Semestrální projekt xx – závěrečná zpráva

Převod barevného obrazu na šedotónový

Ján Brída, xbrida01@stud.fit.vutbr.cz

Vedoucí práce: *doc. Ing. Martin Čadík, Ph.D.,* cadik@fit.vutbr.cz

Květen 2017

1 Úvod

Tento semestrální projekt se zaobírá problematikou transformace barevného obrazu na šedotónový. Černo-bílé fotografie se stále běžně objevují v mnohých sférách vizuální prezentace, avšak problémem šedotónové konverze bývá zachování detailů, především v izoluminantních oblastech. To znehodnocuje celkový dojem a sňižuje schopnost pochopit obsah obrazu.

Bylo navrženo množství metod, kterými jsou barevné obrazy převáděny do šedotónové podoby. Typicky jsou rozděleny na globální a lokální, podle toho, zda-li jsou zkoumány závislosti v obraze. Prvně jmenované bývají často dostatečně rychlé, neposkytují však vždy vizuálne věrohodnou reprezentaci vzhledem ke vstupu. Lokální techniky se typicky snaží brát v úvahu barevné změny v oblastech a uspůsobit podle nich výpočet luminance. Některé takovéto algoritmy pracují v gradientní doméně.

Takovýmto algoritmem je i transformace podle experimentů s barevným systémem zvaným Coloroid [3, 1]. Podobá se metodě [2], která iterativně minimalizuje chybu objektivní funkce, podle lokálních kontrastů mezi pixely, na základě transformace do CIE L*a*b*. Ta se snaží zachovat detaily použitím rozdílů luminance nebo chrominance (maximum) mezi pixely, dochází ale naneštěstí k nekonzistencím. Navíc není algoritmus automaticky adaptabilní, protože vyžaduje nastavení trojice parametrů ovlivňující výslednou transformaci.

2 Šedotónová transformace

Jedná se o redukci alespoň 3 barevných složek (např. RGB) na 1D data. Jak bylo zmíněno, výzvou takovéhoto převodu je zachovat ekvivalentní přechody v novém obraze, aby byl výsledek percepčně blízký vstupu.

2.1 Coloroid

Barevný systém Coloroid byl vytvořen pro popsání spojitého barevného prostoru, který je esteticky uniformní protože dokáže definovat harmonické vztahy mezi barvami, jelikož sousedící barvy jsou určeny stejně velkými celočíselnými intervaly. Spojitost barevného prostoru Coloroidu poskytuje možnost jednoznačne určit vztahy mezi barvami v jiných barevných systémech. Coloroid vynalezli v 60.-80. letech minulého staletí v Budapešti na základě experimentů při sledování barev pozorovatelem, celkově bylo těchto jednotlivců až 70 000.

Barvy jsou v tomto 3D prostoru definovány pomocí HSL (*hue*, *saturation*, *luminance*) komponent (obrázek 1). Coloroid reprezentuje válec, kde angulární souřadnice (A) určuje odstín, nasycení definuje radiální souřadnice (T) a na vertikální ose (V) se pohybuje svítivost barvy. Důležitou informací je, že s nulovou saturací je barva reprezentována v šedotónové oblasti. 48

základních barev v základni kužele představuje vztah k CIE 1931 barevnému gamutua nejvíce nasycené mají hodnoty na obvodu tohoto kruhu.

V práci [3] je Coloroid využit pro převod na percepčně blízký šedotónový výsledek vstupního obrazu. Autoři provedli sledování několika barev, kde náhle jedna z nich byla nahrazena za nesaturovaný vzorek. Pozorovatel po změně zhodnotil uniformitu vnímání v krátkém čase (aby oko nemělo čas se přispůsobit). Na základě těchto experimentů došlo na substituci šedotónových vzorků do vygenerovaných posloupností s fixními A, V komponentami a postupně rostoucí saturací. Výsledkem byl poznatek, že nedochází k monotónní diferenci mezi těmito vzorky v šedotónové oblasti.

Byla vytvořena matice 7×7 šedotónových změn mezi 7 vybranými odstíny (zástupci 7 skupin základních barev Coloroidu). Na diagonále se vyskytují nuly a je anti-symetrická, viz jak byly hodnoty naměřeny [3, 1]. Tato matice je základním stavebním prvkem pro převodovou rovnici na šedotónovou podobu. Zbylé barvy získají své diference bilineární interpolací.

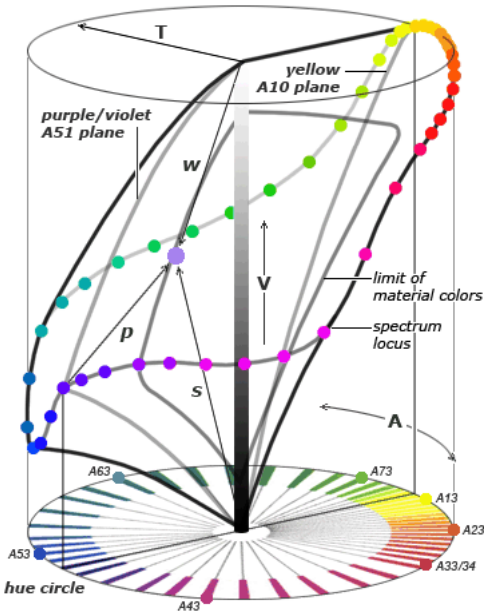

Figure 1: Model barevného prostoru Coloroid.

2.2 Rovnice pro šedotónovou transformaci

Hledanou veličinou je gradient, popisující rozdíl mezi dvěmi Coloroid barvami (A_1, T_1, V_1), (A_2, T_2, V_2) a je vypočten podle rovnice (1). Ta obsahuje termy diferencí luminance (rovnice (4)), saturace (rovnice (3)) a odstínu (rovnice (2)), postupně odvozeny na základě tabulky změn pro 7 barevných skupin.

$$\Delta_{1,2} = dL(L_1, L_2) + S(A_1, T_1, V_1, A_2, T_2, V_2) + h(A_1, T_1, A_2, T_2) \quad (1)$$

Gradient odstínu dvou barev A_1, A_2 je vypočten jako

$$h(A_1, T_1, A_2, T_2) = w_h \cdot H(A_1, A_2) \cdot \sqrt{u(T_{1rel}) \cdot u(T_{2rel})}, \quad (2)$$

kde T_{rel} je relativní saturace pro každou rovinu odstínů v každé úrovni luminance. Term $u(x)$, kde $x = 2 \cdot T_{rel}$, je definován jako $u(x) = 0.5 \cdot x$, pokud $x < 0.5$ a $u(x) = \sqrt{x} - 0.5$ jinak.

Určení šedotónové změny pro korespondující saturaci závisí na

$$S(A_1, T_1, V_1, A_2, T_2, V_2) = w_s \cdot [S(A_2, T_2, V_2) - S(A_1, T_1, V_1)] \quad (3)$$

a pro luminanci

$$dL(L_1, L_2) = L_2 - L_1, \quad (4)$$

jak je podrobněji rozebráno odvození těchto rovnic v [1].

2.3 Výpočet na základě gradientního pole

Obdržený nekonzistentní gradientní obraz je následně stabilizován pomocí speciální metody, užívající ortogonální projekci pro nalezení nejbližšího konzistentního gradientního pole. Výsledek tohoto procesu je nakonec zpracován integrací podle navštíveného gradientu [3].

Hlavní myšlenkou korekce gradientního pole $\mathbf{g}$ je určit chybu $E_{(i,j)}$ jednotlivých elementárních smyček všech pixelů. Konzistentní gradientní pole po této úpravě musí splňovat rovnost

$$\mathbf{g}_{(i+1,j),x} + \mathbf{g}_{(i,j),y} = \mathbf{g}_{(i,j),x} + \mathbf{g}_{(i,j+i),y}.$$

Algoritmus opakovaně prochází daný obraz a pro každý pixel spočte $E_{(i,j)}$ podle výše uvedeného vztahu. Chyba je pak distribuována, čímž je obdržen nový gradient $\mathbf{g}(i, j)$. Definice numerické metody představuje

$$\mathbf{g}^{(k+1)} = \mathbf{g}^{(k)} - \frac{1}{4} \cdot E_{(i,j)} \cdot \mathbf{N}_{(i,j)}, \quad (5)$$

kde $\mathbf{N}_{(i,j)} = (0, \ldots, 0, +1, +1, -1, -1, 0, \ldots, 0)$ identifikuje $2 \cdot N \cdot M$ vektor s odpovídajícími nenulovými koeficienty pro $\mathbf{g}_{(i,j)}$, kde N je výška a M šířka obrazu. $E_{(i,j)}$ postupně konverguje k nule. V algoritmu 1 představuje $1 < \omega < 2$ urychlení konvergence.

Alg. 1 Sekvenční algoritmus korekce gradientního pole $\mathbf{g}$.

```
1:  function CORRECT GRADIENT(g, ω, ϵ)
2:      repeat
3:          E_max ← 0
4:          for i ∈ {0, ..., N} do
5:              for j ∈ {0, ..., M} do
6:                  E ← g_(i+1,j),x + g_(i,j),y − g_(i,j),x − g_(i,j+i),y
7:                  if |E| > E_max then
8:                      E_max ← |E|
9:                  E ← 1/4 · E · ω
10:                 g_(i,j),x ← g_(i,j),x − E
11:                 g_(i,j+1),y ← g_(i,j+1),y − E
12:                 g_(i+1,j),x ← g_(i+1,j),x + E
13:                 g_(i,j),y ← g_(i,j),y + E
14:     until E_max > ϵ
```

2.3.1 Optimalizace

Výše uvedený algoritmus je cyklickým řešením korekce gradientu jeho elementárních smyček obrazu. Urychlení výpočtu pomocí data paralelních algoritmů na grafické kartě rozděluje algoritmus na 3 zájmová místa (následující výčet referuje řádky algoritmu 1):

1. #6: paralelní výpočet chyby $\mathbf{E}_{(i,j)}$ všech elementárních smyček, kde $\mathbf{E}$ je vektor o velikosti $N \cdot M$,
2. #7-8: redukce $\mathbf{E}$ s asociativním operátorem max pro získání $E_{\max}$,
3. #10-13: aby nedocházelo k čtecím/zápisovým datovým konfliktů, jsou jednotlivé přiřazení výsledků provedeny paralelně samostatně.

Je možné vytvořit i efektivnejší modifikaci této korekce výběrem upravované elementární smyčky na základě maximální chyby $E_{\max}$ v jedné iteraci algoritmu [1]. Určení elementární smyčky s $E_{\max}$ v paralelním prostředí může být zjednodušeno na nalezení odpovídajícího indexu pixelu během redukce $\mathbf{E}$. Index je jednoduše definován pracovním identifikátorem výpočtu a dodatečne vyžaduje $\lceil N \cdot M \rceil_{(2 \cdot W)} \div (2 \cdot W)$ paměťového místa pro uchování této informace při rekurzivní redukci, kde W určuje velikost pracovní skupiny.

Další optimalizační návrh představuje *šachovnicový* průchod gradientním polem. Tento způsob korekce gradientu je vyobrazen na obrázku 2. Metoda je rozdělena do 3 volání kernelu pro výpočet chyby elementárních smyček. V jednotlivých voláních jsou aktualizovány pouze jisté smyčky, aby nedocházelo ke zapisovacím konfliktům. To znamená, pro $[i, j]$, kdy:

- $i \bmod 2 = 0 \wedge j \bmod 2 = 0$,

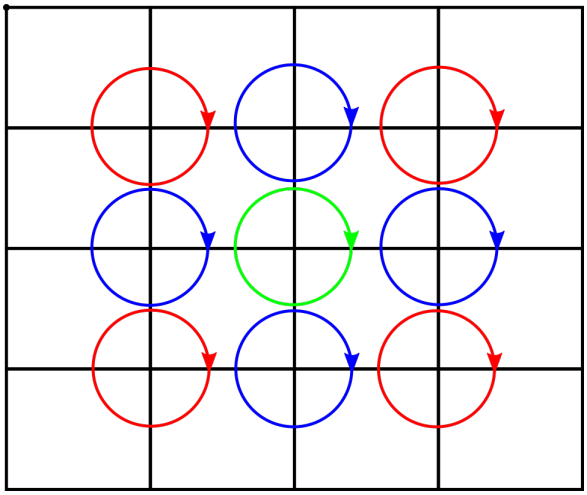

Figure 2: Šachovnicovitá korekce gradientu. Pouze smyčky jedné barvy jsou prováděny paralelně.

- $(i \bmod 2 \neq 0 \wedge j \bmod 2 = 0) \vee (i \bmod 2 = 0 \wedge j \bmod 2 \neq 0)$,
- $i \bmod 2 \neq 0 \wedge j \bmod 2 \neq 0$.

Tento typ modifikování gradientního pole má dobré předpoklady paralelně distribuovat chybu a tím jakkoby rozmazávat lokální nekonzistence.

Poslední z navržených řešení staví na hierarchickém rozdělení gradientního pole obrazu do stromové struktury quadtree. Úmyslem je snadněji identifikovat oblasti, ve kterých je chyba elementárních smyček je nad prahem ϵ a pouze tyto smyčky opravovat. Pro jednoduché paralelní vybudování quadtree se gradientní pole uloží podle z-křivky do 1D pole. Uzly ve vyšší úrovni quadtree sumují hodnoty svých potomků a podle uspořádání v z-křivce platí:

$$Q^n_{(i)} = Q^{n-1}_{(4\cdot i)} + Q^{n-1}_{(4\cdot i+1)} + Q^{n-1}_{(4\cdot i+2)} + Q^{n-1}_{(4\cdot i+3)} \tag{6}$$

Takto obydlený quadtree už lze použít pro hierarchickou korekci, třeba si však uvědomit, že v každé iteraci budou muset být znovu vypočteny vyšší úrovně. Následující body popisují průběh algoritmu:

1. přepočti vyšší úrovně quadtree,
2. inicializuj seznam $\textsc{ParentIndex}_{n-2}$ na indexy všech uzlů $n-2$ úrovně (je jich spolu 16), nastav počet zpracovávaných uzlů na $N^a_i = 64$
3. postupuj sestupně v quadtree od úrovně $n-3$, pro každou úroveň proveď:

(a) vypočti chybu E v úrovni i pro potomky uzlů indexovaných podle $\textsc{ParentIndex}_{i+1}$, zároveň ulož Mortonový kódy těchto potomků do seznamu $\textsc{ActNodes}_i$ a inicializuj nový seznam $\textsc{Flag}_i$ tak, aby se pro smyčky, kterých hodnota $|E| > \epsilon$, byla uložena na odpovídajícím indexu 1, jinak 0,

(b) proveď exkluzivní sken nad $\textsc{Flag}_i$, výsledek ulož do seznamu $\textsc{Offsets}_i$,

(c) $N^a_{i-1} = \textsc{Flag}_{i,(N^a_i-1)} + \textsc{Offsets}_{i,(N^a_i-1)}$,

(d) pokud $N^a_i = 0$ skonči průchod stromem,

(e) uskutečni kompakci $\textsc{ActNodes}_i$ podle $\textsc{Flag}_i$ na pozici definované $\textsc{Offsets}_i$, výsledkem bude $\textsc{ParentIndex}_i$ o velikosti N^a_{i-1},

(f) $N^a_{i-1} = 4 \cdot N^a_{i-1}$.

4. pokud $N^a_0 = 0$ skonči iterativní algoritmus,

5. proveď šachovnicovou korekci gradientu podle $\textsc{ParentIndex}_1$, chyba se zapisuje do seznamu $\textsc{Error}_0$,

6. redukuj $\textsc{Error}_0$ pro nalezení absolutního maxima $E_{\max}$,

7. pokud $E_{\max} > \epsilon$ vrať se na krok #1, jinak skonči.

3 Implementace

Implementace v jazyce C++ používá zdrojový kód od Laszla Neumanna pro výpočet gradientu luminance podle Coloroidu. Optimalizace korekce gradientu proběhla za pomoci GPGPU OpenCL API. Toto rozhraní je zapouzdřeno do vlastních tříd v jmenném prostoru `cl` (složka `compute`). Program samostatně implementuje redukci i exkluzivní sken. Program byl napsán jako plugin do aplikace `TMOcmd/TMOgui`, proto hlavní třídu přestavuje `TMOCadik08`, která se stará nejen o vstupní parametry, ale ovládá i celkový průběh konverze.

Nejdpostatnější část korekce gradientu sa nachází v metodě `TMOCadik08::correctGrad`. Ta volá kernely definované v `resources/kernels/color2gray.cl` a řídí proces iterativní korekce gradientu. Implementováno bylo několik verzí, jak bylo naznačeno v předcházející kapitole.

Cyklická korekce gradientu: naivní implementace algoritmu prakticky opisuje paralelní variantu algoritmu 1. Nejprve jsou do vektoru **E** vypočteny chyby všech elementárních smyček. Poté sa provede korekce a také redukce **E** pro nalezení maximální chyby $E_{\max}$.

Korekce maximální chyby: tato varianta poskytuje výpočet **E** a nalezení $E_{\max}$ podobně jako v předcházející verzi, korekce se však provádí

jen pro smyčku s $E_{\max}$. Nalezení indexu elementární smyčky s odpovídající maximální chybou lze provést během redukce. Dodatečně se použijí seznamy odpovídajících indexů parciálních zpracování vstupního pole.

Šachovnicová korekce gradientu: tato verze vychází z principu vyobrazeném v obrázku 2. Podobně jako dříve sa vypočte **E** a pomocí redukce absolutního maxima je nalezeno $E_{\max}$. Korekce probíhá ve 3 cyklech, kdy proměnná `mode` identifikuje aktuálně zpracovávané elementární smyčky.

Hierarchická korekce gradientu pomocí z-uspořádaného quadtree: soubory `morton.h` a `morton.cpp` definují kódování/dekódování Mortonových kódů. Pro urychlení těchto procesů jsou předpočítány byty Mortonových kódů do shiftovacích look-up tabulek. Třída `quadtree` (`quadtree.h` a `quadtree.cpp`) zřejmě poskytuje zapouzdření quadtree pro obrazový gradient. Alokace probíhá podobně jako pro 4-ární strom, ovšem v každé úrovni jsou uzly uspořádány podle z-křivky. Korekce gradientu probíhá zase iterativně při selekci maximální chyby $E_{\max}$ z **E**, ovšem **E** je redukované gradientní pole na základě průchodu stromem. Přepočtení vyšších úrovní v quadtree se děje v kernelu `avg_grad` podle vzorce (6). Kernel `calc_error` provádí selekci potomků na zpracování v další úrovni nastavením bufferu $\textsc{Flag}_i$. $\textsc{ActNodes}_i$ jsou vypočítány jako $(\textsc{ParentIndex}_{i+1,(j)}/4] \ll 2) \mid (j \bmod 4)$, kde j je číslo pracovního objektu. Exkluzivním skenem jsou zjištěny ofsety a kompakcí se určí následující Mortonovy kódy aktivních uzlu ($\textsc{ParentIndex}_i$). Nakonec se korekce gradientu provede podobně jako v předchozí variantě.

Další: za zmínku stojí pokus o korekci gradientu s procházením delší cesty. Vyskoušena byla *spirálovitá* varianta, kdy smyčky postupně snižují svoji velikost jak se blíží ke středu obrazu. Problémem takovéhoto průchodu bylo nerovnoměrné zatížení vláken. Uvažována byla i varianta Hilbertovy křivky, tzv. Moorova křivka, které počátek a konec představují sousední uzly (je tedy prakticky uzavřená), ovšem nebyly k dispozici gradienty do všech směrů (jejich výpočet podle dostupných kódu způsobil *segmentation fault*).

Také byla implementována paralelní verze integrace výsledného gradientního pole. Integrace probíhá nejprve sekvenčně pro první sloupec obrazu ve směru osy Y, následně se pro každý řádek provede výpočet na základě komponenty x. Nad výsledným obrazem se nakonec provede kalibrace.

4 Výsledky

Testování proběhlo nad sérií obrázků z `http://cadik.posvete.cz/color_to_gray_evaluation`. Mnoho z nich poskytuje dobré případy izoluminantních oblastí, které by se v klasické konverzy podle CIE Y vytratily. Pro studium estetických rozdílů byl zvolen obrázek 3. Z důvodu nefunkčních ovladačů OpenCL na GNU/Linux a problému při překladu `TMS` na Windows

bylo naneštěstí prozatím volání kernelů vykonáno na CPU.

Figure 3: Testovaný obrázek.

Výsledky:
Cyklická korekce gradientu: $\epsilon = 0.5$, 0.273868 [s].
Korekce maximální chyby: $\epsilon = 0.5$, více jak 5 [s].
Šachovnicová korekce gradientu: $\epsilon = 1.5$, 0.0630407 [s].
Hierarchická korekce gradientu: $\epsilon = 0.1$, 0.464364 [s].

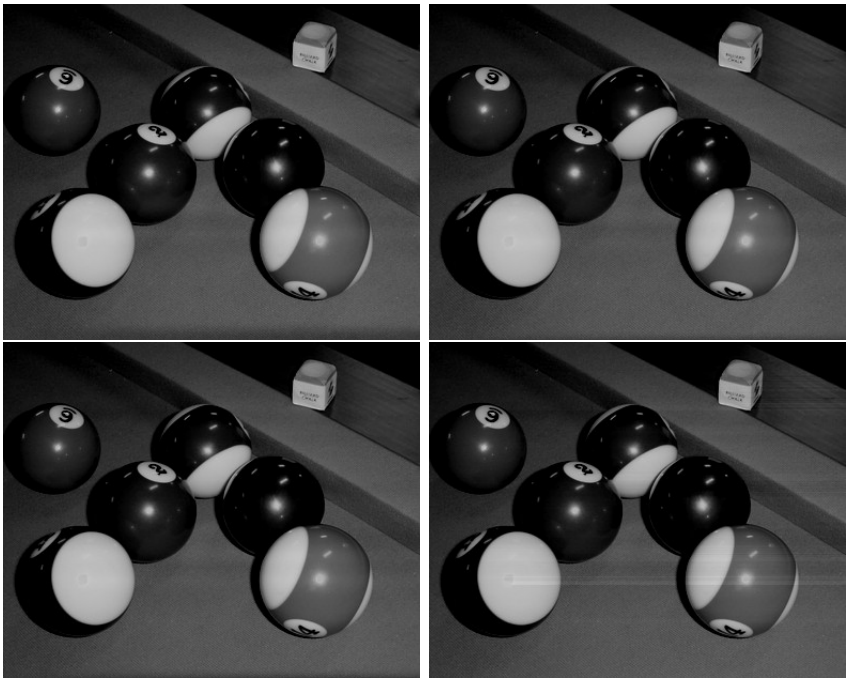

Figure 4: Výsledek konverze: cyklické, max., šachovnicové, hierarchické.

4.1 Uživatelská studie

Z výsledků výše je patrné, které varianty jsou nejpoužitelnejší. Nejlépe si vedla korekce podle šachovnicového vzoru. Ta dokáže i při poměrně vysokém ϵ distribuovat chybu tak, aby byl výsledek stále přijatelný a tím také omezí délku trvání algoritmu. Naivní cyklická korekce bohužel neposkytuje dostatečné zrychlení a navíc způsobuje dodatečné artefakty. Korekce maximální chyby nepřineslo slibované zrychlení, i když metoda dokáže poměrně účinně odstranit nekonzistence. U hierarchické korekce se momentálně zdá být problematické používat ve vyšších úrovních zesumovanou hodnotu gradientu potomků. Pravděpodobně bude nutné upravit porovnávání s ϵ anebo změnit typ informace, která bude uložena. Potíž při opravovaní je, že ϵ musí být poměrně nízké, aby technika produkovala skutečně dobrý výsledek. Zároveň se však zdá být asi nejlepší z nabízených metod.

5 Závěr

Byly implementovány, otestovány a porovnány metody korekce inkonzistentího gradientního pole pro převod barevných obrázků do šedotónové podoby podle Coloroidu. Z vymyšlených technik nejvíce zaujala šachovnicovitá varianta, poskytující při vyšším ϵ uspokojivé výsledky. Komplexnější hierarchická korekce pomocí quadtree vyžaduje pár vylepšení, ale je taktéž použitelná.

References

[1] Čadík, M. *Perceptually Based Image Quality Assessment and Image Transformations*. Ph.d. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, January 2008.

[2] Gooch, A. A., Olsen, S. C., Tumblin, J., and Gooch, B. Color2gray: salience-preserving color removal. In *ACM Transactions on Graphics (TOG)* (2005), vol. 24, ACM, pp. 634–639.

[3] Neumann, L., Čadík, M., and Nemcsics, A. An efficient perception-based adaptive color to gray transformation. In *Proceedings of Computational Aesthetics 2007* (Banff, Canada, 2007), Eurographics Association, pp. 73– 80.