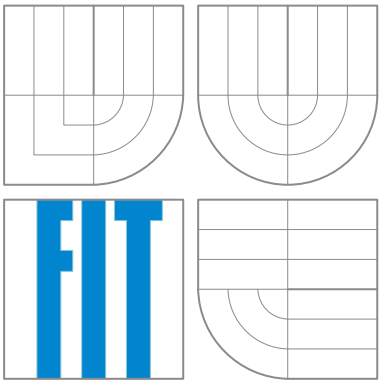


Zpracování obrazu

Detekce hran s využitím dynamického programování

9. května 2014

Autor: Pavel Macenauer, xmacen02@stud.fit.vutbr.cz
Jan Bureš, xbures19@stud.fit.vutbr.cz

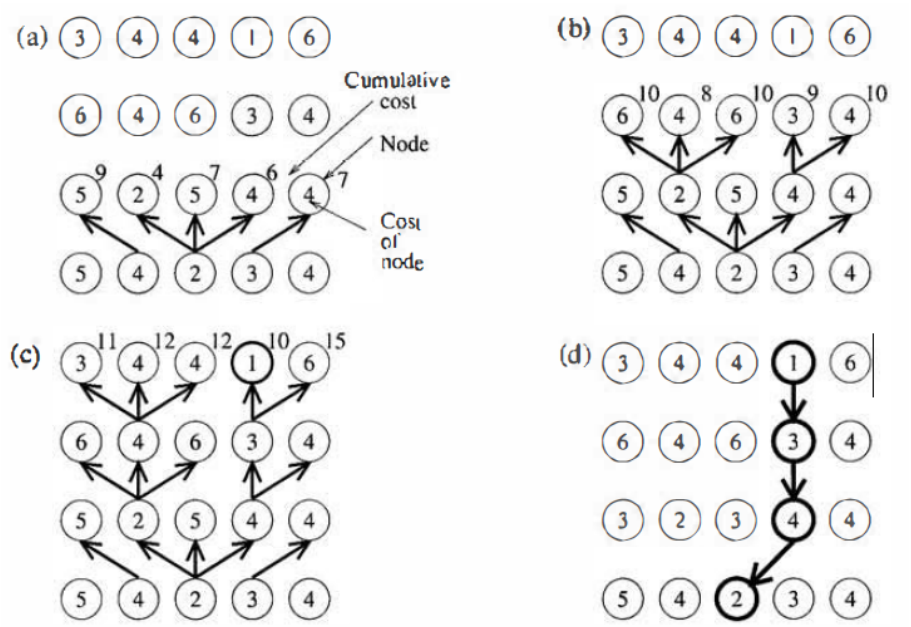
Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

Obsah

Dynamické programování a detekce hran

Dynamické programování **(DP)** stojí na Bellmanovo principu optimality, který zjednodušeně říká, že: zbývající část optimální strategie je rovněž optimální, pokud proces začíná ve stavu, do kterého se dostal v důsledku použití optimální strategie.

Na detekci hran pomocí DP můžeme nahlížet i jako na grafový problém, kdy hledáme cesty grafem, které reprezentují hrany.

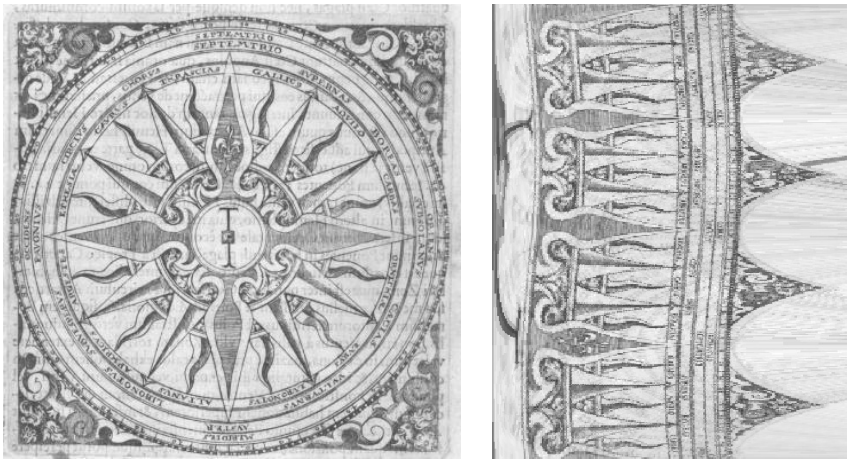


Obrázek 1: Princip DP při detekování hran

Minimální cesta grafem pak reprezentuje potenciální hranu. Při výpočtu ohodnocení se postupuje v DP po úrovních - pro zjednodušení např. po sloupcích, ale může se jednat i o řádky nebo jinou formu.

1. Vypočtou se ceny uzlů v prvním sloupci
2. Prochází se obraz po sloupcích, kdy se vypočte cena uzlu a zjistí minimální předek. Předek se vybírá ze sousedních pixelů v předchozím sloupci. Rozdíl Y souřadnice nemusí být pouze v intervalu $< -1; 1 >$. Cena předka se pak přičte k akumulované ceně uzlu.
3. Ve chvíli kdy dojdeme na poslední úroveň. Sledujeme minimální cestu zpět. Je proto vhodné si pro výpočet uchovávat tabulku indexů pixelů ukazující na své předky.

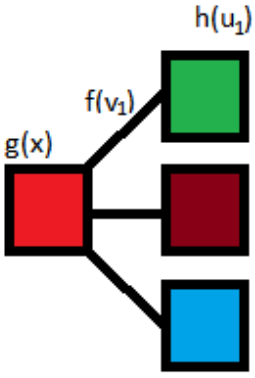
Převod na polární souřadnice Problém u lineárního průchodu obrazem je ten, že kulaté objekty lze detekovat jen těžce. I přes sebelepší cenovou funkci se nejspíše průchod pouze "sveze"po hraně a bude pokračovat stále dál. Při lineárním průchodem polárního prostoru se kulaté objekty začnou tvářit jako lineární a s tím už DP problém nemá. Následně stačí pouze přepočítat detekované souřadnice zpět.



Obrázek 2: Kulatý obraz v polárních souřadnicích

Cenové funkce

Při detekci hran je důležité vytvořit cenovou funkci, kterou je ohodnocen každý pixel a přechod mezi jednotlivými pixely (viz. Obrázek 3). Můžeme tedy mít dvě funkce. Jednu (na obrázku 1 jako funkce h), která přiřazuje hodnotu pixelu podle jeho vlastností a druhou (na obrázku 1 jako funkce f), která urči cenu přechodu mezi dvěma pixely. Výslednou hodnotu (na obrázku 3 jako funkce g) získáme součtem výsledků těchto dvou funkcí. Pokud se do požadovaného pixelu můžeme dostat z více předchozích pixelů je jako předchůdce zvolen ten s nejmenší hodnotou výsledné cenové funkce g.



Obrázek 3: Cenové funkce

V naší aplikace jsou využity tyto 4 způsoby určení ceny přechodů mezi pixely:

Rozdíl RGB složek pixelů Tato cenová funkce je vypočtena na základě rozdílnosti jednotlivých barevných složek dvou pixelů. Cenová hodnota samostatného pixelu je tvořena součtem

cen všech předchozích přechodů, které vedly do daného pixelu. Hodnota počátečního pixelu detekce hrany je 0.

Rozdíl CMYK složek pixelů Každý pixel je nejprve převeden do barevného modelu CMYK. Následné je spočítán rozdíl každé barevné složky. Hodnota přechodu je pak tvořena součtem všech rozdílů barevných složek. Stejně jako u předchozí funkce je cenová hodnota pixelu tvořena součtem cen všech předchozích přechodů, které vedly do daného pixelu. Hodnota počátečního pixelu je taktéž 0.

Převod odstínů šedé Hodnota přechodu je spočítána tak, že se nejprve vypočte hodnota šedé barvy, kterou by měl daný pixel při převodu na odstíny šedé a spočítá se rozdíl těchto hodnot mezi dvěma pixely. Hodnotu pixelů tvoří součet všech předchozích cen hran, které vedly do tohoto pixelu. Hodnota počátečního pixelu je stejně jako u předchozích dvou funkcí nulová.

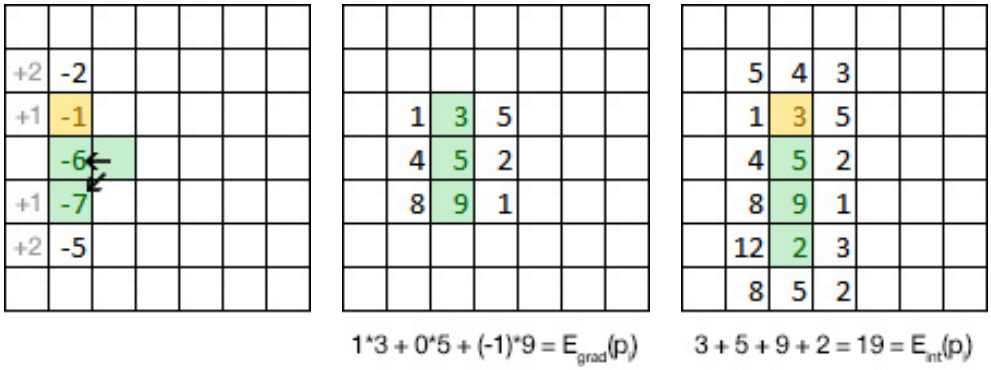
Intenzita a gradient

$$E_{sum} = C_{discont}E_{discont}(p_{i-1}, p_i) - C_{int}E_{int}(p_i) - C_{grad}E_{grad}(p_i)$$

Cena uzlu je vypočítána pomocí předchozího vzorce, kde C určuje váhu dané složky a E energii, resp. ohodnocení. Pomocí vah můžeme experimentovat s jednotlivými složkami. *Discont* pak určuje sousednost/diskontinuitu předchozího pixelu a jeho cenu. Předchozí pixely nemusí být jen ve vzdálenosti 1, ale můžeme zvažovat i pixely ve vzdálenosti 2 nebo vyšší.

Int značí intenzitu pixelu, kdy čím vyšší intenzita, tím světlejší pixel. Je to tak vhodné pokud prohledáváme např. rentgenové snímky na černém pozadí.

Grad značí gradient nebo-li přechod. Maximální hodnoty na nabývá ve chvíli, kdy je rozdíl intenzit sousedních pixelů maximální.



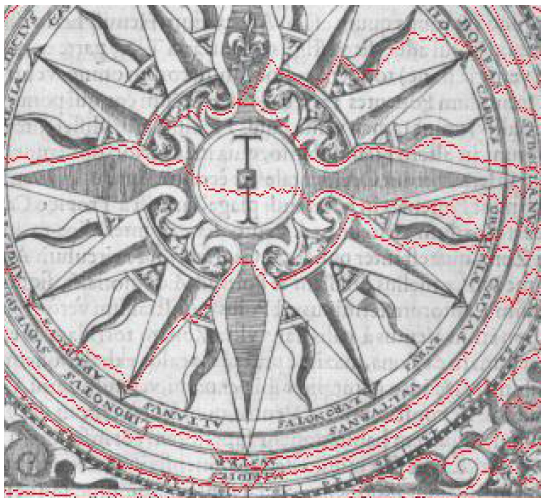
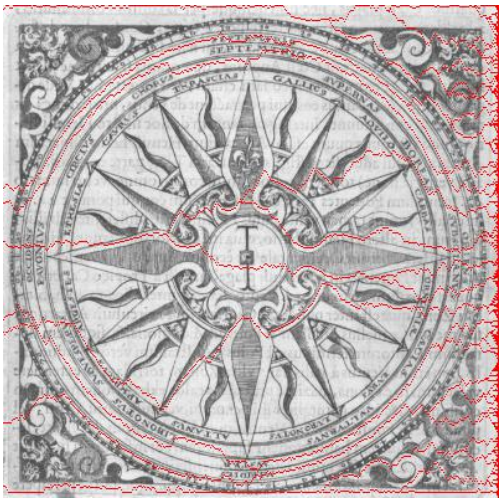
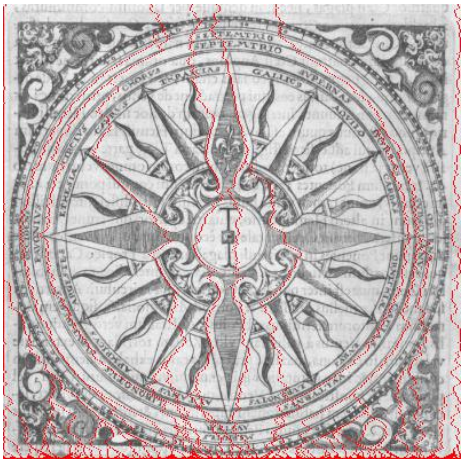
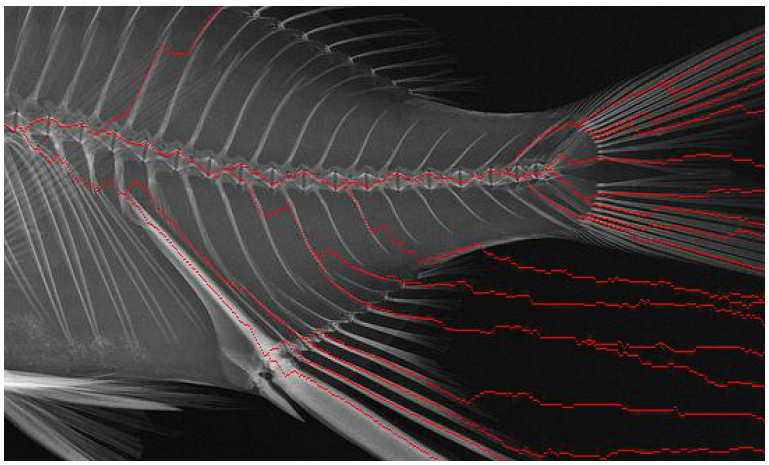
Obrázek 4: Výpočet diskontinuity, gradientu a intenzity

Výsledky

Lineární detekce hran Lineární detekce hran - neboli procházení zleva-doprava, odshora-dolu. Dle výsledků je vidět, že při průchodu obrazem se cesta chytá na příslušné hrany, nicméně problémy a především nápady na jejich řešení jsou následující.

- Detekovat cesty v různých částech obrazu. Detekce začíná vždy z jedné strany. Ve chvíli, kdy je pozadí jednobarevné, tak jde spíše o náhodu, že cesta natrefí na nějakou hranu poté, co projde segmentem obrazu, který je jednobarevný s minimálním rozdílem intenzity. Toho by mohlo jít docílit např. vyhledáním značných rozdílů cen mezi sousedními pixely, neb tak by se velmi pravděpodobně jednalo o část nějaké hrany a z těchto bodů by pak na základě principu optimality šlo začínat.

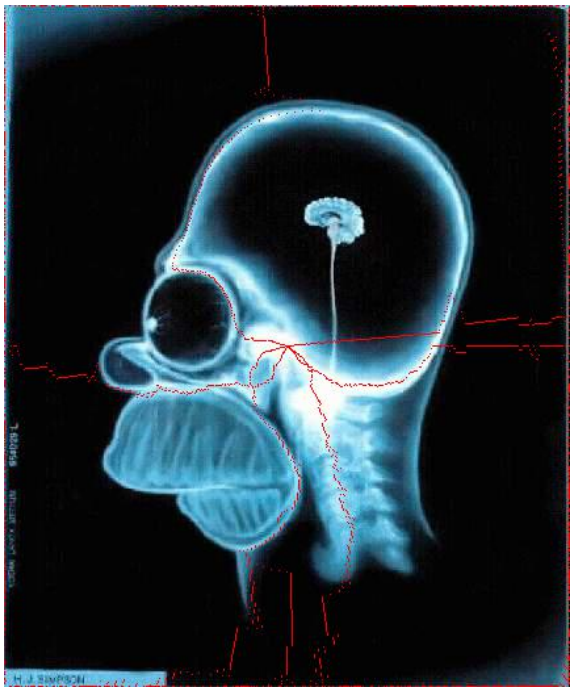
- Více průchodů - složení vertikálního a horizontálního průchodu zaručí více hran.

Polární detekce hran Dalším problémem, který vyvstal je, že pomocí lineárního průchodu není možné detekovat hranu kolem kulatých objektů. Napadlo nás tak, že převodem na polární souřadnice lze kulaté objekty "narovnat", detekovat tak jejich hranu lineárně a přepočítat souřadnice zpět na kartézské.

Problémy tohoto převodu jsou především, polární souřadnice jsou dány vektorem a úhlem, který svírá s počátkem souřadné osy. Při iterování pak v zásadě kroužíme dokola a zvyšujeme poloměr. Rozlišení pro detekování hrany tak není ve všech částech obrazu stejné. Zhoršuje se v rozích polárního prostoru.

Zajímavým poznatkem také je, že pomocí této metody lze obraz segmentovat. Ve chvíli kdy se vhodně určí střed polárního prostoru je detekce hran velmi efektivní a od daného středu rozdělí obraz do více částí.



Obsluha programu

Jedná se o jednoduchou GUI aplikaci spustitelnou pomocí `dynamic-edge-detection.exe` na Windows. V záložce soubor načteme obrázek, který se zobrazí v levé části GUI. V prostřední části pak zvolíme typ cenové funkce (více v sekci Cenové funkce). Směr průchodu grafem. Je možné nastavit průchod zleva-doprava, shora-dolů nebo pomocí polárních souřadnic. Jemnost pak určuje jemnost vykreslení. Následně už jen nastavíme barvu hrany a klikneme na tlačítko Detekuj hrany. Výsledek se zobrazí v pravé části aplikace.

Kompilace Aplikace je multiplatformní. Ve složce `src` je přiložen projekt v Qt Creatoru využívajícího qmake. Stačí tedy použít standardní Qt nástroje ke kompilaci pro Linux a další platformy podporující knihovny Qt.

Referenční obrázky Ve složce `test-images` jsou obrázky používané pro testování

Literatura

[1] Dana H. Ballard, Christopher M. Brown. Computer Vision, ch. 4.5, 4.6.
http://homepages.inf.ed.ac.uk/rbf/BOOKS/BANDB/LIB/.

[2] Ghassan Hamarneh Rafeef Abu-Gharbieh, Chalmers University of Technology. Implementation and Comparison of Four Different Boundary Detection Algorithms for Quantitative Ultrasonic Measurements of the Human Carotid Artery.
http://www.cs.sfu.ca/~hamarneh/ecopy/msee1996.pdf.

[3] M. Sonka, V. Hlavac, R. Boyle. Image Processing, Analysis, and Machine Vision, ch. 6.2.5.

[4] Sheila Timp, Nico Karssemeijer, Department of Radiology, University Medical Center Nijmegen. A new 2D segmentation method based on dynamic programming applied to computer aided detectection in mammography.
http://stimp.home.xs4all.nl/sheila/Timp04_segmentatie.pdf.

[5] Vincent Tan. Converting between raster, Cartesian and polar coordinates.
http://polymathprogrammer.com/2008/10/02/converting-between-raster-cartesian-and-pola