

Interaktivní obarvování kreseb

Dominik Fiala

Čtvrtek 11:00

OI, Počítačové vidění a digitální obraz

fialado1@fel.cvut.cz

I. ZADÁNÍ

A. Popis problému

Cílem této práce je vytvořit program, s jehož pomocí bude uživatel schopný obarvit šedotónovou kresbu, a to v reálném čase a s co nejmenším vynaložením práce. Idea je taková, že uživatel do původního obrazu zakreslí kurzorem čáry, jejichž barva bude odpovídat požadované barvě, kterou má být daná oblast obarvena (viz Obrázek 1).

Obarvení můžeme převést na obecnější problém označování vrcholů grafu \mathcal{G} (dále jen jako *značkování*), kdy množina značek \mathcal{L} bude reprezentovat barvy zakreslené uživatelem. V našem případě bude vstupní obrázek I představovat graf $\mathcal{G} = \{\mathcal{V}, \mathcal{E}_p\}$, kde \mathcal{V} je množina vrcholů obsahující pixely P z obrázku I a \mathcal{E}_p je množina hran, tvořená každou dvojicí přilehlých pixelů z P . Tento graf \mathcal{G} má důležitou vlastnost a sice, jedná se o rovinný graf. Úkol lze tedy přeformulovat jako nalezení značkovací funkce c , která každému pixelu $p \in P$ přiřadí značku $c_p \in \mathcal{L}$, přičemž obecně platí, že c je po částech spojitá a je konzistentní se zkoumanými daty[1].

Jak je uvedeno v článkách [1][2][3], lze náš problém formulovat jako minimalizaci energie E , přičemž v tomto případě budeme hledat takovou funkci c , která minimalizuje energii

$$E(f) = \sum_{\{p,q\} \in \mathcal{E}} V_{p,q}(c_p, c_q) + \sum_{p \in P} D_p(c_p), \quad (1)$$

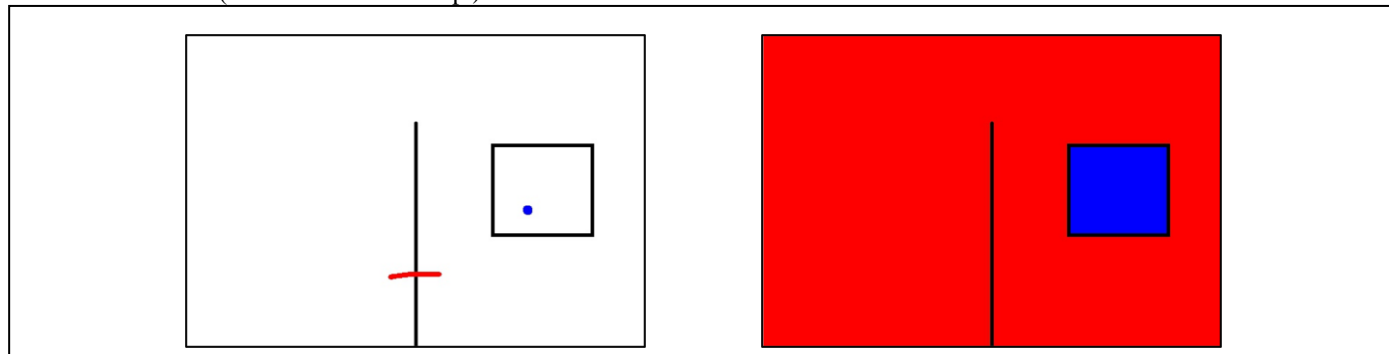
kde proměnná $V_{p,q}$ představuje energii nespojitosti značení mezi dvěma sousedními pixely p a q . Zároveň pro ni platí tyto vlastnosti:

$$\begin{aligned} V(\alpha, \beta) &= V(\beta, \alpha) \geq 0, \\ V(\alpha, \beta) &= 0 \Leftrightarrow \alpha = \beta, \\ V(\alpha, \beta) &\leq V(\alpha, \gamma) + V(\gamma, \beta). \end{aligned} \quad (2)$$

Proměnná D_p představuje energii přiřazené značky f_p pixelu p .

Pokud v našem původním grafu \mathcal{G} rozšíříme množinu vrcholů o pomocné vrcholy C , tedy $\mathcal{V} = \{P, C\}$, a množinu hran o pomocné hrany \mathcal{E}_c , tedy $\mathcal{E} = \{\mathcal{E}_p, \mathcal{E}_c\}$, a přiřadíme-li každé hraně váhu, tak potom lze dle článku [5] převést minimalizaci funkce (1) na řešení „*multiway cut*“ problému pro neorientovaný graf $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. Pro hrany z \mathcal{E}_p se váha počítá z $V_{p,q}$ a pro hrany náležící do \mathcal{E}_c z D_p .

Je nutné poznamenat, že způsob výpočtu proměnných D_p a $V_{p,q}$, a složení množin C a \mathcal{E}_c není nijak striktně daný a pro různá řešení se může lišit (naše volba viz III. kap.)



Obrázek 1- Vlevo je původní kresba se zakreslenými čáry. Vpravo je požadovaný výsledek po obarvení. [2]

B. Kategorizace problému

V případě, že se v kresbě přiřazují právě dvě barvy, tedy množina \mathcal{L} obsahuje dva prvky, je možné problém vyřešit v polynomiálním čase (problém minimálního řezu/maximálního toku). Avšak, pro více terminálů se tento problém stává NP-složitým[4].

II. SOUVISEJÍCÍ PRÁCE

V článku od Y. Boykova [3] je popsán algoritmus (dále jako BK alg.) na řešení úlohy minimálního řezu/maximálního toku, který řeší problém značení pro dvě značky. Ten je při použití na specifických úlohách počítačového vidění rychlejší než jiné, do té doby používané, algoritmy.

Avšak, v naší úloze se počítá s tím, že množina \mathcal{L} bude obsahovat více jak dva prvky. Jak v článku od Y. Boykova [1], tak i v článku od D. Sýkory [2] je tento problém řešen tak, že se pro každou značku z \mathcal{L} se iterativně aplikuje algoritmus na hledání minimálního řezu/maximálního toku. V prvním zmíněném článku se pojednává o tzv. α -expansi, ve druhém článku o nástroji *LazyBrush*. Jedním z rozdílů mezi těmito dvěma postupy je v tom, jaké prvky obsahují množiny \mathcal{C} a \mathcal{E}_c . Dalším rozdíl je v tom, že algoritmus α -expansi v každé iteraci pro výpočet minimálního řezu používá celý graf, kdežto *LazyBrush* metoda postupně v každé iteraci redukuje velikost grafu, z něhož se minimální řez počítá. To má za následek to, že *LazyBrush* je rychlejší, avšak výsledek je silně závislý na pořadí, ve kterém jsou značky v \mathcal{L} seřazeny [2].

III. ŘEŠENÍ PROBLÉMU

A. Návrh řešení

Řešení našeho problému lze rozdělit na dvě části. První z nich je vyřešit úlohu značkování pro dvě značky. Druhou částí, stěžejní, je vyřešit značkování pro více značek, a to za využití znalosti získané vyřešením předcházejícího podproblému.

Jak už bylo zmíněno ve II. kapitole, problém binárního značení lze řešit pomocí algoritmů zabývajících se problémem minimálního řezu/maximálního toku. V této práci k tomu použijeme jednak Ford-Fulkersonovu metodu (dále F-F), tak i BK algoritmus popsáný v článku [3]. Jeho autoři tvrdí, že horní asymptotická složitost tohoto algoritmu je $\mathcal{O}(EV^2|C|)$, kde $|C|$ je cena minimálního řezu, E je počet hran a V počet vrcholů v odpovídajícím grafu. Což je horší, než složitost u standardních algoritmů ($\mathcal{O}(E|C|)$ pro metodu F-F a $\mathcal{O}(VE^2)$ pro algoritmus Edmonds-Karp, jež vychází z metody F-F), ale při praktickém použití je tento algoritmus rychlejší než zmíněné standardní postupy.

K vyřešení značení pro více značek byla dána přednost algoritmu *LazyBrush* oproti α -expansi. Důvodem je jeho avizovaná rychlost, která je v této práci hlavním kritériem.

B. Implementace

Zvolené metody pracují s grafy, proto je nejprve nutné si ze vstupních dat (šedo-tónová kresba a v ní uživatelem zakreslené barevné čáry) takový graf vytvořit.

Tvorba grafu

Jak bylo uvedeno v úvodní kapitole, náš graf \mathcal{G} se bude skládat z vrcholů $\mathcal{V} = \{P, C\}$ a hran $\mathcal{E} = \{\mathcal{E}_p, \mathcal{E}_c\}$. Množina P bude obsahovat vrchol pro každý z pixelů v obrázku a C bude obsahovat pomocný vrchol pro každou z použitých barev. \mathcal{E}_p bude množina hran, tvořená každou dvojicí přilehlých pixelů $p, q \in P$ a bude jim přiřazena cena $w_{p,q}$. \mathcal{E}_c budou hrany spojující pomocný vrchol $c \in C$ s pixelem $p \in P$, jež byl obarven čarou barvy c , a bude jim přiřazena cena $w_{p,c}$ (viz Obrázek 2).

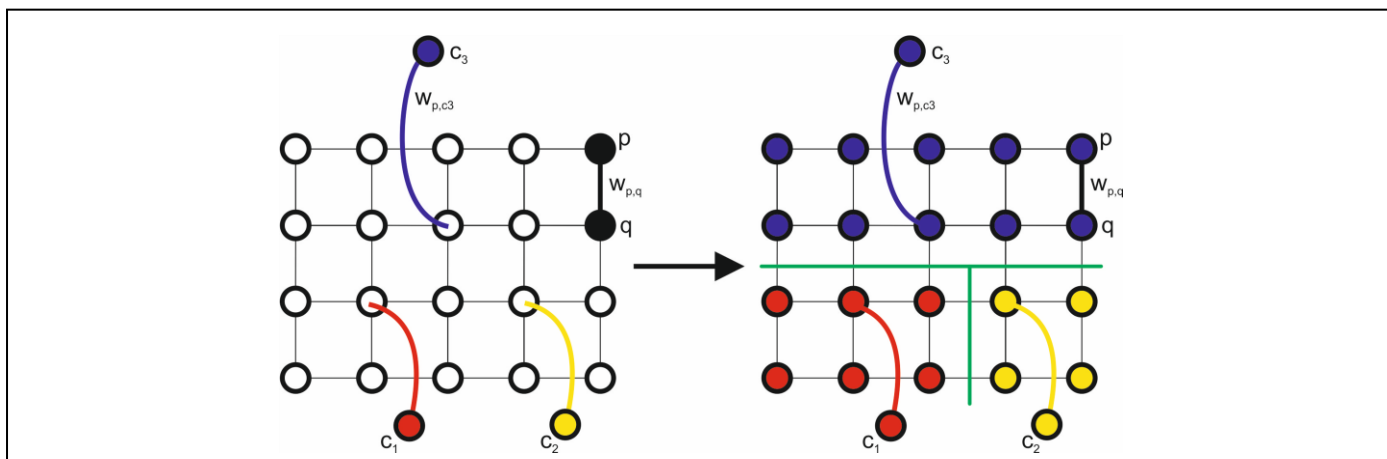
Výpočet ceny

Výsledek řešení je mj. závislý na tom, jak se budou počítat ceny pro jednotlivé hrany. Pro hrany \mathcal{E}_p bude cena $w_{p,q}$ vycházet z energie nespojitosti $V_{p,q}(c_p, c_q)$, která zde bude reprezentovat změnu barvy mezi dvěma pixely. Přechod mezi dvěma barvami je v našem případě žádoucí právě v takovém místě kresby, kde se nachází hrana kresby, tj. intenzita pixelů je nízká. Díky tomu pak výsledná kresba bude obsahovat jednoduše barevně monotónní plochy oddělené hranami původní kresby. V souladu s tímto faktem, bude hodnota energie $V_{p,q}$ následující:

$$w_{p,q} = V_{p,q} = \begin{cases} K \cdot \beta + 1 & \text{pro } c_p \neq c_q \\ 0 & \text{jinak} \end{cases} \quad (3)$$

Kde $\beta = e^{-A \cdot A/\sigma} \in (0; 1)$, A je rozdíl intenzity jasu mezi pixelem p a q , tedy $A = I_p - I_q$, σ je empiricky zjištěná konstanta (zde 0.0012) a $K = 2 \cdot (h + w)$ (h je výška a w je šířka vstupního obrázku), přičemž tento vztah splňuje všechny podmínky z I. kapitoly. Tím dosáhneme toho, že v místech přechodu bude energie minimální, kdežto v místech, kde je bílá plocha k vybarvení, bude energie maximální.

Cena hran z množiny \mathcal{E}_c bude závislá na energii $D_p(c_p)$, jejíž hodnota bývá v obdobných aplikacích odvozena z intenzity barvy c_p . Avšak, zde bude velikost D_p závislá na typu zakreslené čáry, která může být buď to *silná*, *slabá* nebo *žádná*. Díky tomu bude



Obrázek 2 – Vlevo struktura grafu vytvořeného ze vstupních dat. Vpravo výsledné přiřazení značek jednotlivým pixelům. [2]

barvení rezistentní vůči malým přetahům slabých čar, čehož uživatel bude moci využít při určování barvy pro malé anebo komplikované oblasti. Toto chování se zajistí tak, že hodnota D_p bude pro slabé čáry penalizována:

$$D_p(c_p) = \lambda \cdot K \quad (4)$$

Kde $\lambda \in \langle 0,1 \rangle$, přičemž $\lambda = 1$ je pro pixely, přes které byly nakresleny *silné* čáry a $\lambda = 0$ pro pixely, přes které nevedou *žádné* barevné čáry (proto pak není potřeba vést hrany mezi každým pixelem a pomocným vrcholem). Abychom docílili toho, že energie bude největší pro *silné* čáry a menší pro *slabé*, bude výsledný vztah pro cenu $w_{p,c}$ následující:

$$w_{p,c} = K - D_p = \begin{cases} K & \text{pro silné čáry} \\ (1 - \lambda) \cdot K & \text{pro slabé čáry} \end{cases} \quad (5)$$

Algoritmus LazyBrush

Jedná se o hladový algoritmus, který řeší několikanásobné značení grafu. Zhruba řečeno, postupně se pro každou z použitých barev vyřeší problém maximálního toku/minimálního řezu pro binární značení. Zde je bodově popsán postup:

1. Pro vstupní obraz o rozměrech $h \times w$, inicializuj množinu zakreslených barev \mathcal{C} a vytvoř masku $\mathcal{M} \in \mathbb{N}^{h \times w}$ neoznačených pixelů.
2. Pro každý pixel, který byl označen barevnou čarou, přiřaď v masce \mathcal{M} příslušnou značku.
3. Pokud \mathcal{C} je prázdné, ukonči program.
4. Vyber libovolnou barvu $c \in \mathcal{C}$.
5. Vytvoř graf \mathcal{G} z neoznačených pixelů z \mathcal{M} , přičemž cena hran bude $w_{p,q}$ (viz rovnice 3).
6. Do grafu přidej uzly \mathcal{S} a \mathcal{T} (zdroj a spotřebič). Pixely p , které jsou obarveny čarou barvy c , spoj hranou \mathcal{E}_c o ceně $w_{p,c}$ (viz rovnice 5) s \mathcal{S} , a pixely, které jsou obarvené jinou barvou, spoj s \mathcal{T} .
7. Vyřeš problém maximálního toku/minimálního řezu pro binární značení v grafu \mathcal{G} a uzly \mathcal{S} a \mathcal{T} (viz níže).
8. Pro pixely, již byly přiřazeny ke zdroji \mathcal{S} , nastav hodnotu značení v masce \mathcal{M} na c .
9. Odeber barvu c z \mathcal{C} a přejdi na krok (3).

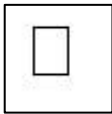
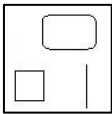
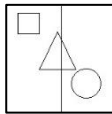
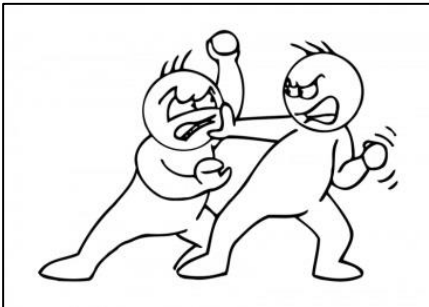
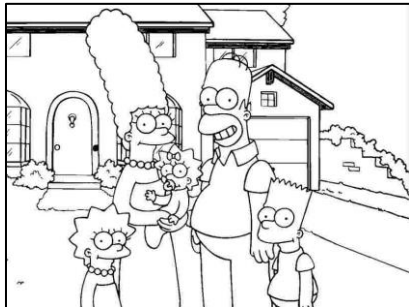
Metoda Ford-Fulkerson

Pomocí této metody lze vyřešit problém maximálního toku. Vstupem je graf $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ a indexy \mathcal{S} a \mathcal{T} určující který z vrcholů je zdrojem a spotřebičem. Úlohu lze formulovat přes lineární programování:

$$\begin{aligned} & \max \sum_{e \in \delta^+(s)} f(e) - \sum_{e \in \delta^-(s)} f(e) \\ & \text{s. t. } \sum_{e \in \delta^-(s)} f(e) = \sum_{e \in \delta^+(s)} f(e) \quad v \in \mathcal{V} \setminus \{\mathcal{S}, \mathcal{T}\} \\ & \quad f(e) \leq c(e) \quad e \in \mathcal{E} \end{aligned}$$

Kde δ^+ resp. δ^- je množina výstupních, resp. vstupních hran z/do vrcholu s , $c(e)$ je cena pro hranu e a $f(e)$ je aktuální tok na hraně e . Minimální řez získáme z výpočtu maximálního toku (Ford-Fulkersonův teorém). Z toho lze poté určit, které z vrcholů grafu jsou dosažitelné ze zdroje \mathcal{S} , a těm pak přiřadit značku, kterou \mathcal{S} reprezentovalo.

Způsobů, jak tuto metodu implementovat je několik. Zde byla vybrán algoritmus Edmonds-Karp, jehož asymptotická složitost je $\mathcal{O}(VE^2)$ (V je počet vrcholů grafu a E je počet hran).

Testovací obrázky – sada 1			
ID	1	2	3
Rozlišení	50x50	90x90	150x150
Náhled			
Testovací obrázky – sada 2			
ID	4	5	
Rozlišení	364x258	800x600	
Náhled			

Tabulka 1 - Soupis použitých obrázků k testování

IV. VÝSLEDKY EXPERIMENTU

A. Testování

V této práci byly provedeny testy porovnávající rychlost obarvování pro 3 různé varianty algoritmů (viz kap. III.B.):

- *LazyBrush* používající k binárnímu barvení BK algoritmus (dále jako LB-BK),
- *LazyBrush* používající k binárnímu barvení *Edmonds-Karp* algoritmus (dále jako LB-EK),
- α -*expanse* používající k binárnímu barvení *GridCut* algoritmus.

Testovacími instancemi zde byly 2 sady obrázků. První sada sloužila ke zjištění doby trvání barvení pro všechny tři varianty. Jednalo se o tři obrázky obsahující základní geometrické obrazce s poměrně nízkým rozlišením. Druhá sada zahrnovala dva obrázky s komplexnější „scénou“ a jejich účelem bylo porovnat kvalitu barvení, tj. správnost přiřazení jednotlivých značek každému pixelu (viz Tabulka 1).

Testování bylo provedeno na osobním laptopu s následující konfigurací:

- operační systém: Windows 10 Education 64bit,
- procesor: Intel Core i5-4200M 2.5GHz,
- paměť: 8192MB RAM.

Algoritmy α -*expanse* a BK zde nebyly přímo implementovány, ale byly použity volně dostupné zdrojové kódy [6] (BK je zde v optimalizované podobě jako *GridCut*).

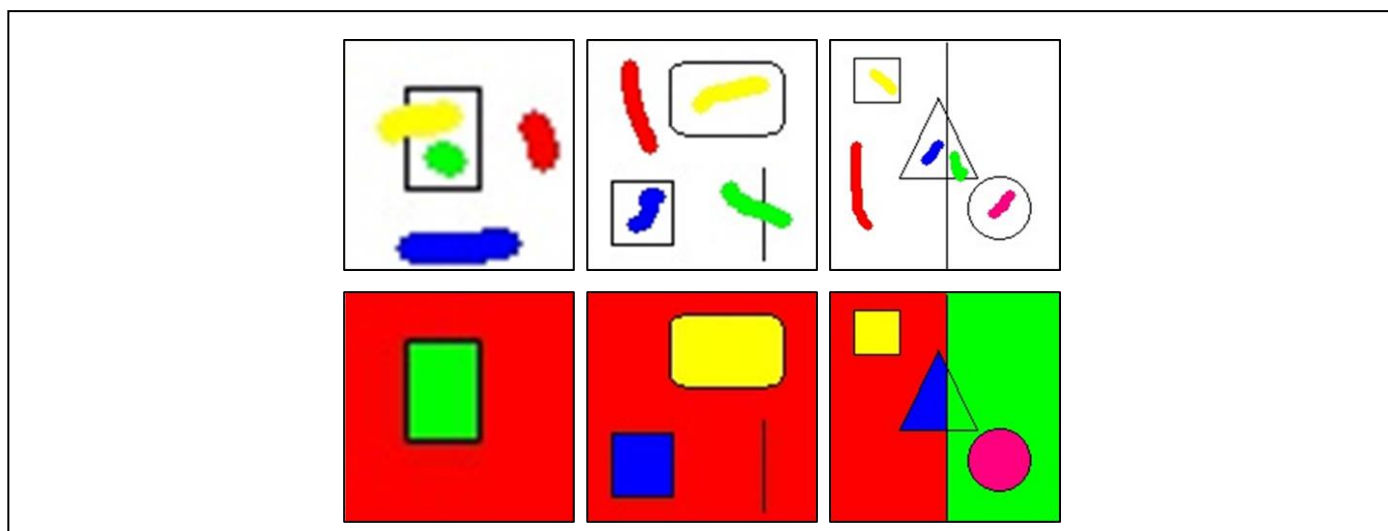
B. Výsledky

Pro obrázky z první sady, se při zakreslení každé nové čáry zaznamenalo, jak dlouho proces barvení trval. Tím se zjistilo, jak moc je ovlivněna rychlost barvení se zvětšujícím se počtem barev. Čas se měřil u všech třech typů algoritmů. U druhé sady se již testoval pouze algoritmus LB-BK a α -*expanse* a zde se již měřil čas CPU jen u finálního barvení, kdy už byly do obrázku zakresleny všechny požadované čáry.

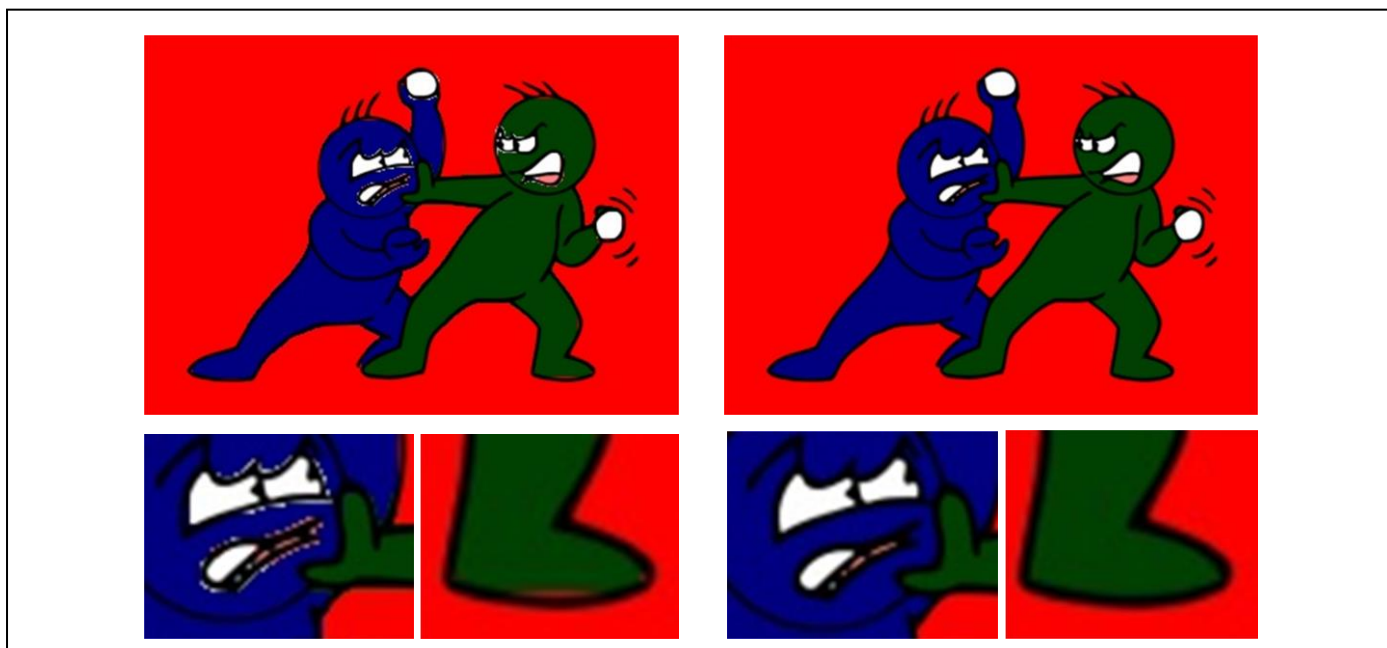
Co se kvality barvení týče, ta se ověřovala tak, že se u obrázků v první sadě naschvál zakreslili přesahy slabýma čarami. V obrázcích z druhé sady vyskytovali malé oblasti k vybarvování (viz detail u Obrázku 4 a 5), což rovněž posloužilo k ověření chování algoritmů.

Čas CPU (v ms) potřebný na vykonání barvení				
ID obrázku	Počet barev	Typ algoritmu		
		LB-EK	LB-BK	α -expanse
1	1	82	1	17
	2	2748	2	22
	3	8893	4	25
	4	12618	6	42
2	1	617	3	51
	2	26916	5	60
	3	43996	7	63
	4	112597	11	81
3	1	<i>přetečení zásobníku</i>	9	140
	2	-	11	160
	3	-	17	211
	4	-	25	227
	5	-	32	295
4	5	-	143	2010
5	19	-	899	45140

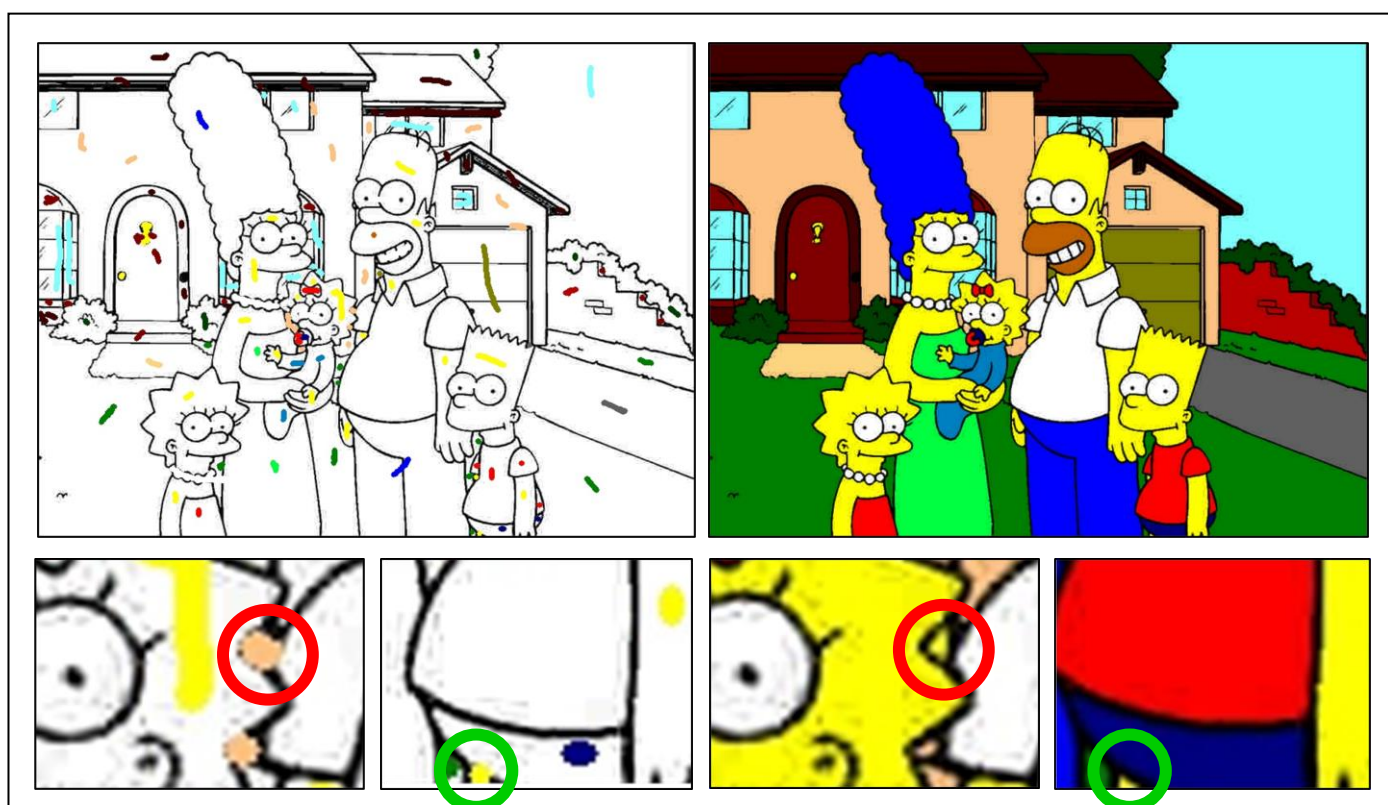
Tabulka 2 – Výpočetní náročnost procesu barvení v závislosti na velikosti obrázku a počtu barev.



Obrázek 3 – Výsledek barvení obrázků z první sady (seřazeno dle ID vzestupně zleva doprava). Nahoře jsou vstupní obrázky se zakreslenými čarami, kdy slabými čarami byly u prvního obrázku žlutá a modrá, u druhého žlutá, modrá a zelená, a u třetího byly všechny silné. Dole jsou obrázky po obarvení.



Obrázek 5 – Výsledek barvení pro obrázek 4. V levé polovině je výstup po použití α -expansce, v pravé po použití LB-BK. V dolním řádku jsou zobrazeny detaily těch částí obrázku, kde se pro α -expansi vyskytlo chybné přiřazení barvy.



Obrázek 4 – Výsledek barvení pro obrázek 5. Vlevo je vstupní obrázek se zakreslenými čarami ve finální podobě. Vpravo výsledek barvení za použití LB-BK. V dolním řádku jsou zobrazeny dva detaily, první z nich poukazuje na špatně přiřazenou barvu, druhý na správně přiřazenou.

C. Diskuse

Z Tabulky 1 je jasně patrné, že použití běžného algoritmu (LB-EK) na řešení binárního barvení, je v naší úloze rychlostně nevyhovující – aplikace by nebyla schopna v reálném čase obarvovat kresbu. Takovýto algoritmus je navíc i paměťově náročný, což se projevilo tak, že už pro obrázky o velikosti 150x150 pixelů, docházelo na testovacím stroji k přetečení zásobníku, a tedy program ani nedoběhl. Rovněž se potvrdilo, že algoritmus LazyBrush je také rychlejší, než α -expansce, a to velmi markantně.

Při vyhodnocení kvality barvení, lze říci, že algoritmus LazyBrush přiřazuje barvy pixelům ve většině případů správně. Ovšem v případě, že je malý region, jež je obarven slabou čarou, obklopen z větší části regionem, který byl obarven silnou čarou, tak může dojít k nesprávnému přidělení barvy (viz Obrázek 5). Tomu by šlo předejít buďto zakreslením silné čáry místo slabé (mohlo by ale hrozit v případě přetahu k zanesení špatné barvy), nebo pokusit se kreslit silné čáry co nejdále od původních křivek kresby a nekreslit je příliš dlouhé. Jak je vidět z Obrázku 4, tak nejenže je LB-BK rychlejší než α -expanse (viz Tabulka 1), ale také dává přesnější výsledky.

V. ZÁVĚR

Úkolem této semestrální práce bylo vytvoření aplikace pro interaktivní barvení kreseb, a to co v nejkratším čase.

Na začátku bylo předpokládáno, že nejvhodnějším řešením bude algoritmus LazyBrush (viz kap. II.). Aby mohl být tento předpoklad potvrzen nebo vyvrácen, použili se k porovnání další dva možné postupy (viz kap. IV.A). Ukázalo se, že předpoklad byl správný, a že tedy pomocí alg. LazyBrush lze vytvořit takový nástroj, který bude přímo po každém zakreslení barevné čáry do kresby, schopný rychle vypočítat výsledek.

REFERENCE

- [1] Y. Boykov, O. Veksler and R. Zabih, Fast Approximate Energy Minimization via Graph Cuts, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 11, pp. 1222-1239, Nov 2001.
- [2] D. Sýkora, J. Dingliana S. and Collins, LazyBrush: Flexible Painting Tool for Hand-drawn Cartoons. Computer Graphics Forum, vol. 28, no. 2, pp. 599–608, 2009.
- [3] Y. Boykov, V. Kolmogorov, An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 9, pp. 1124-1137, Sept 2004.
- [4] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, M. Yannakakis, The complexity of multiterminal cuts, SIAM Journal on Computing, pp. 864-894, 1994.
- [5] Y. Boykov, O. Veksler, R. Zabih, Markov Random Fields with Efficient Approximations, in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 648-655, 1998.
- [6] "Gridcut: Home". Gridcut.com. N.p., 2017. Web. 14.5. 2017.