvičení budeme s programy v Pythonu pracovat v prostředí Jupyter Notebook (https://jupyter.org/). Ke každému počítačovému cvičení budeme mít jeden nebo více těchto Jupyter Notebooků, které jsou uloženy na úložišti Binder (https://mybinder.org/). Díky tomu si pro cvičení nemusíte nic instalovat a vše bude pro vás již přichystáno.

Tento seminář je pořádán s finanční podporou Fakulty elektrotechniky a informatiky (http://www.fei.vsb.cz), statutárního města Ostravy a projektu Math Exercises for You 2 (http://math4u.vsb.cz) podpořeného programem Erasmus+. Nad akcí převzala záštitu Jednota českých matematiků a fyziků (http://jcmf.vsb.cz).

VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

OSTRAVA!!!

Spolufinancováno z programu Evropské unie Erasmus+

MATH 4 U

Cvičení 1: Python – nástroj pro matematické modelování

Abychom se mohli věnovat pokročilejším matematickým úlohám, potřebujeme vhodné prostředí, které nám jejich řešení umožní. K tomuto účelu použijeme skriptovací programovací jazyk Python, konkrétně jazyk Python 3. Podrobně se o něm můžete dozvědět např. v textu „Učíme se programovat v jazyce Python 3“ (viz [1]).

V tomto cvičení se seznámíme se základním přehledem datových struktur a příkazů Pythonu 3. Protože práci s proměnnými, cykly a podmínkami, základními funkcemi a knihovnami, najdete podrobně popsánu v Jupyter Notebook „cv1_uvod“, který naleznete na adrese https://mybinder.org/v2/gh/Beremi/SKOMAM/main a také jsme připravili video na YouTube https://youtu.be/2BVFtD5C-r8 provádějící tímto cvičením, uvedeme si v tomto textu jen úkoly k procvičení, které jsou součástí úvodního cvičení.

Úkol 1.1 Napište funkci, která pro zadané rozměry a hustotu kvádru vrátí jeho objem, povrch a hmotnost. ▲

Úkol 1.2 Běžná cihla má rozměry 29 cm $\times$ 14 cm $\times$ 6,5 cm a hustotu $1,9\ \frac{\mathrm{g}}{\mathrm{cm}^3}$. Jaká je její hmotnost? ▲

Úkol 1.3 Napište funkci, která pro zadaný poloměr a výšku válce vrátí jeho objem a povrch. Dále vypočtěte objem a povrch válce o poloměru 10 cm a výšce 5 cm. ▲

Úkol 1.4 Napište funkci, která pro zadané koeficienty a, b, c a hodnoty $d_0 < d_1$ vykreslí graf funkce $f(x) = ax^2 + bx + c$ na intervalu (d_0, d_1). ▲

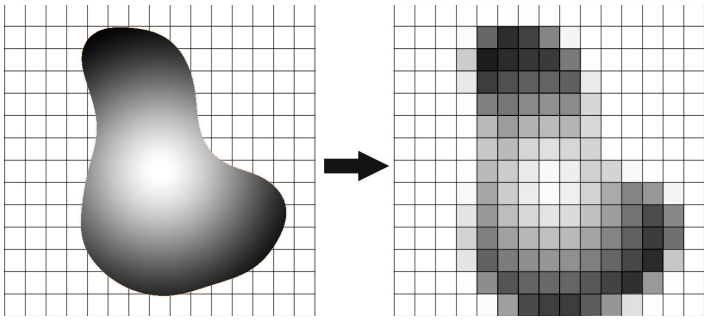
Úkol 1.5 Napište funkci, která pro zadané koeficienty a, b, c vypíše kořeny kvadratické funkce $ax^2 + bx + c$. ▲

Úkol 1.6 Využijte předchozí úkoly a vytvořte funkci, která pro zadané koeficienty a, b, c a hodnoty $d_0 < d_1$ vykreslí graf funkce $f(x) = ax^2 + bx + c$ na intervalu (d_0, d_1) a zároveň znázorní kořeny funkce f. ▲

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,[1] 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`.



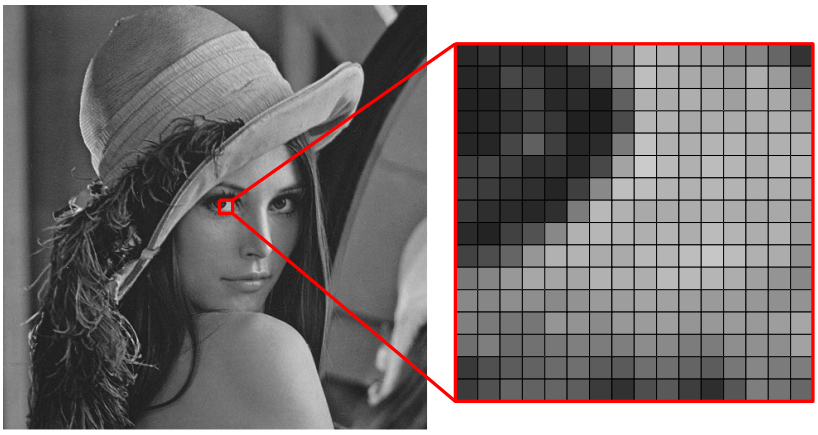
Obrázek 1: Příklad kvantování a vzorkování

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`:

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

Příklad části matice obrazu můžeme vidět na obr. 2.

[1] Tyto hodnoty odpovídají osmibitové reprezentaci čísla.

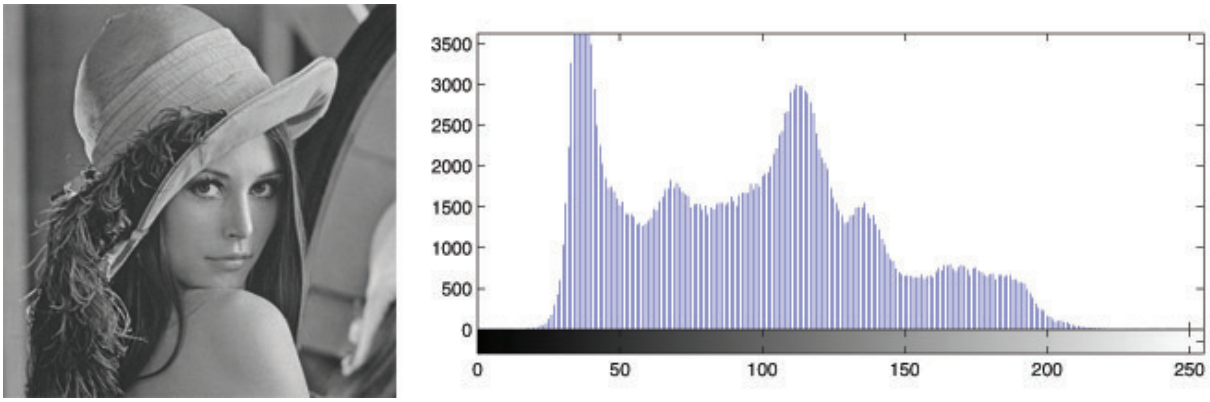


41	51	51	44	49	75	106	139	182	175	160	155	137	134	102	51
38	41	71	64	46	47	78	133	190	174	169	163	156	174	155	97
34	35	50	68	40	43	30	96	177	171	174	169	159	171	168	137
35	39	50	67	55	33	32	75	182	177	168	173	171	172	174	164
38	38	70	96	63	38	43	122	194	176	174	180	179	173	176	177
62	68	63	49	50	40	72	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

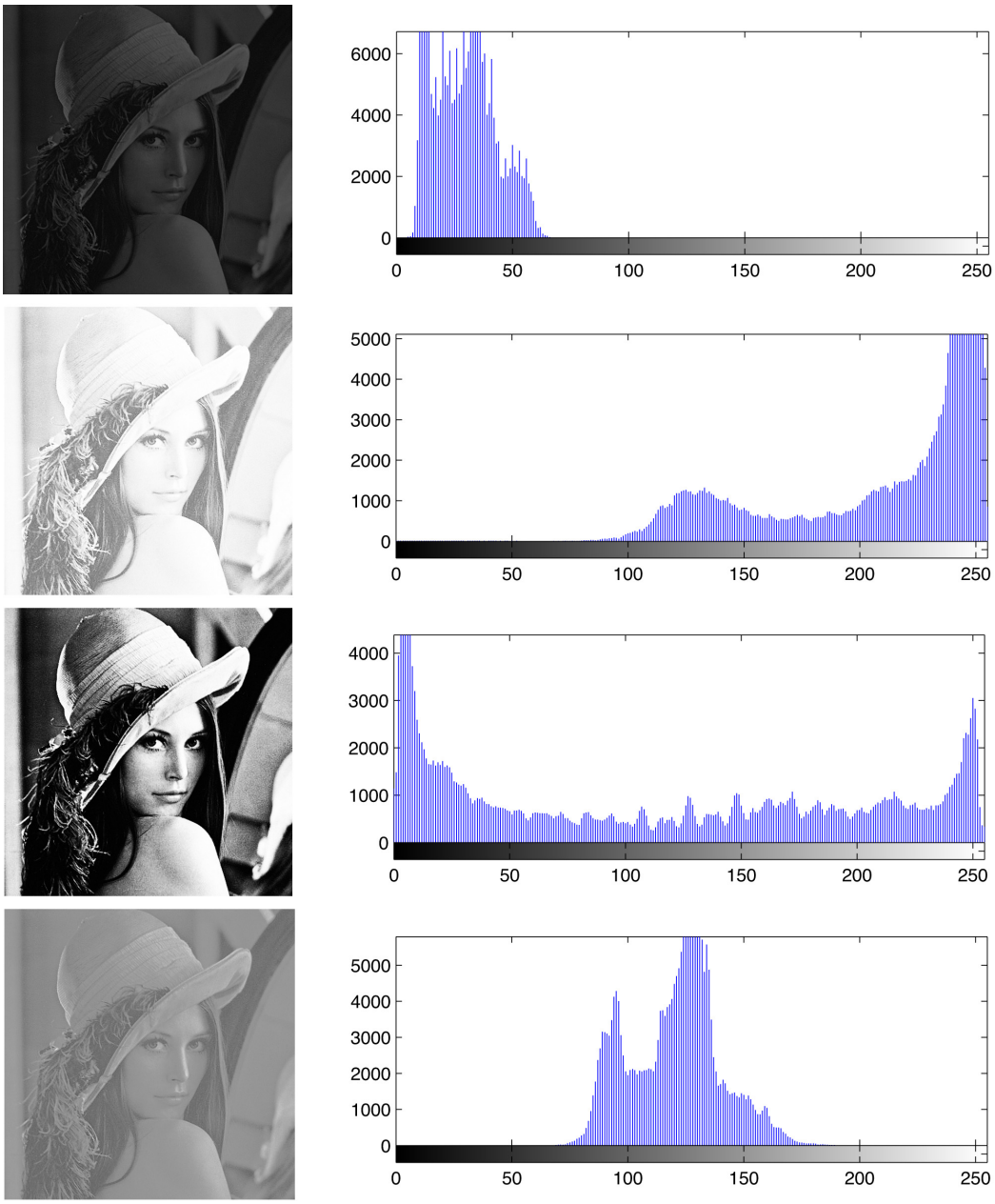
Histogram

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 262 144 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.

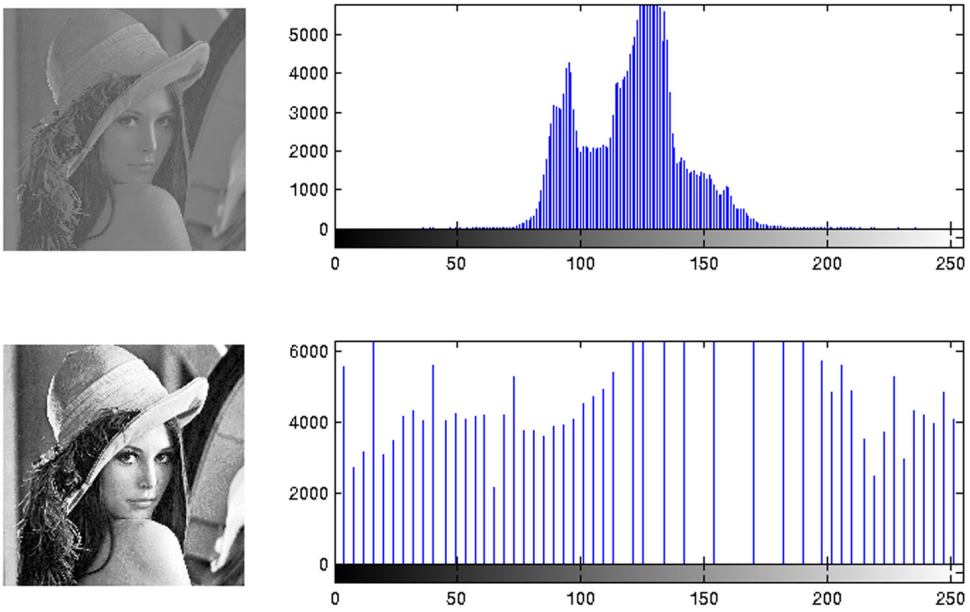
- Ztmavení obrázku můžeme provést obdobně jako zesvětlení, ale musíme zkontrolovat, zda některé hodnoty nevyšly záporné. Pokud ano, nastavíme je na 0. Další možností je vynásobit všechny hodnoty matice obrazu nějakým číslem z intervalu $(0, 1)$. Zde nehrozí, že bychom se dostali mimo rozsah 0 až 255, 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.[2] 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

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

černá je 0 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 Vytvořte funkci, která zadaný obrázek zesvětlí o zadaný počet odstínů. Zesvětlený obrázek vykreslete. ▲

Úkol 2.2 Vytvořte funkci, která zadaný obrázek ztmaví o zadaný počet odstínů. Tmavší

obrázek vykreslete. ▲

Úkol 2.3 Vytvořte funkci, která vytvoří inverzi (negativ) zadaného obrázku. Negativ vykreslete. ▲

Úkol 2.4 Vytvořte funkci, která provede prahování obrázku pomocí zadané hodnoty. Výsledný obrázek vykreslete. ▲

Úkol 2.5 Vytvořte funkci, která k zadanému obrázku vytvoří histogram a použijte ji na obrázek *img_uneq*. Výsledný histogram vykreslete.[3] ▲

Úkol 2.6 Vytvořte funkci, která vyrovná histogram zadaného obrázku a použijte ji na obrázek *img_uneq*. Výsledný obrázek a histogram vykreslete. ▲

Na závěr této části poznamenejme, že mnohem více lze o problematice zpracování obrazu nalézt v textu [2].

Konvoluce

V této části 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:

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.[4]

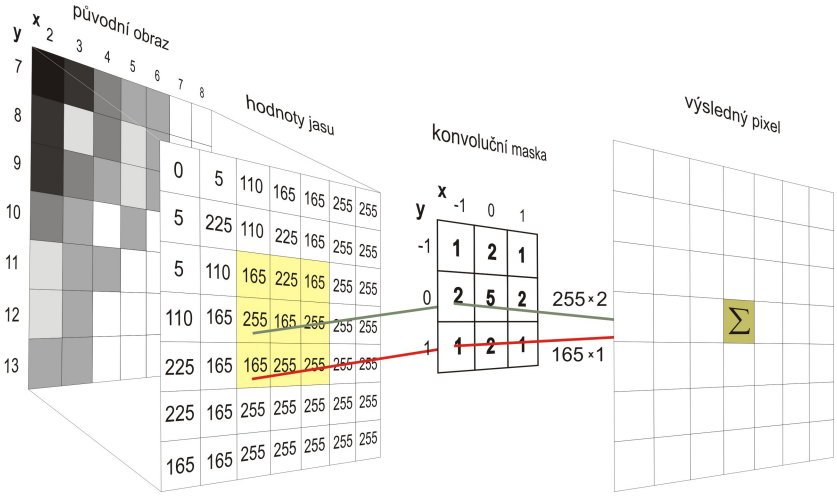
Pomocí konvoluce jsme schopni odstranit z obrázků šum nebo zvýraznit hrany, což je hojně využíváno, chceme-li, aby počítač dokázal sám rozpoznat objekty na obrázku (třeba při zpracování dat z průmyslových kamer).

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

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

[3] **Nápověda:** K vykreslení histogramu se hodí použít sloupcový graf. Ten získáme pomocí příkazu *bar* z knihovny *matplotlib*.

[4] 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.



Obrázek 8: Schéma diskrétní konvoluce

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:

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

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.

Již jsme si vyzkoušeli, jak se v Pythonu 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 2.7 Vytvořte funkci, která provede diskrétní konvoluci zadaného obrázku se zadanou konvoluční maskou. Otestujte efekty různých konvolučních masek.

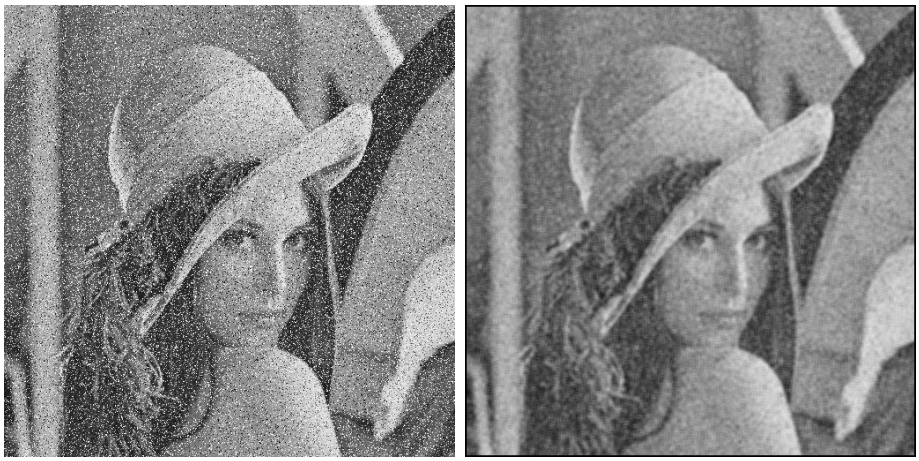
- $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ ... odstraní šum
- $\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ ... Gaussova maska
- $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ ... Laplaceova maska
- $\begin{bmatrix}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



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

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.

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 opět v textu [2].

Vše najdete podrobně popsáno v Jupyter Notebook „cv2_obrazky“ a „cv2_obrazky_reseni“, který naleznete na adrese https://mybinder.org/v2/gh/Beremi/SKOMAM/main a také jsme připravili video na YouTube https://youtu.be/yL-A0N5JDJo provádějící tímto cvičením.

Cvičení 3: Jak se šíří epidemie?

Jednoduchý popis, jak se epidemie šíří v populaci, lze simulovat s využitím tzv. modelu SIR[5]. Tento model byl navržen v roce 1927 skotskými vědci Williamem Ogilvym Kermackem a Andersonem Grayem McKendrickem ve formě následující soustavy lineárních diferenciálních rovnic

$$\left.\begin{aligned} s'(t) &= -\beta s(t) i(t), \\ i'(t) &= \beta s(t) i(t) - \gamma i(t), \quad t > t_0 \\ r'(t) &= \gamma i(t), \\ \text{s počátečními podmínkami } & s(t_0) = s_0,\ i(t_0) = i_0,\ r(t_0) = r_0, \end{aligned}\right\} \tag{1}$$

kde proměnná t označuje čas, konstanta t_0 udává počáteční čas, funkce $s(t)$, $i(t)$, $r(t)$ označují poměr lidí v populaci v daném stavu (tedy $s(t) = \frac{S(t)}{N}$, $i(t) = \frac{I(t)}{N}$, $r(t) = \frac{R(t)}{N}$, kde N je celkový počet osob v populaci a $S(t)$, $I(t)$ a $R(t)$ jsou počty lidí v daném stavu), koeficient β popisuje míru, s jakou se šíří nákaza (anglicky infection rate) a koeficient γ popisuje míru přechodu ze stavu I do stavu R (anglicky recovery rate). Hodnoty β a γ určují v tomto modelu hodnotu základního reprodukčního čísla R_0 v čase t, které je dáno předpisem

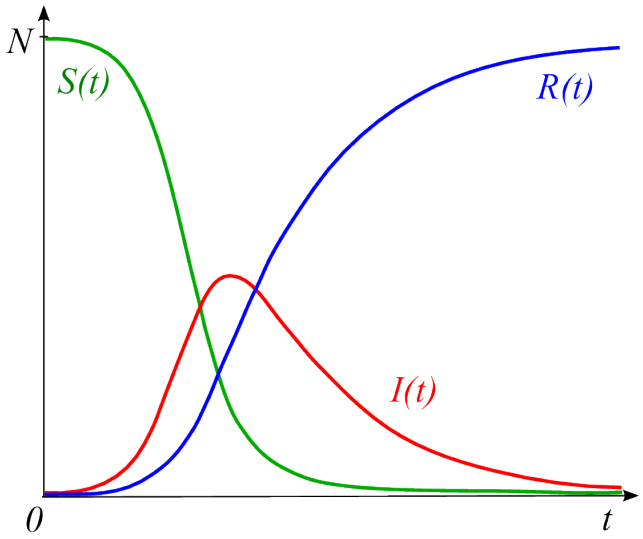
$$R_0(t) = \frac{\beta s(t)}{\gamma}.$$

Reprodukční číslo se dá chápat jako počet lidí, které nakazí jeden nakažený v čase t (v průměru přes všechny nakažené).

Soustava (1) popisuje, za jakých podmínek osoba ve stavu S (tzn. nakazitelná) může přejít do stavu I (tzn. nakažená) a následně ze stavu I do stavu R (tzn. vyléčená či mrtvá). Stav R je pro danou epidemii již konečný a nelze z něj přejít do předchozích stavů. Příklad vývoje počtu lidí procházejících fázemi S, I a R, tzn. hodnoty $S(t)$, $I(t)$ a $R(t)$ získané pomocí řešení soustavy (1) pro konkrétní hodnoty s_0, i_0, r_0, β, γ, můžeme vidět na obr. 12.

Model daný výše uvedenou soustavou diferenciálních rovnic má několik nedostatků: hodnoty koeficientů β a γ obvykle neznáme a v modelu je obsažen předpoklad, že se všechny osoby dané populace chovají stejně a epidemie se šíří stále rovnoměrně. Z těchto důvodů a také proto, že pracovat se soustavou diferenciálních rovnic pouze se středoškolskými znalostmi je přece jen složité, použijeme simulaci tohoto modelu. Na začátku simulace uvažujme, že v populaci o velikosti N máme daný počet osob ve stavu S, I a R (tyto počty označíme jako S, I a R). Současně platí $S+I+R = N$, tzn. že v populaci nejsou jiné osoby než v těchto třech stavech. Součástí modelu bude i simulace pohybu jednotlivých osob. Dále budeme uvažovat, že osoba ve stavu S, která se dostane dostatečně blízko k osobě ve stavu I, přejde s určitou pravděpodobností do stavu I. Osoba ve stavu I vždy po pevně zadaném čase přejde do stavu R. Pokud chceme simulovat opatření jako nošení roušek, nařizování

[5]Zkratka slov Susceptible-Infected-Recovered, které bychom mohli volně přeložit jako Nakazitelný - Nakažený - Vyléčený (popř. mrtvý).



Obrázek 12: Příklad průběhu vývoje S, I, R

karantény na osoby ve stavu I, či omezení pohybu celé populace, můžeme tato opatření v simulaci popsat pomocí jednotlivých parametrů simulace jako je simulace pohybu osob nebo změna pravděpodobnosti, se kterou osoba ve stavu S přejde do stavu I při setkání s nakaženou osobou.

O SIR modelu se můžete podrobněji dočíst v článku [3].

Úkol 3.1 Vygenerujte pro každého jedince v populaci o velikosti např. $N = 1000$ náhodnou pozici v oblasti $\langle 0, 1\rangle \times \langle 0, 1\rangle$ a vykreslete celou populaci. ▲

Úkol 3.2 Uvažujme, že každý jedinec v populaci má jednoznačný index. Převeďte osoby s indexem 0-99 z S do I a osoby s indexem 100-299 z S do R a vykreslete celou populaci (pro každý stav volte jinou barvu). ▲

Úkol 3.3 Vypočtěte vzdálenosti mezi nakaženými (osoby ve stavu I a nakazitelnými (osoby ve stavu S). Pokud má nakazitelný vzdálenost od některého nakaženého menší než hodnota proměnné vzdalenost_nakazy, pak vepište do příslušné pozice v poli mohl_se_nakazit hodnotu True. ▲

Úkol 3.4 Implementujte funkci nove_nakazeni, ve které je pro každou osobu ve stavu S, která je v kontaktu s osobou ve stavu I, aplikována šance na nákazu pravdepodobnost_nakazy. Výstupem bude pole nově nakažených. ▲

Úkol 3.5 Přidejte do funkce update načítání času pro nakažené. Dále v každém kroku kontrolujte, jestli čas nepřekročil hodnotu delka_nakazy, která udává, po jaké době se osoba ve stavu I přesune do stavu R. Pokud čas uvedenou hodnotu překročí, přesuňte osobu z I do R. ▲

Úkol 3.6 Naimplementujte funkci do_prace_a_zpet, která bude splňovat následující:

- v čase od 6:00 do 14:00 se jedinci (znázornění tečkami) přesouvají/jsou v práci,
- ve zbytku času se jedinci přesouvají/jsou doma,
- rychlost přesunu jedinců je 0,1 v jednom časovém kroku,
- funkce vrací vektor posunu (změny pozice) jedinců,
- pokud jsou jedinci do vzdálenosti 0,2 od pracoviště, už se nemusí dále přesouvat,
- pokud jsou jedinci do vzdálenosti 0,05 od domova, už se nemusí dále přesouvat. ▲

Úkol 3.7 Upravte model tak, aby osoby ve stavu I přešly se zadanou pravděpodobností pravdepodobnost_zjisteni_a_karanteny do nově zavedeného stavu Q (osoba je v karanténě). Osoba ve stavu Q je izolována a nemůže nikoho nakazit. ▲

Úkol 3.8 Upravte model tak, aby část osob přestala chodit do práce a zůstala pracovat doma. ▲

Vše najdete podrobně popsáno v Jupyter Notebook „cv3“ a „cv3_reseni“, který naleznete na adrese https://mybinder.org/v2/gh/Beremi/SKOMAM/main a také jsme připravili video na YouTube https://youtu.be/FLCeIwfrmxI provádějící tímto cvičením.

Reference

[1] J. Kubias: Učíme se programovat v jazyce Python 3. [cit. 25.1.2021]. Dostupné z: http://howto.py.cz/index.htm

[2] E. Sojka, J. Gaura, M. Krumnikl: Matematické základy digitálního zpracování obrazu. [cit. 25.1.2021]. Text vytvořený při realizaci projektu „Matematika pro inženýry 21. století“, Vysoká škola báňská - Technická univerzita Ostrava (2012). Dostupné z: http://mi21.vsb.cz/modul/matematicke-zaklady-digitalniho-zpracovani-obrazu

[3] T. Bártlová: Matematika koronaviru: Matematické modely šíření epidemie. [cit. 25.1.2021]. Dostupné z: https://www.matfyz.cz/clanky/matematika-koronaviru-matematicke-modely-sireni-epidemie

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 $A = [a_{i,j}]$.

Definice B.1 Buď $f:\ \mathbb{R}\to\mathbb{R}$ a $x\in\mathbb{R}$. Existuje-li

$$\lim_{h\to 0}\frac{f(x+h)-f(x)}{h},$$

značíme ji $f'(x)$ a nazýváme derivací funkce f v bodě x.

Poznámka B.1 Většinou – a nejinak je to v tomto textu – se pod pojmem derivace rozumí konečná (tzv. vlastní) derivace.

Věta B.1 Buď $f,g:\ \mathbb{R}\to\mathbb{R}$ a $x\in\mathbb{R}$. Pak platí

- $(f\pm g)'(x)=f'(x)\pm g'(x)$, má-li pravá strana rovnosti smysl,
- $(fg)'(x)=f'(x)g(x)+f(x)g'(x)$, existují-li (vlastní) derivace $f'(x)$ a $g'(x)$,
- $\left(\dfrac{f}{g}\right)'(x)=\dfrac{f'(x)g(x)-f(x)g'(x)}{g^2(x)}$, existují-li (vlastní) derivace $f'(x)$ a $g'(x)$ a je-li $g(x)\neq 0$.

Pozorování B.1

- $(c)'=0,\ c\in\mathbb{R}$ (konst.), $x\in\mathbb{R}$,
- $(x^r)'=rx^{r-1},\ r\in\mathbb{R},\ x\in(0,+\infty)$,
- $(\sin x)'=\cos x,\ x\in\mathbb{R}$,
- $(\cos x)'=-\sin x,\ x\in\mathbb{R}$,
- $(\operatorname{tg}x)'=\dfrac{1}{\cos^2 x},\ x\in\mathbb{R}\setminus\left\{\dfrac{\pi}{2}+k\pi:\ k\in\mathbb{Z}\right\}$,
- $(\operatorname{cotg}x)'=-\dfrac{1}{\sin^2 x},\ x\in\mathbb{R}\setminus\{k\pi:\ k\in\mathbb{Z}\}$,
- $(\mathrm{e}^x)'=\mathrm{e}^x,\ x\in\mathbb{R}$,
- $(\ln x)'=\dfrac{1}{x},\ x\in(0,+\infty)$.