České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačové grafiky a interakce

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Pavlína Ulrichová**

Studijní program: Softwarové technologie a management
Obor: Web a multimedia

Název tématu: **Editace obrazu v gradientní oblasti**

Pokyny pro vypracování:

Seznamte se s metodou editace obrazu v gradientní oblasti popsanou v článcích [1,2]. Navrhněte a implementujte jednoduchý obrázkový editor, který umožní provádět základní operace s obrazem v gradientní oblasti (kreslení pomocí gradientu, retušovací štětec, výběr a vkládání výřezu metodou přímého nahrazení, součtu a maximálního gradientu). Implementaci proveďte v jazyce C++ s využitím knihoven Qt a MKL. Editor otestujte na sadě obrázků, které dodá vedoucí práce.

Seznam odborné literatury:

[1] Pérez et al.: "Poisson image editing", ACM Transactions on Graphics 22(3):313-318, 2003.
[2] McCann & Pollard: "Real-time gradient-domain painting", ACM Transactions on Graphics 27(3):93, 2008.

Vedoucí: Ing. Daniel Sýkora, Ph.D.

Platnost zadání: do konce zimního semestru 2013/2014

prof. Ing. Jiří Žára, CSc.
vedoucí katedry

L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 14. 2. 2013

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Bakalářská práce

Editace obrazu v gradientní oblasti

Pavlína Ulrichová

Vedoucí práce: Ing. Sýkora Daniel, Ph.D.

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Web a multimedia

22. května 2013

Poděkování

Ráda bych poděkovala vedoucímu této bakalářské práce, panu Ing. Danielovi Sýkorovi, Ph.D., za konzultace a rady poskytnuté během vzniku této práce.

Prohlášení

Prohlašuji, že jsem práci vypracovala samostatně a použila jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 22. 5. 2013 .....

Abstract

This thesis tackles the issue of gradient editing of digital pictures and realization of a simple gradient picture editor. We describe elementary information about gradients, their use in practical tools for image editing and foremost an implementation solution of the gradient editor with use of Math Kernel Library. This thesis presents picture results of our editor's work and also an evaluation of its practical usage, for example a feedback time for the user in dependence on the picture size.

Abstrakt

Tato práce se zabývá gradientními editacemi digitálních obrazů a realizací jednoduchého gradientního obrazového editoru. Popisujeme zde základní informace o gradientech, jejich využití v praktických nástrojích pro úpravu obrazu, a především pak řešení implementace gradientního editoru s využitím knihovny Math Kernel Library. Práce prezentuje obrazové výsledky práce našeho editoru a taktéž zhodnocení jeho využitelnosti v praxi, jako například čas zpětné vazby uživateli v závislosti na velikosti obrazu.

Obsah

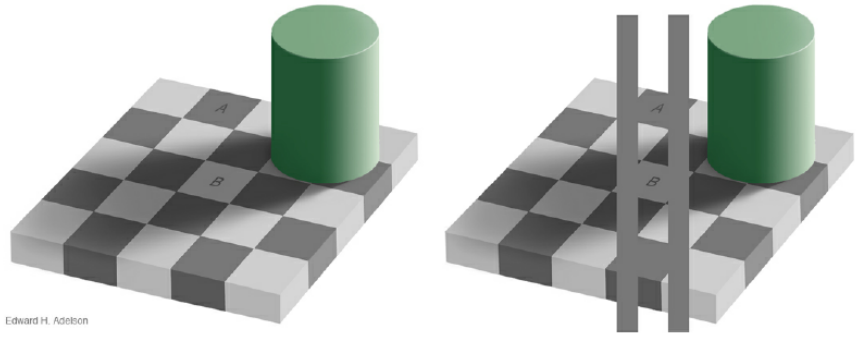
Seznam obrázků

Kapitola 1

Úvod

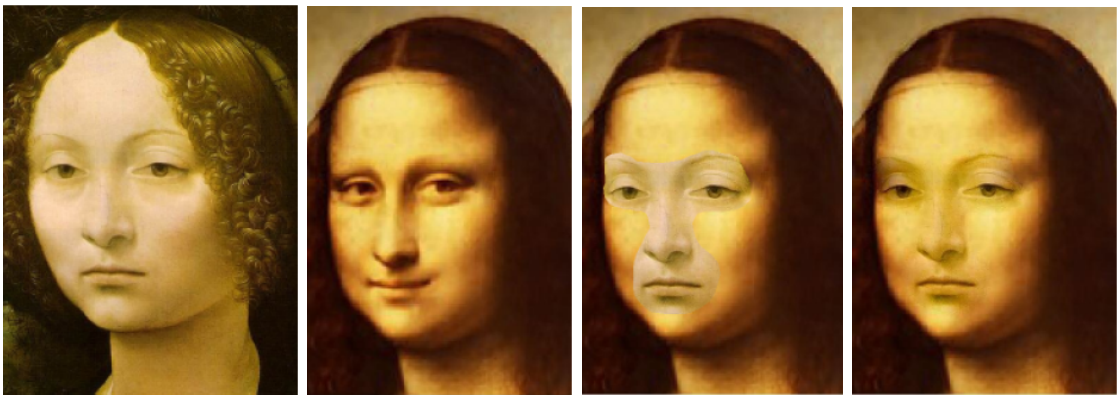
Gradient čili lokální změna intenzity mezi sousedními pixely je základ pro nové nástroje na digitální úpravu obrazu. Ty představují alternativu k běžným obrazovým editorům, které pracují přímo s barevnou intenzitou jednotlivých bodů.

Jak Edward H. Adelson v rámci své práce *Lightness Perception and Lightness Illusions* [5] ukazuje, Human visual system (HVS, systém lidkého vizuálního vnímání) se soustředí spíše než na konkrétní hodnotu intenzity na lokální změny této intenzity mezi dvěma sousedními body obrazu, tedy na gradient. Dokazuje to i jeden z optických klamů (Obrázek 1.1), kdy se na první pohled čtverce A a B zdají mít odlišný odstín šedé barvy, ačkoliv ve skutečnosti jsou tyto odstíny totožné.



Obrázek 1.1: Optický klam demonstrující citlivost na lokální změny intenzity. Převzato z [9].

Jedním z důvodů, proč se gradientními úpravami zabýváme, je fakt, že při editacích obrazu nedochází ke skokovým změnám jasů. Přechody zůstávají plynulé, i když pracujeme pouze s částí obrazu. Gradientní úpravy také umožňují větší svobodu obrazových editací a přitom nezpůsobují tak výraznou změnu ve výsledném obrazu, jako je tomu u intenzitních úprav (Obrázek 1.2).



Obrázek 1.2: Ukázka gradientního bezešvého klonování. Převzato z [7].

V současné době neexistuje plnohodnotný editor pracující na základě gradientních úprav. Lze však nalézt dema, ve kterých je možné provádět některé základní obrazové editace pomocí gradientních úprav uživatelsky snáze, než by se stejná editace provedla pomocí úprav založených na práci s intenzitami.

Cílem této práce je prostudování gradientního zpracování digitálního obrazu, jeho aplikace v praxi, a především pak implementace jednoduchého gradientního obrazového editoru.

Kapitola 2

Teorie

Práce s obrazem v gradientní oblasti funguje na principu jisté modifikace gradientního pole a následné integrace nové podoby obrazu z tohoto pozměněného pole. Po každé změně v gradientním poli se musí dopočítat aktuální hodnota barevné intenzity pro každý pixel obrazu.

2.1 Gradient a gradientní pole

Gradientní pole je pole vektorů. Pro 2D obrazy platí, že každý takový vektor se skládá ze dvou složek – hodnoty gradientu ve směru souřadnicové osy x (dále jen g_x) a hodnoty gradientu ve směru souřadnicové osy y (dále jen g_y) (Obrázek 2.1) [6]. Gradient se označuje operátorem nabla ∇ a je to diferenciální operátor definovaný jako

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right), \tag{2.1}$$

kde f je skalární funkce definující obraz a x a y jsou proměnné funkce f. V praxi gradient vyjadřuje lokální změnu intenzity mezi sousedními jednotkami barevného obrazu. Směr vektoru odpovídá směru největší změny ve skalárním poli obrazu a velikost vektoru je rovna velikosti změny.

2.2 Konzervativita pole

Pokud chceme z gradientního pole přímo integrovat obraz, musí být toto pole konzervativní. Pro každé konzervativní pole platí, že integrál po libovolné uzavřené křivce v daném poli je roven nule [9]

$$\oint G = 0. \tag{2.2}$$

Obrázek 2.1: Příklad původního obrazu a jeho rozkladu na gradientní pole v ose x a y. Převzato z [9].

Při některých změnách v gradientním poli může dojít k porušení konzervativity, což se projeví nenulovostí integrálu.

2.3 Poissonova rovnice

Pokud pole gradientů konzervativní není, pak nelze obraz integrovat přímo. Můžeme ale najít obraz I^*, který má konzervativní pole gradientů a těmito svými gradienty se co nejvíce blíží očekávanému poli změněných gradientů. Matematicky lze tento minimalizační problém zapsat [7]

$$I^* = arg\, min_I \iint \| \nabla I - G \|^2, \tag{2.3}$$

kde argument dvojného integrálu vyjadřuje rozdíl mezi gradienty původního obrazu a očekávanými gradienty výsledku. S pomocí Euler-Lagrangeovi rovnosti lze minimalizační problém převést na odpovídající parciálně diferenciální rovnici [9]

$$\Delta I = div\, G, \tag{2.4}$$

která se nazývá Poissonova. I zde reprezentuje hledanou funkci obrazu, $div\, G$ je očekávaná divergence gradientů, kterou definujeme jako parciální derivaci jednotlivých složek gradientního vektoru [7]

$$div\, G = \frac{\partial g_x}{\partial x} + \frac{\partial g_y}{\partial y} \tag{2.5}$$

a znak delta (Δ) zastupuje Laplaceův operátor. Ten definujeme jako operátor divergence gradientu daného skalárního pole. Pro různě velké dimenzionální prostory má odlišnou podobu. Ve 2D prostoru ho zapíšeme jako (2.6) [7]

$$\Delta = \nabla^2 = \nabla \cdot \nabla = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}. \tag{2.6}$$

Jedná se tedy o druhou parciální derivaci podle proměnných x a y. Poissonova rovnice má obecné a partikulární řešení. My se snažíme najít partikulární řešení této rovnice, k jehož získání potřebujeme mít počáteční podmínky.

2.4 Počáteční podmínky

Počáteční podmínky jsou jedním z faktorů, které ovlivňují jednoznačnost řešení Poissonovy rovnice nad určenou oblastí. Těchto podmínek je více druhů, ale v našem případě se zaměřujeme pouze na dva nejběžnější druhy, a to Neumannovy a Dirichletovy [8]. Podmínky lze na jednu oblast aplikovat jednotlivě, nebo je kombinovat dohromady.

V případě Dirichletových počátečních podmínek máme definované funkční hodnoty v pevně daných bodech. Pro použití v Poissonově rovnici tedy platí

$$f(x) = f_x, \tag{2.7}$$

kde x je libovolný bod s počáteční podmínkou, f je reálná funkce a f_x je konkrétní funkční hodnota v daném bodě. Body s počátečními podmínkami se nemusí nutně nacházet pouze v okrajových bodech oblasti.

Neumannovy počáteční podmínky jsou definovány tak, že normály zadaných bodů se rovnají nule. U těchto podmínek bude při použití v Poissonově rovnici platit

$$f'(x) = 0, \tag{2.8}$$

tedy že první derivace funkce v daném bodě je rovna nule.

2.5 Diskretizace a řešení Poissonovy rovnice

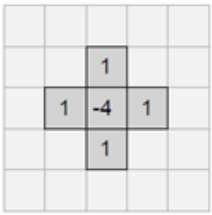
Díky tomu, že neznáme podobu funkce definující oblast, nedokážeme Poissonovu rovnici řešit pomocí analytických postupů. V takovém případě můžeme Poissonovu rovnici zdiskretizovat, a tím jí převést na odpovídající soustavu lineárních rovnic [9]. Tuto soustavu lze následně převést do maticového tvaru s rozměry

$$w' = h' = w \cdot h, \tag{2.9}$$

kde w' a h' odpovídají počtu sloupců a řádků matice a w a h jsou šířka a výška počítané oblasti. Tuto zkonstruovanou matici označíme jako matici A.

Každý řádek matice A bude obsahovat maximálně pět nenulových prvků (konkrétně -4,1,1,1,1), protože každá rovnice této soustavy odpovídá diskretizovanému Laplaceovu operátoru pro jeden bod [8]. Ten ve 2D prostoru vzniká konvolucí obrazu s jádrem (Obrázek 2.2) a má podobu

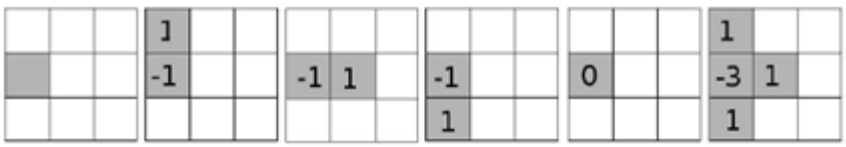
$$\nabla^2 f(x,y) \approx f(x-1,y) + f(x+1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y). \tag{2.10}$$



Obrázek 2.2: Laplaceův operátor pro 2D prostor. Převzato z [8].

Diskretizují se také počáteční podmínky. V případě Dirichletových pořátečních podmínek se diskretizace projeví tak, že do rovnice se na příslušné místo umístí jednička a do odpovídajícího místa na pravé straně se uloží hodnota podmínky [8]. V maticovém zápisu to znamená, že jednička bude zapsána na odpovídajícím řádku hlavní diagonály a hodnota podmínky bude na odpovídajícím místě ve vektoru pravých stran.

V případě Neumannových počátečních podmínek potřebný Laplaceův operátor pro danou lineární rovnici vznikne konvolucí všech směrových derivačních operátorů, které mají podobu [1 - 1] (Obrázek 2.3). Pokud má některý z nich hodnotu nula, pak se neprojeví [8].

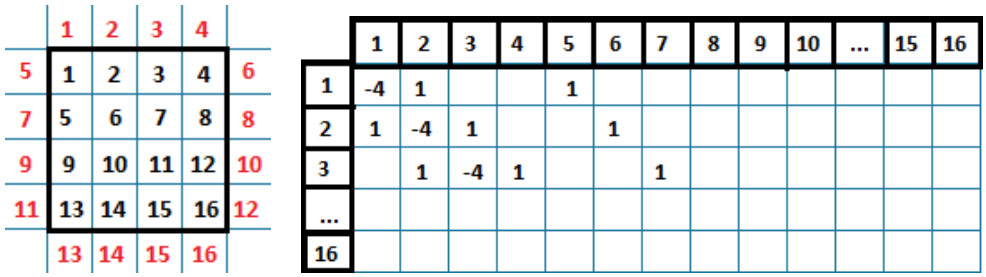


Obrázek 2.3: Diskretizace Neumannových vstupních podmínek pro levý kraj. Převzato z [8].

Pokud tedy budeme mít například pro oblast velikosti 4 x 4 body s Neumannovými počátečními podmínkami v okrajových bodech (Obrázek 2.4). Pak jednotlivé řádky matice

budou tvořeny distrétními laplaciany (2.10) a výsledná matice A bude mít podobu

$$\begin{pmatrix}
-4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -4 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4
\end{pmatrix}$$



Obrázek 2.4: Příklad počítané oblasti (černě orámované body) a počátečních podmínek (červené body) a náznak tvorby rovnic bez pravých stran pro body 1, 2 a 3. Inspirováno obrázky z [9].

Díky tomu můžeme z pozorování konstatovat, že počet nulových prvků matice výrazně převyšuje počet prvků nenulových. Můžeme tedy celou soustavu lineárních rovnic označit za řídkou.

Vysokého počtu nulových prvků se využívá například při ukládání takovýchto matic do paměti. Neukládá se celá matice, ale pouze informace o souřadnicích (sloupec, řádek) a hodnotách nenulových prvků. Rozměry této matice narůstají úměrně s velikostí vstupní oblasti.

Zkonstruovanou matici A potom můžeme dosadit do základní maticové rovnosti

$$Av = b, \tag{2.11}$$

kde A je maticový tvar soustavy lineárních rovnic bez pravých stran, v je hledaný vektor řešení a b je vektor pravých stran. Ten v oblasti gradientních úprav odpovídá očekávaným gradientům řešení a hodnotám počátečních podmínek [9].

2.5.1 Řešení velkých řídkých matic

Pro řešení velkých řídkých soustav lineárních rovnic máme dva přístupy, které využívají velkého počtu nulových prvků k efektivnějšímu řešení. Tyto přístupy jsou přímý a iterativní [8].

Metody založené na přímém principu řešení velkých řídkých matic poskytují po konečném počtu kroků přesný výsledek soustavy. Základem tohoto principu je převedení řešené matice na ekvivalentní tvar, který se lépe řeší.

K řešení se využívají maticové rozklady, jako např. LU rozklad, QR rozklad a jiné. Před samotným rozkladem se matice přeskupuje tak, aby rozklad nebyl příliš hustý. Výhodami tohoto přístupu jsou velká přesnost a rychlost výpočtu. Nevýhodou jsou poměrně velké nároky na paměť, které během řešení narůstají.

Naproti tomu iterativní metody mají menší paměťové nároky, protože nepotřebují reprezentovat v jednom okamžiku celou matici. Při výpočtu se vždy vytvoří pouze jeden odpovídající řádek výpočtu, vyřeší se, výsledek uloží a pak se teprve přejde na další řádek v pořadí. Tím nenarůstají nároky na paměť ani během výpočtu.

Nevýhodou těchto metod může ale být míra jejich přesnosti a čas, za jaký se metoda k výsledku konverguje. Před začátkem řešení je tedy dobré definovat přijatelnou přesnost výpočtu, při jejímž dosažení se cyklus může zastavit, a maximální počet iterací, které se během výpočtu provedou.

U tohoto přístupu proto nikdy nedostaneme přesné řešení. Rychlost konvergence jednotlivých metod se pro různé případy liší a čím větší přesnost vyžadujeme, tím více klesá rychlost konvergence k výsledku.

Příkladem iterativních metod řešení jsou Jacobiova metoda, Gauss-Seidelova metoda, metoda konjugovaných gradientů a jiné.

Kapitola 3

Aplikace

3.1 Gradientní štětce

Gradientní štětec je základní nástroj pro aplikaci gradientních úprav analogický k těm, které se využívají v běžných intenzitních editorech. Od klasických štětců se liší v tom, že gradientní štětce musí pracovat se dvěma gradietními obrazy (dvěma složkami vektorového gradientního pole g_x a g_y)

$$g = (g_x, g_y). \tag{3.1}$$

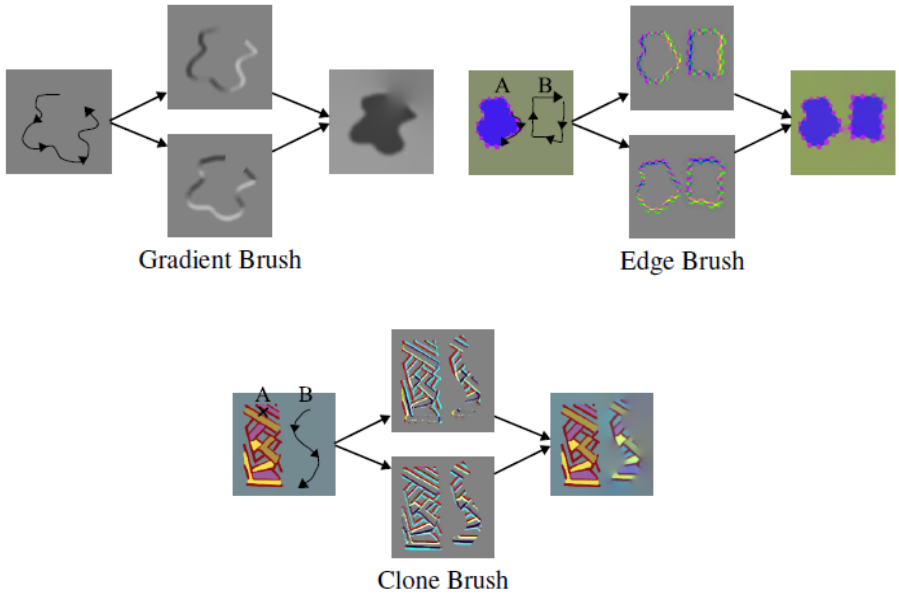
Výsledek ovlivní jak hodnota gradientu nastavená na štětci, tak i směr tahu. Hodnoty pro jednotlivé gradientní obrazy se rozkládají za pomoci Pythagorovy věty, kde složky g_x a g_y jsou odvěsny a kreslený tah je přepona pravoúhlého trojúhelníku. Příklady rozdělení typů gradientních štětců dle McCanna [6] jsou (Obrázek 3.1)

- čistý gradient brush (gradientní štětec), který pouze vysílá nastavené gradienty
- edge brush (hranový štětec), který označenou oblast gradientu kopíruje ve smyčce
- clone brush (klonovací štětec), který kopíruje gradienty v relativní pozici od označeného počátečního místa.

3.2 Typy míchání gradientů

Pracovat s gradienty můžeme různými způsoby. V případě editace obrazů je můžeme nahrazovat novými, slučovat je dohromady nebo upravovat dle jiných kritérií. Tyto změny se aplikují po jednotlivých pixelech a pro každý barevný kanál zvlášť. Příkladem toho jsou blendig módy, které jsou popsané v [6].

Definujme, že (g_x,g_y) jsou gradienty na pozadí upravovaného obrazu a (b_x,b_y) jsou gradienty, kterými uživatel pomocí gradientního štětce zasahuje do původního obrazu. V základě



Obrázek 3.1: Logika gradientních štětců. Převzato z [6].

se jednotlivé módy liší pouze tím, jakým způsobem se porovnávají hodnoty gradientu na pozadí a na gradientním štětci a jakým způsobem se z nich určí výsledná hodnota gradientu pro nový obraz.

Míchání přidáváním (additive blending)

Jedná se o základní mód, kdy se gradient štětce jednoduše přičte k hodnotě gradientu na pozadí upravovaného obrazu. Tento mód je vhodný především při črtání a úpravách stínů. Využívá se např. při základním gradientním kreslení.

$$(g_x, g_y) \longleftarrow (g_x, g_y) + (b_x, b_y) \tag{3.2}$$

Míchání maxima/minima (maximum/minimum blending)

Logika u tohoto módu je založena na porovnání hodnot gradientu mezi gradientním štětcem a pozadím obrazu. V případě maxima se ve výsledném obraze ponechá vyšší hodnota z těchto dvou.

$$(g_x, g_y) \longleftarrow \begin{cases} (g_x, g_y), & \text{pokud } |(g_x, g_y)| > |(b_x, b_y)| \\ (b_x, b_y), & \text{jinak} \end{cases} \tag{3.3}$$

Mód míchání maximem se hodí především na klonování a kopírování hran. Míchání minimem má logickou rovnost přesně opačnou a využívá se na příležitostné klonování hladkých

oblastí přes ty vzorkované, např. odstraňování vrásek.

Míchání přímým nahrazením (overblending)
Hodnota gradientu v původním obraze je plně nahrazena hodnotou na gradientním štětci. Tento přístup se využívá např. při bezešvém klonování nebo odstraňování textur.

$$(g_x, g_y) \longleftarrow (b_x, b_y) \tag{3.4}$$

Směrové míchání (directional blending)
Tento přístup pracuje tím způsobem, že při tahu gradientním štětcem posiluje hodnoty gradientů, které směřují stejným směrem jako tah a naopak utlumují gradienty mající opačný směr. Využívá se např. při kontrastním posílení určité oblasti nebo práci s osvětlením v obraze.

$$(g_x, g_y) \longleftarrow (g_x, g_y)\left(1 + \frac{b_x \cdot g_x + b_y \cdot g_y}{g_x \cdot g_x + g_y \cdot g_y}\right) \tag{3.5}$$

3.3 Nástroje využívající modifikace gradientu

Různé principy míchání gradientu se využívají při různých úpravách obrazů, fotomontážích nebo gradientním kreslení. Jako příklady jsou uvedeny nejčastěji používané nástroje spadající do kategorií bezešvého klonování (import a míchání gradientů) a úprav označené oblasti (úprava textury, změny osvětlení a lokální změny barvy).

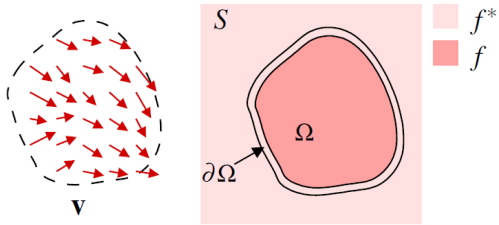
Pro matematické vysvětlení následujících nástrojů v souladu s prací [7], které využívá pro řešení problémů řízené interpolace (guided interpolation), definujeme a upřesňujeme následující označení (Obrázek 3.2) :

v – řídící pole (guidance field), což je vektorové pole očekávaných gradientů
f^* - je známá skalární funkce nad obrazem nebo jeho částí, bez počítané oblasti
f – je neznámá skalární funkce nad počítanou oblastí
Ω – označuje počítanou oblast
$\partial\Omega$ – značí oblast počátečních podmínek.

U bezešvého klonování a úprav oblastí jde o to, vhodně sestavit řídící pole v, které se pak dosadí do Poissonovy rovnice (2.4) s počátečními podmínkami $\partial\Omega$.

3.3.1 Bezešvé klonování

U úprav tohoto typu se ze dvou gradientních polí (zdroj a vkládaná oblast) vytváří dle různých pravidel řídící pole v.



Obrázek 3.2: Ilustrace k definici značení. Převzato z [7].

Vkládání objektů – import gradientů

Využívá metody přímého nahrazení gradientů, tedy pole očekávaných gratientů v (pravá strana výpočetní rovnice) je tvořeno přímo hodnotami gradientu z vkládaného druhého obrázku a nezanechává žádné původní gradienty v dané oblasti. Pak bude platit

$$v = \Delta g, \tag{3.6}$$

kde g je gradientní pole druhého zdrojového obrazu. Tento nástroj je vhodný pro přidávání a ubírání objektů z obrazu. Na rozdíl od ekvivalentních úprav v intenzitních editorech nejsou vidět žádné přechodové švy (Obrázek 3.3).

Obrázek 3.3: Příklad vkládání objektů. Převzato z [7].

Vkládání transparentních objektů - míchání gradientů

Pokud je potřeba pracovat s poloprůhlednými objekty nebo s objekty obsahující otvory, jako např. otvory mezi žebry kostlivce, pak se volí místo přímého nahrazení jako v předchozím případě metoda mixování gradientů.

Z uživatelského hlediska gradientní nástroje tohoto přístupu představují značné zjednodušení práce, protože není nutné pečlivě vybírat pouze požadovanou oblast přenesení, ale lze označit oblast větší. Stále však budou přitom průhlednost a otvory zobrazeny se stejně dobrým výsledkem (Obrázek 3.4).

Existují dvě možnosti, jak vytvořit řídící pole v. První a jednodušší možnost je taková, že se pouze utvoří lineární kombinaci zdrojového a cílového gradientního pole. Při tomto postupu však dochází ke ztrátě textury.

V druhém případě se využívá faktu, že Poissonova rovnice dovoluje využití nekonzervativních řídících polí [7]. Řídící pole v se pak vytvoří pro počítanou oblast tak, že v každém bodě je ponechána větší změna (3.7).

$$v(x) = \begin{cases} \Delta f^*(x), & \text{pokud } |\Delta f^*(x)| > |\Delta g(x)| \\ \Delta g(x), & \text{jinak} \end{cases} \tag{3.7}$$

Tento přístup se hodí i v případě, že vkládaná oblast je umístěná velmi blízko již existujícímu objektu ve zdrojovém obraze.

Obrázek 3.4: Příklad vkládání průhledných objektů. Převzato z [7].

3.3.2 Úpravy označené oblasti

Tento druh úprav vytváří řídící pole v pouze z pole gradientů daného zdrojem.

Zplošťování textury

Konstrukce řídícího pole v je zde založena na nelineární úpravě původního gradientního pole Δf^* nad danou oblastí [7]. Pole v tedy pak má podobu:

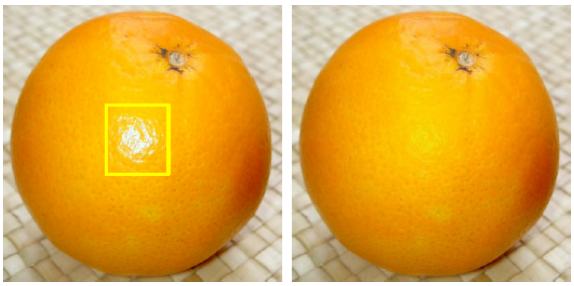
$$v(x) = M(x)\Delta f^*(x), \tag{3.8}$$

kde M je binární maska, pomocí které detekujeme oblasti zájmu. Masky mohou být různých druhů dle potřeby. Výhodnou volbou je maska detekující hrany.

Díky binární masce se odstraní malé detaily a výsledným efektem je vizuální zploštění textury. Míra zploštění závisí na hustotě binární masky. Čím je hustší maska, tím výraznější výsledný efekt bude.

Změny lokálního osvětlení

Smyslem tohoto nástroje je modifikace spekulárních odrazů světla na označené oblasti. Základní matematickou myšlenkou je, že gradietní pole logaritmu obrázku je transformováno tak, aby se posílily nízké hodnoty gradientu a naopak zeslabily ty vysoké. Z transformovaného pole v se pak zrekonstruuje logaritmus obrazu klasicky pomocí Poissonovy rovnice. Ta se počítá na celém obraze s užitím Neumannových okrajových podmínek.



Obrázek 3.5: Příklad změny lokálního osvětlení. Převzato z [7].

Mezi další nástroje, které jsou podrobněji popsané v [7] patří mimo jiné ještě lokální změny barev, editace vybrané oblasti (zesvětlení ztmavené popředí/pozadí) a jiné.

3.3.3 Jiné využití

Gradient painting

James McCann představuje v [6] obrazové nástroje založené právě na gradientních editacích. Díky různým druhům štětců (Sekce 3.1) a módům míchání (Sekce 3.2) může uživatel ovlivnit, které gradienty a jakým způsobem bude přenášet či jakým způsobem se budou navzájem ovlivňovat.

Obrázek 3.6: Kreslení pomocí gradientů. Převzato z [6].

Kapitola 4

Použité knihovny a rozhraní

4.1 Knihovna stb_image

Pro načítání obrázků ze souboru byla využita knihovna *stb_image* [4], která dokáže zpracovávat základní obrazové formáty jako JPEG, PNG (8 - bitová reprezentace), TGA, BMP, GIF a jiné.

Funkce pro načítání se nazývá *stbi_load* a jako vstupní parametry vyžaduje adresu k souboru, proměnné, do kterých se uloží informace o šířce a výšce obrazu, počet obrazových komponent na jeden pixel ve zdrojovém obrazu a mód načítání, podle kterého bude funkce reprezentovat jednotlivé pixely v poli. Data o obraze vrací funkce ve formátu unsigned char pole. Konkrétní volání vypadá následovně

```
unsigned char* obrazek=stbi_load(const char* source, int* width,
 int* height, int &n, 0);
```

Existují čtyři možné módy pro načítání dat. V závislosti na tom, který mód zvolíme, bude jeden pixel reprezentován různým počtem po sobě jdoucích hodnot ve výsledném poli obrazu. Módy jsou následující a korespodují různými s barevnými reprezentacemi obrazu:

- odstíny šedé (1 pixel = 1 hodnota v poli)
- odstíny šedé a průhlednost (1 pixel = 2 hodnoty v poli)
- RGB (1 pixel = 3 hodnoty v poli)
- RGB s průhledností (1 pixel = 4 hodnoty v poli)

Jednotlivé pixely pak lze adresovat následovně:

$$obrazek[i \cdot w + j],$$

kde *obrazek*[] je název proměnné, kde jsou uloženy hodnoty, w je šířka obrázku, i je sloupec a j řádek, na kterých se pixel nachází.

Toto platí v případě, že načteme obraz v odstínech šedé, tedy reprezentace jedné hodnoty na jeden pixel. Při jiných reprezentacích se v adresaci pixelů projeví zvětšená délka pole a posun, který určuje, se kterým kanálem obrazu zrovna pracujeme.

Zvolenému typu reprezentace musí odpovídat námi implementované metody získávající informace pro konstrukci výpočetní matice, neboť v nich záleží na pořadí prvků za sebou. Např. při módu načtení šedé jsou jednotlivé hodnoty přímo za sebou, zatímco u RGB módu jsou sousední hodnoty pro jeden barevný filtr uloženy ob tři prvky (pro každý barevný kanál jeden údaj).

4.2 Intel MKL DSS

4.2.1 Základní informace

Math Kerner Library Direct Sparse Solver (MKL DSS) [2] je knihovna od firmy Intel, která se specializuje na řešení velkých řídkých matic. Je určená pro jazyky Fortran a C/C++ (defaultně Fortran). Jednotlivé verze se liší pouze v názvech některých volaných metod. MKL DSS využívá přímého výpočtu řídkých matic.

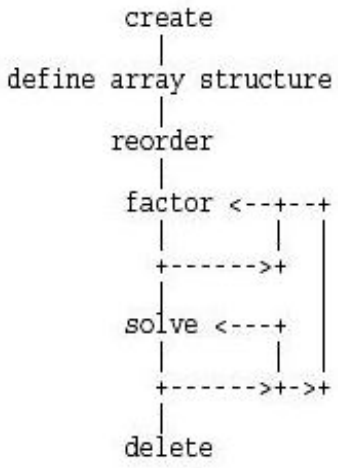
Implementační postup řešení v základě můžeme rozdělit do několika po sobě jdoucích kroků (Obrázek 4.1), kterým odpovídají volané metody v následujícím pořadí [1]

- *dss_create*
- *dss_define_structure*
- *dss_reorder*
- *dss_factor*
- *dss_solve*
- *dss_delete*

Pomocí metody *dss_create* si solver sám vytvoří vnitřní strukturu pro uložení a řešení matice. Do této struktury jsou dalšími metodami předány odkazy na konkrétní data, jako rozměry matice, informace o nenulových hodnotách matice, jejich pozici, hodnoty pravé strany atd. (4.2.2), které solver potřebuje. MKL DSS si převezme ukazatele na data, upraví je vhodně pro svou vnitřní reprezentaci, uloží je, následně na jejich základě provede výpočet a vrátí uživateli už jen výsledné řešení matice v poli typu double.

Jednu zkonstruovanou matici lze řešit s několika různými pravými stranami. Buď je možné se vracet k metodě *dss_factor* (Obrázek 4.1) a potom zavolat všechny následující metody, nebo lze volat několikrát za sebou *dss_solve*, pochopitelně vždy s prázdným polem pro výsledek.

MKL DSS si nekopíruje do paměti při výpočtu pole dodané uživatelem, ale pouze si na ně zachovává ukazatele. Tím pádem uživatel po předání solveru už nemůže daná pole měnit. Existuje i možnost volit mezi in-core, kdy se všechna data nachází v paměti, nebo out-of-core (OOC) verzemi MKL DSS.



Obrázek 4.1: Grafické znázornění logiky práce MKL DSS solveru. Převzato z [1].

K dispozici je freewarová třicetidenní licence programu, kde je uživateli zaslán na email odkaz se souborem ke stažení. Po instalaci je ještě třeba například ve vývojovém prostředí Microsoft Visual Studio nastavit v Project/Project Properties, že budeme s touto knihovnou v projektu pracovat a zvolíme způsob zpracování např. paralelní nebo sekvenční.

4.2.2 Příklad užití MKL DSS

Předpokládejme, že máme předem připravené a naplněné následující proměnné [2]:

Název proměnné	**Popis**
double∗ vysledek	prázdné vynulované pole o velikosti předpokládaného výsledku
int maticeSize	velikost jednoho rozměru námi konstruované matice (proměnné radky/sloupce hodnotou odpovídají maticeSize, rozlišujeme je jen pro větší přehlednost, jakou informaci metoda vyžaduje)
MKL_INT∗ prvniNaRadku	obsahuje indexy nenulových prvků, které jsou v matici první na řádku, velikost pole se rovná počtu nenulových prvků plus jedna (závěrečný dummy, který metoda vyžaduje)
MKL_INT∗ indexyX	obsahuje informaci o umístění nenulových prvků ve sloupcích matice, velikost pole se rovná počtu nenulových prvků
MKL_INT nenul	počet nenulových prvků matice
double∗ hodnoty	obsahuje nenulové hodnoty matice, velikost pole se rovná počtu nenulových prvků
_DOUBLE_PRECISION_t∗ pravaS	obsahuje hodnoty pravé strany maticové rovnice, velikost pole se rovna hodnotě *maticeSize*

Pak v samotném kódu nejprve vytvoříme *handler* pro solver, který bude po celou dobu odkazovat na aktuální podobu solveru. Dále nastavíme počáteční parametry, jako jsou možnosti, informace o symetričnosti a typu matice a počtu plánovaných pravých stran, pro které se bude daná matice řešit. Pak už se pouze volají metody s odpovídajícími argumenty:

```
_MKL_DSS_HANDLE_t handle;
_CHARACTER_t statIn[] = "determinant", *uplo;
_DOUBLE_PRECISION_t statOut[5], eps = 1e-6;
MKL_INT opt=MKL_DSS_DEFAULTS, opt1;
MKL_INT sym=MKL_DSS_NON_SYMMETRIC;
 MKL_INT type = MKL_DSS_INDEFINITE;
MKL_INT nRhs=1;

dss_create(handle, opt);
dss_define_structure(handle, sym, prvniNaRadku, radky, sloupce, indexyX, nenul);
dss_reorder(handle, opt, 0);
dss_factor_real(handle, type, hodnoty);

uplo = "non-transposed";
opt1 = MK_DSS_DEFAULTS;
opt |= opt1;

dss_solve_real(handle, opt, pravaS, nRhs, vysledek);
opt &= ~opt1;
dss_delete(handle, opt);
```

4.3 Grafické rozhraní Imgui

Pro aplikaci bylo využito dodané jednoduché grafické rozhraní Imgui [3]. Toto rozhraní poskytuje jednoduché nástroje pro tvorbu oken, popisků, rámců, tlačítek, posuvníků a jiných základních prvků potřebných pro funkční grafické prostředí. Rozhraní je psané v jazyce C++ a využívá freewarový multiplatformní Framework Qt. Autorem je Ondřej Jamriška.

K dispozici jsou buď samostatné zdrojové kódy, anebo již zkompilované celé rozhraní, které stačí jen přidat do projektu.

Obrázek 4.2: Ukázka grafického rozhraní Imgui

Kapitola 5

Implementace

5.1 Návrh uživatelského rozhraní

Implementovaný editor bude realizován v jednoduchém grafickém prostředí Imgui a bude mít dva základní kreslící módy. První bude čisté gradientní kreslení s možností nastavení hodnoty kresleného gradientu. V tomto módu se využije idea čistého gradientního štětce s funkcí mísení gradientů overblending.

Druhý mód bude realizovat bezešvé klonování z dvou zdrojových obrazů. Bude podporovat nástroj výběru a vložení oblasti. Pro tento režim bude možné si zvolit, jakým způsobem bude gradientní pole ovlivněno. Dostupné možnosti ovlivnění budou: overblending, additive blending a maximální gradient popsané v sekci 3.2.

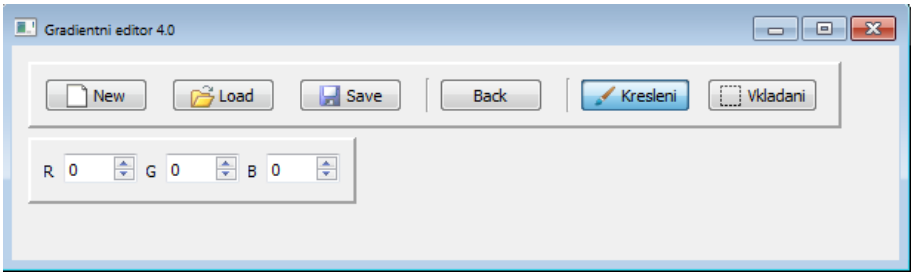
Krom těchto dvou pracovních módů bude editor umět načítat obrazy, ukládat je, vytvářet nové s možností volby velikosti obrazu. Vykreslování se bude snažit přiblížit práci v reálném čase. Nové intenzity se budou vypočítávat po pixelech a po jednotlivých kanálech barevného schématu RGB.

Na implementovaném editoru budou testovány například následující skutečnosti: čas zpětné vazby uživateli, maximální možnost velikosti obrazu, při níž bude program schopen v přijatelném čase poskytovat zpětnou vazbu uživateli, nebo celková míra splnění očekávání.

5.2 Grafická podoba editoru a podpůrné funkce

Editor obsahuje hlavní panel se základními funkcemi, jako načítání, ukládání, tvorba nového obrazu a volba nástroje, se kterým chce uživatel pracovat.

V případě, že uživatel zvolí jeden z nástrojů, buď štětec, nebo vkládání, zobrazí se mu vždy přídavný panel s příslušnými dalšími nástroji a nastaveními. V případě štětce je to nastavení velikosti gradientu na štětci pro jednotlivé barevné kanály (Obrázek 5.1) a v případě nástroje vkládání je to možnost načtení druhého zdrojového obrazu, volba modifikace gradientů, aktivace nástroje výběru a nastavení parametrů nástroje pro označení vkládané oblasti.



Obrázek 5.1: Ukázka hlavního panelu a panelu pro kreslení.

5.2.1 Načítání, ukládání a nový obraz

Data obrazu jsou načtena pomocí *stb_image*, které načítá obrázek ve schématu RGBA, tedy čtyřsložkové reprezentaci v poli. Čtvrtou složkou je kanál průhlednosti alfa. Důvod pro zvolení RGBA módu je ten, že metoda pro vykreslení obrazu do grafického prostředí (GUI) tento formát reprezentace obrazu vyžaduje. Adresace jednotlivých pixelů v poli pak vypadá

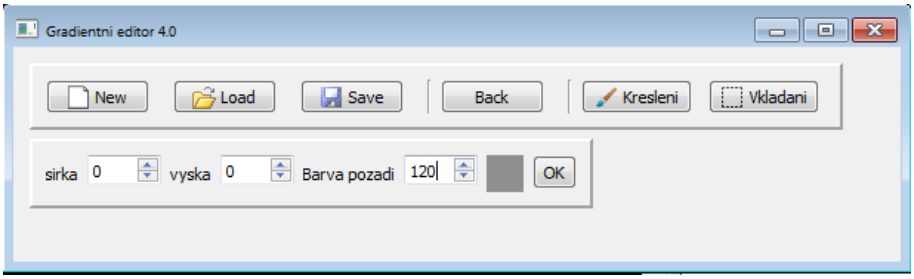
$$obrazek[i \cdot w \cdot 4 + j \cdot 4 + posun],$$

kde *posun* reprezentuje hodnoty určitého barevného kanálu pro daný pixel. Nula odpovídá kanálu modré, jednička zelené a dvojka červené barvy. Poslední čtvrtá hodnota reprezentuje kanál průhlednosti, tedy alfu. Výška obrazu zůstává stejná a na šířce se projeví to, kolika hodnotami je jeden pixel definován, v tomto případě bude šířka $w \cdot 4$ a celková velikost pole *obrazek* bude $h \cdot w \cdot 4$.

Obrázek se ukládá pomocí jednoduchého skriptu v barevném modelu RGB a formátu TGA. Při výpočtu a ukládání se kanál pro průhlednost zanedbává a v potaz se berou pouze hodnoty reprezentující RGB kanály. V praxi to znamená, že se při výpočtu hodnoty alfy nemění, a při ukládání vybereme cyklem z pole pouze hodnoty kanálů RBG, které uložíme do menšího pole a předáme ho metodě pro uložení.

Při načítání, zobrazování a ukládání obrazu se musí věnovat zvýšená pozornost pořadí, v jakém jsou za sebou v proměnné uloženy jednotlivé hodnoty pro barevné kanály pro jeden pixel. Pořadí, v jakém jsou z obrazu načteny, totiž neodpovídá tomu, jaké vyžaduje metoda $pixmapBit(w, h, obrazek)$, která zajišťuje zobrazení do GUI. Metoda pro uložení dokonce vyžaduje úplně opačné přeskládání pixelů. Je tedy nezbytné po načtení a před uložením obrazu pomocí cyklů a pomocného pole pořadí hodnot přeskládat.

V případě nového obrazu je pozadí jednobarevné a gradienty jsou nastaveny na nulu. Pole lze nastavit nulové přímo ručně, protože víme z obecné definice gradientu (Sekce 2.1), že u jednobarevného obrazu žádná změna v gradientním poli nenastane. Velikost nového obrazu lze nastavit v rozmezí nula až tisíc pixelů v obou rozměrech a defaultní barvou pozadí je odstín šedé bez jakékoliv průhlednosti. Odstín si uživatel může zvolit při vytváření obrazu tradičně v rozmezí hodnot 0 - 255 (Obrázek 5.2).



Obrázek 5.2: Ukázka možnosti nastavení parametrů nového obrazu.

5.3 Gradientní kreslení a vkládání výřezu

5.3.1 Gradientní štětec

Při volbě kreslení štětcem si na liště uživatel zvolí jednotlivé hodnoty gradientu, které chce, aby jeho štětec v jednotlivých kanálech emitoval do pozadí. Mód štětce je provázán logickými podmínkami s módem vkládání výřezu tak, aby nemohly být aktivní oba současně.

Pro práci v tomto módu jsou vytvořena dvě pomocná pole $poleGradX$ a $poleGradY$ o velikosti $h \cdot w \cdot 4$, ze kterých se vypočítává pravá strana maticové rovnosti dle následujícího vzorce

$$g_{(x,y)} = g_{x(x,y)} - g_{x(x-1,y)} + g_{y(x,y)} + g_{y(x,y-1)} \tag{5.1}$$

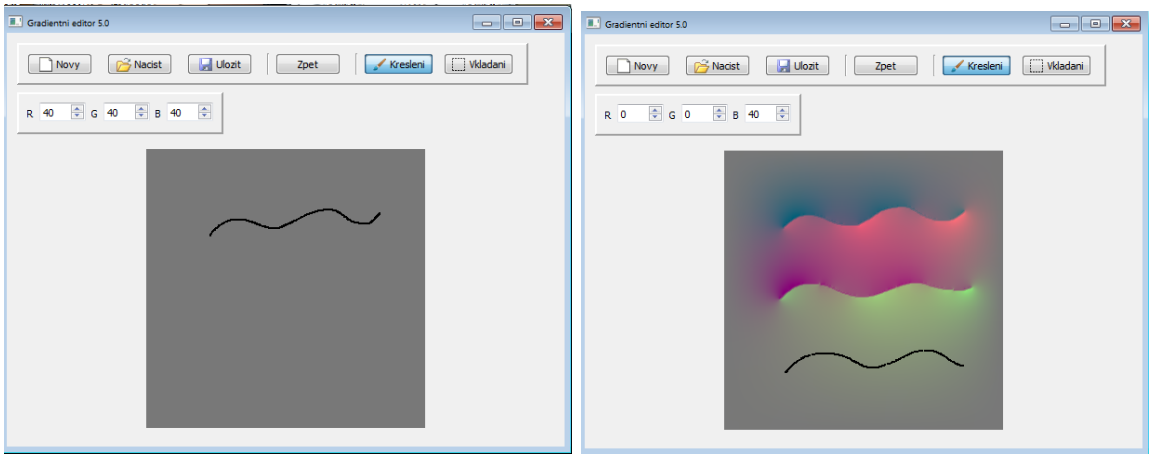
s tím, že pro okrajové pixely obrazu platí Dirichletovy počáteční podmínky (2.7). Jiné počáteční podmínky než tyto okrajové Dirichletovy se na obrázek aplikovat nebudou. Výpočet pravé strany probíhá od levého horního rohu po jednotlivých pixelech, tedy zleva doprava a odshora dolů.

Základní logika tohoto nástroje pak vypadá tak, že při tahu myší po ploše obrazu se zachytávají pomocí fukncí GUI $mouseX()$ a $mouseY()$ souřadnice myši vzhledem k ploše obrazu a na tyto souřadnice se s odpovídajícím posunem pro jednotlivé kanály uloží do proměnných $poleGradX$ a $poleGradY$ příslušná hodnota gradientu, která se získá rozkladem z nastaveného gradientu od uživatele.

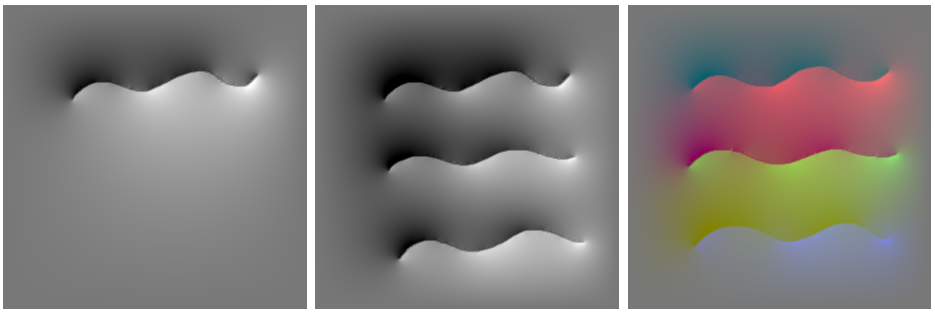
Po dokončení tahu, tedy při uvolnění stisku levého tlačítka myši, se zavolají dvě podpůrné metody, z nichž jedna vypočítá hodnotu pravé strany (5.1) a druhá pro usnadnění přístupu k hodnotám aktuálně počítaného barevného kanálu naplní globální proměnnou *kanal* ukazateli na hodnoty pouze jednoho kanálu. Pracujeme tedy s polem o čtvrtinové velikosti.

Teprve pak se zavolá přímo solver, který si dopřipraví zbytek proměnných (4.2.2) a provede výpočet. Před zavoláním *dss_delete* uloží spočtené hodnoty pomocí jednoho cyklu na příslušná místa v proměnné *obrazek*[].

DSS Solver se volá pouze jednou a připravená matice se řeší třikrát pomocí *dss_solve*, pokaždé s jinou pravou stranou, která odpovídá hodnotám pro jeden barevný kanál. Výpočet



Obrázek 5.3: Kreslení gradientem v odstínech šedé a barvě. Tah.



Obrázek 5.4: Výsledeky po výpočtech obrázků 5.3.

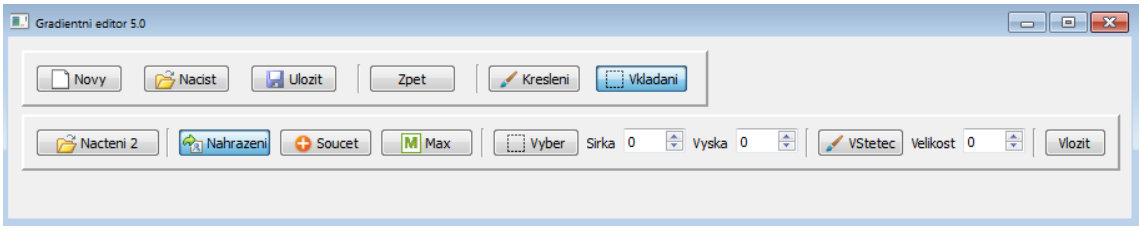
se netýká kanálu alfa, se kterým se vůbec nepočítá, a tak zůstává celou dobu nastaven na defaultní hodnotu 255, tedy plnou neprůhlednost.

V implementaci bylo nutné ošetřit také přesah při stisknutém tlačítku mimo pole obrazu. Z toho důvodu, že při přesažení se gradienty dále zaznamenávaly, a to na opačné straně obrazu, a kreslili jsme tak tam, kde jsme vůbec nechtěli.

Stejně tak je potřeba kontrolovat výsledné hodnoty od solveru předtím, než se uloží do proměnné. To pro případ, že se po výpočtu pohybují mimo rozmezí 0 – 255 (od černé po bílou), pak je potřeba tyto hodnoty před uložením upravit na krajní meze. Jinak by došlo k přetečení informace do okolních pixelů, a tedy ke změně obrazu. Platí pro všechna ukládání po dopočítávaných změnách obrazu.

5.3.2 Vkládání výřezu

Po stisknutí tlačítka výběru na hlavním panelu se zobrazí přídavný panel s možnostmi nutnými pro tento mód.



Obrázek 5.5: Přídavný panel pro mód vkládání výřezu.

Tlačítka pro volbu možnosti mísení gradientů jsou, obdobně jako je tomu v případě dvojice tlačítek štětec - vkládání, ošetřena pomocí logických podmínek tak, aby vždy mohlo být v daném okamžiku stisknuté pouze jedno.

Defaultně je nastavena možnost přímého nahrazení gradientů, která se i automaticky nastaví v případě, že se nezmáčkne nové tlačítko, ale pouze zruší označení momentálně zvoleného. Tlačítka jsou realizovaná pomocí ToggleButton a díky tomu zůstávají po dobu, kdy s nimi svázaná proměnná se rovná true, zmáčknutá. Lze tedy na první pohled identifikovat, který mód je zrovna aktivní, a uživatel vždy ví, s čím zrovna pracuje.

Při načtení druhého zdrojového obrazu se druhý obraz zobrazí napravo od prvního obrazu (Obrázek 5.6). V tomto zdroji je možné v módu výběru označit část obrazu, kterou budeme chtít kopírovat. Oblast o stejné velikosti pak můžeme umístit v levém obraze na místo, kam se dané gradienty přenesou. Systémově je ošetřeno, že vybírat oblast lze pouze v pravém obraze a vkládat lze pouze do levého obrazu.

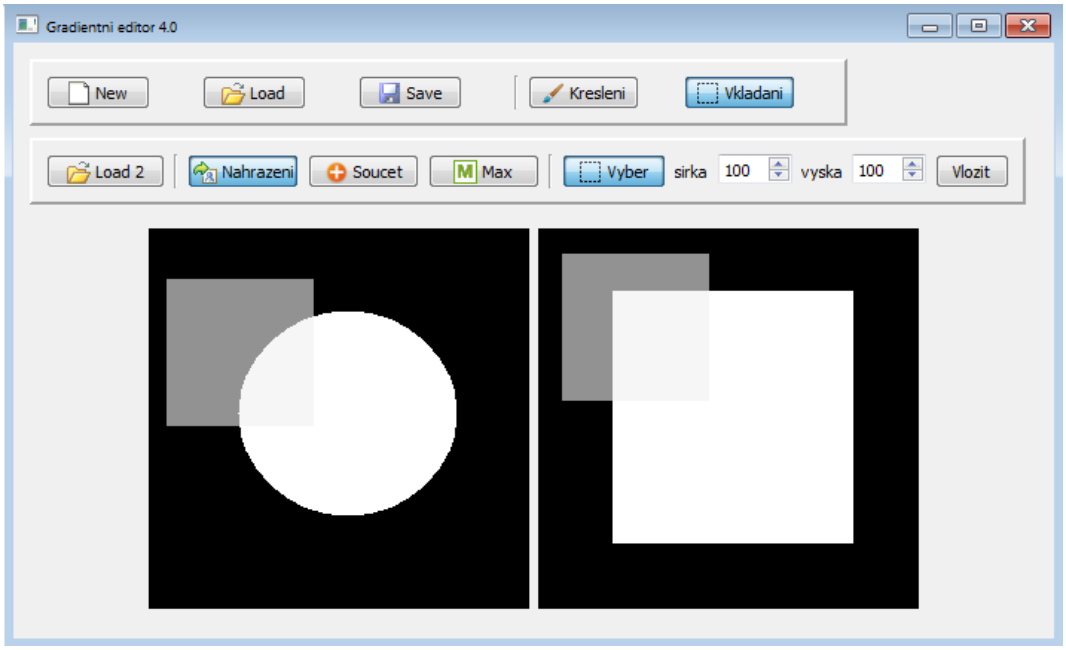
Vkládaná oblast je detekovatelná s využitím jinak nevyužitého kanálu pro průhlednost. Pokud se pixel nachází v oblasti výběru, pak je jeho hodnota alfa kanálu místo defaultních 255 nastavena na hodnotu 100. Zaprvé tím můžeme v dalších krocích jasně identifikovat vybrané pixely a zadruhé je uživateli okamžitě poskytnuta zpětná vazba o tom, které pixely vlastně vybírá. Pokud pixel opustí vybranou oblast, je mu opět nastavena průhlednost na hodnotu 255.

V případě pevně daného obdélníkového výběru je to realizováno dvěma cykly, které projíždějí obraz a zjišťují, zda jednotlivé pixely patří oblasti výběru pomocí logických podmínek:

$$i > (y - \frac{hV}{2}) \,\&\&\, i < (y + \frac{hV}{2}) \,\&\&\, j < (x + \frac{wV}{2}) \,\&\&\, j > (x - \frac{wV}{2}), \tag{5.2}$$

kde i a j jsou souřadnice aktuálně hodnoceného pixelu, x a y jsou souřadnice myši a hV a wV jsou rozměry obdélníkového výřezu zadané uživatelem. Aktuální souřadnice myši se zachytávají stejným způsobem jako v případě gradientního kreslení. Kurzor bude vždy ve středu oblasti (Obrázek 5.6).

V případě druhého nástroje výběru, štětce, se přímo přepisují alfa hodnoty pixelů, které uživatel přejel myší, a také těch, které spadají do okolí, jehož velikost uživatel nastavuje vedle tlačítka pro štětec. Kurzor je opět situovaný doprostřed oblasti (Obrázek 5.7).



Obrázek 5.6: Ukázka označování oblastí při vkládání pevného výřezu.

V rámci výběru oblasti štětcem se zachovají v paměti souřadnice levého horního pixelu. Vzhledem k tomu se pak určují pozice dalších, pomocí rozdílu souřadnic x a y aktuálního pixelu a levého horního pixelu. Tento rozdíl je pak přičten k souřadnicím myši, které zachytíme v levém obraze a díky tomu můžeme identifikovat pixely patřící vybrané oblasti i v druhém obraze.

Obrázek 5.7: Ukázka označování oblastí pomocí štětce.

Aby nedocházelo k náhodným označováním oblastí, je obrázek citlivý na zachytávání souřadnic myši až po stisknutí tlačítka výběr/štětec na přídavném panelu. V případě opuštění obou nástrojů se všechny hodnoty v kanálu alfa cyklem opět nastaví na hodnotu 255. Stejně tak tomu bude při celkovém opuštění módu vkládání nebo dokončení výpočtu.

Příprava pravé strany pro Solver vypadá tak, že spočteme laplaciany pro levý obraz (2.10), přičemž se bere ohled na hranice vkládané oblasti, pro které platí Dirichletovy vstupní podmínky (2.7) a jsou definovány tak, že se v jejich okolí v buď vodorovném, nebo svislém směru nachází pixel s hodnotou průhlednosti 100.

Následně se detekují vyznačené oblasti v obou obrazech a opět pomocí relativního mapování vzhledem k levým horním rohům obou oblastí jsou upraveny hodnoty laplaců v levém obraze těmi z obrazu pravého, které se dočítají pouze pro přenášené pixely.

To, jakým způsobem jsou upraveny, záleží na zvoleném módu. V případě přímého nahrazení se hodnoty plně přepíší (3.4), v případě součtu je nová hodnota součtem nové a původní (3.2) a v případě maxima se vybere větší z absolutních hodnot obou čísel (3.3).

Připrava samotné matice, řešení DSS Solverem a ukládání hodnot pak proběhne stejně jako v případě gradientního kreslení popsaného v sekci 5.3.1

5.3.3 Další funkčnosti

U většiny základních funkcí, které jsou běžné v intenzitních editorech, není realizace v gradientní oblasti tak triviální, jako je tomu v případě intenzitních editorů.

Jednou z takových věcí je implementace zpětné vazby tahu štětcem, aby bylo zřetelné, kudy vede trajektorie tahu, který právě uživatel provádí (Obrázek 5.3). V intenzitních editorech se rovnou hodnota barvy ukládá do obrazu a změna je ihned viditelná a rovná se křivce tahu. Obecně v gradientním editoru by tento přístup nebyl možný, protože tahem štětce bychom mohli přepsat hodnotu počáteční podmínky, a následný výpočet nových intenzit by tak nebyl správný.

V našem případě můžeme tuto skutečnost zanedbat, neboť víme, že počáteční podmínky budou pouze v okrajových pixelech obrazu a že hodnoty gradientů jsou ukládány v samostatných polích ještě před tahem, tudíž změna barvy trajektorie tahu uvnitř obrazu neovlivní výpočet a dostaneme i tak správné výsledky.

Případně lze využít kanálu alfa a měnit jeho hodnoty, jako toho využijeme u nástroje pro vkládání. Tady ovšem může být nevýhodou špatná rozpoznatelnost, protože jak pozadí, tak okno editoru má šedou barvu.

V editoru je implementované i jednoduché tlačítko Back. To nám umožňuje vrátit právě jednu poslední provedenou akci. Realizováno je pomocí pomocných proměnných. V módu kreslení se vytváří kopie proměnných *poleGradX* a *poleGradY* a a to vždy, pokud se dokončí tah a levé tlačítko myši přestane být stiknuté. V módu vkládání výřezu se pak do pomocné proměnné ukládají konkrétní hodnoty intenzit z proměnné *obrazek* a to před voláním solveru.

V případě stisknutí tlačítka Back se do příslušných proměnných dle aktuálního módu práce nahrají uchované hodnoty z pomocných proměnných a v případě kreslení dojde k přepočítání hodnot intenzit dle obnovených hodnot gradientního pole.

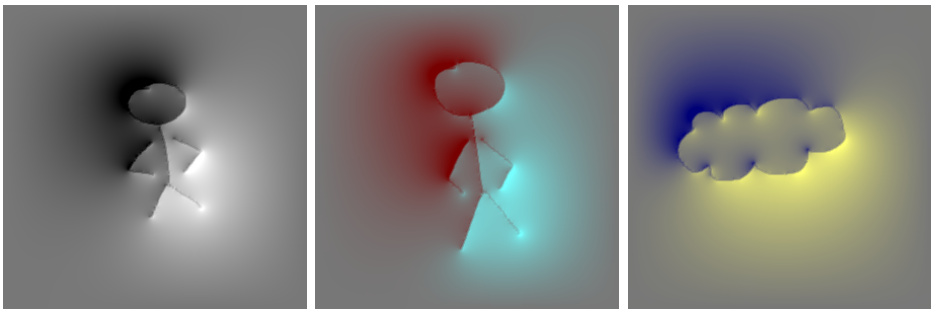
5.4 Výsledky a pozorování

Pozorování a zmíněné časové údaje byly získany na notebooku s následujícími parametry:

- Model: ASUS Notebook UL50VT/UL50Vg
- CPU: Genuine Intel(R) CPU U7300, 1.3 GHz
- OS: Windows 7 Professional, 64 bit
- RAM: 4,00 GB
- Grafická karta: Nvidia Geforce G210M

V módu gradientního kreslení můžeme prezentovat následující výsledky: V případě, že všechny gradienty pro jednotlivé kanály jsou nastaveny na stejnou hodnotu, pak dochází k kreslení přechodů v odstínech šedé barvy (Obrázek 5.8).

Pokud se hodnoty na štětci pro jednotlivé barevné kanály liší, pak se místo v odstínech šedé kreslí barevné přechody odpovídající nastaveným hodnotám pro RGB kanály (Obrázek 5.8).



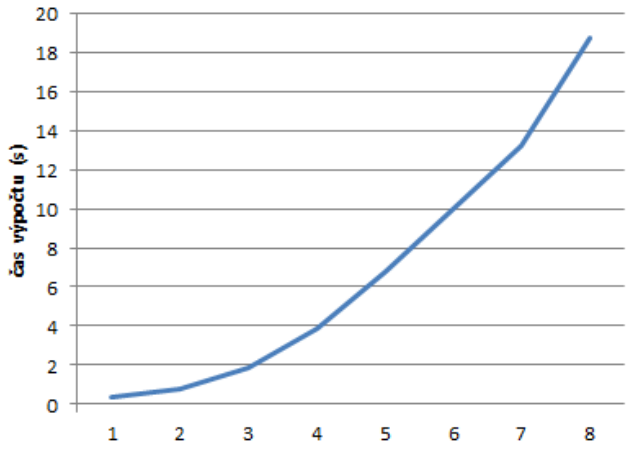
Obrázek 5.8: Ukázka kreslení v odstínech šedé barvy a kanálů RGB.

V módu vkládání výřezu potom můžeme prezentovat tyto výsledky pro jednotlivé možnosti vkládání: přímé nahrazení (Obrázky 5.10, 5.11), součet gradientů (Obrázky 5.12, 5.13, 5.14) a maximální gradient (Obrázky 5.15). Pro lepší demostraci výsledků byly převzaty obrázky z [7] a [9].

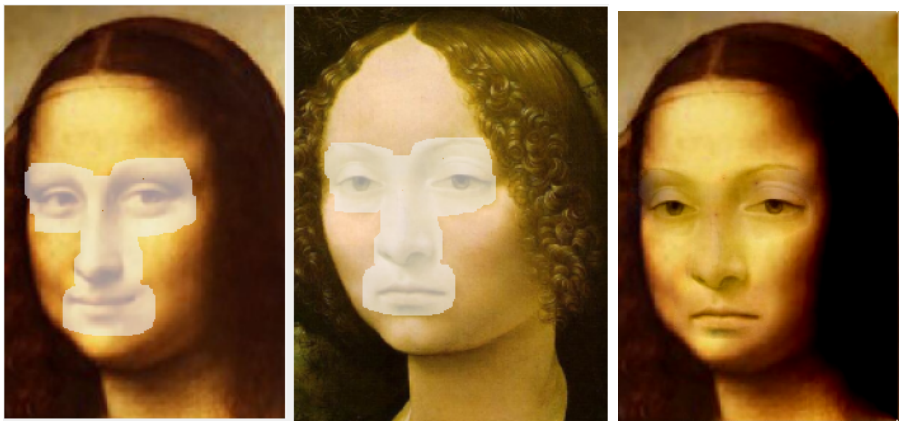
Základní podmínkou použitelnosti gradientních úprav v praxi je co nejkratší zpětná vazba pro uživatele. V ideálním případě by tato vazba měla být v reálném čase, ale jelikož se nejedná o triviální změnu intenzity, která se zapisuje přímo do obrazu a je ihned viditelná, může se zpětná reakce při gradientních úpravách protáhnout do řádů sekund, ale i minut. V našem případě se nesoustředíme na implementaci metod, které pracují v reálném čase, ale cílem je vytvoření uživatelského prostředí gradientního editoru.

V našem editoru se přibližně do velikosti obrazů 300 x 300 pixelů doba výpočtu DSS solveru pohybuje okolo 1,5 - 2 sekud pro celkový výpočet všech tří kanálů. Jako časově nejnáročnější byla pozorováním určena volání funkce *dss_reorder*. Od velikosti obrazu přibližně 500 x 500 pixelů a více se výpočet pro každý kanál poměrně znatelně prodlužuje. Velikost prodloužení výpočtu záleží mimo jiné i na celkovém vytížení počítače. Průměrné hodnoty časů výpočtů znázorňuje jednoduchý graf 5.9.

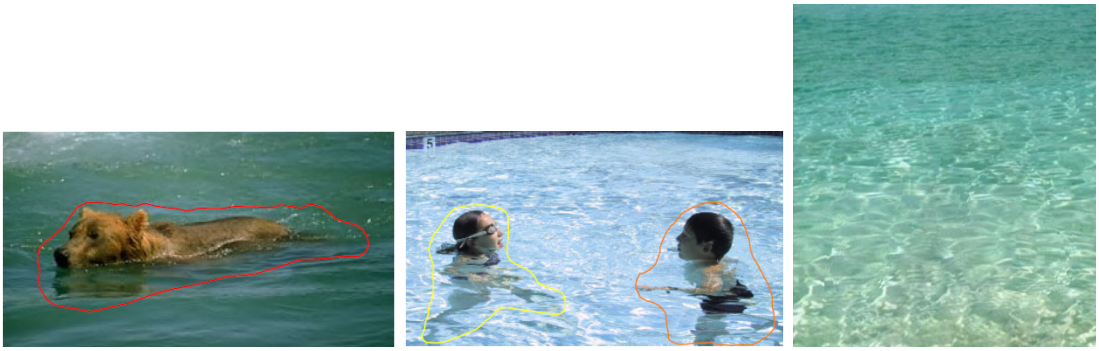
Bohužel výpočet větší velikosti obrazů se protahují přes 10 sekund pro celý obraz, a v případě velikosti 900 x 900 pixelů a více dokonce dochází k přerušení výpočtu v důsledku nedostatku výpočetní paměti. Způsobeno je to mimo jiné zvolenou přímou metodou pro řešení soustavy rovnic, jejíž nároky na paměť s velikostí počítané oblasti rostou.



Obrázek 5.9: Graf nárůstu času výpočtu s ohledem na velikost obrazu. Osa x je v měřítku 1=100 pixelů a představuje rozměr jedné strany čtvercového vstupního obrazu.



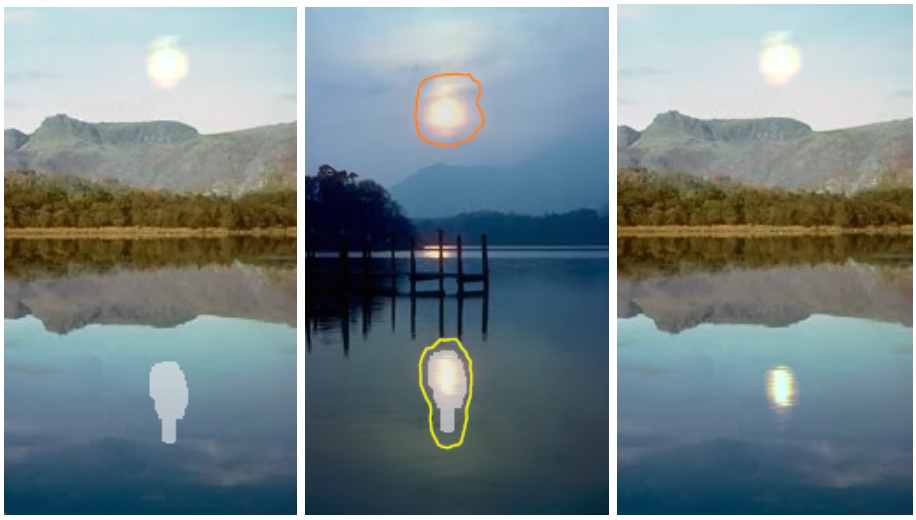
Obrázek 5.10: Vkládání přímým nahrazením gradientů. Zdrojové obrázky převzaty [7].



Obrázek 5.11: Zdrojové obrázky pro vkládání přímým nahrazením. Převzato z [7].

Obrázek 5.12: Výsledky vkládání přímým nahrazením gradientů 5.11. Vložen medvěd a lidé.

Obrázek 5.13: Vkládání pomocí sčítání gradientu. Zdrojové obrázky převzaty [7].



Obrázek 5.14: Vkládání pomocí sčítání gradientu. Vloženo slunce a jeho odraz ve vodě. Zdrojové obrázky převzaty [7].

Obrázek 5.15: Vkládání pomocí maximálního gradientu. Zdrojové obrázky převzaty [7].

Kapitola 6

Závěr

Cílem práce bylo seznámit se s metodou úpravy obrazu v gradientní oblasti a implementace jednoduchého obrázkového editoru na základě získaných poznatků.

Seznámili jsme se s pojmy gradient a gradientní pole, jejich definicemi pro 2D obrazy a s postupem integrace obrazu z gradientního pole pomocí diskretizované Poissonovy parciálně derivační rovnice s počátečními podmínkami. Tuto rovnici lze převést na velkou soustavu lineárních rovnic, která je v našem editoru řešena za pomoci knihovny MKL Direct Sparse Solver.

Dále jsme prostudovali metody využití gradientních nástrojů v praxi, např. v gradietních štětcích, bezešvém klonování, lokálních úpravách osvětlení a textur atd.

Pro náš obrázkový editor byly implementovány funkce načtení obrazu pomocí knihovny *stb_image* a barevném modelu RGBA, ukládání obrazu ve formátu TGA, jednoduché tlačítko pro navrácení poslední provedé akce a dva módy pro práci s obrazem: gradientní kreslení a vkládání výřezu.

V módu gradientního kreslení je realizován gradientní štětec, který nahrazuje hodnoty gradientů na pozadí v jednotlivých barevných kanálech odpovídajícími novými, uživatelem nastavenými, hodnotami.

Mód vkládání výřezu pak nabízí implementované tři různé možnosti změnění gradientního pole, a to nahrazením, součtem a zachováním maximálního gradientu. Tento mód podporuje dva různé nástroje výběru, buď obdélníkový, kde se označená oblast detekuje do všech čtyř směrů v závislosti na nastavených rozměrech oblasti, nebo volný štětec, kdy uživatel štětcem vybarví oblast, kterou chce vkládat.

Můžeme konstatovat, že pro obrázky o rozměrech do 300 x 300 pixelů je čas výpočtu editoru přijatený přibližně okolo 2 sekund pro všechny tři barevné kanály. Pro větší obrázky se čas výpočtu výrazně zvyšuje a od hranice 900 x 900 pixelů dochází k problému nedostatku výpočetní paměti a výpočet se přerušuje. Co se obrazových výsledků týče, ověřili jsme, že náš obrázkový editor dává ekvivalentní výsledky jako jsou uvedeny v [7].

V budoucnu by se tento editor dal lépe uvést do praxe, pokud by se podařilo ještě více zkrátit dobu zpětné vazby, ideálně do reálného času, a to i pro větší obrázky. Zároveň se jako další možnost vylepšení nabízí řešení otázky velkých požadavků na výpočetní paměť, což by umožnilo práci i na větších obrázcích.

Literatura

[1] Math Kerner Library Direct Sparse Solver – Interface routines, 2013.

[2] *Sparse Solver Routines Manual*, 2007. Dostupné z: <`http://www.ncsa.illinois.edu/UserInfo/Resources/Hardware/SGIAltix/Doc/mkl10/doc/ssr_suppl.pdf`>.

[3] IMGUI source code, 2013. Dostupné z: <`https://code.google.com/p/imgui/`>.

[4] Simple Components for Visual: stb_image.h, 2013. Dostupné z: <`http://www-usr.inf.ufsm.br/~pozzer/scv/stb__image_8h_source.html`>.

[5] ADELSON, E. H. *Lightness Perception and Lightness Illusions*, s. 339–351. MA: MIT Press - Cambridge, second edition, 2000.

[6] MCCANN, J. – POLLARD, N. S. Real-time gradient-domain painting. *ACM Transactions on Graphics*. 2008, 27, 3, s. 93.

[7] PÉREZ, P. – GANGNET, M. – BLAKE, A. Poisson image editing. *ACM Transactions on Graphics*. 2003, 22, 3, s. 313–318.

[8] SEIDELOVÁ, L. Porovnání metod pro řešení velkých řídkých soustav lineárních rovnic. Bakalářská práce, České vysoké učení technické v Praze, 2013.

[9] SÝKORA, D. Digital Image Processing – Image Editing. Přednáškové slidy.

Kapitola 7

Seznam použitých zkratek

HVS Human visual system

RGB Barevný model red - blue - green

RGBA Barevný model red - blue - green s alfou (průhledností)

TGA Truevision TGA/Targa

PNG Portable Network Graphics

2D Two-dimensional space

MKL Math Kernel Library

DSS Direct Sparse Solver Interface Routines

GUI Graphical user interface

Kapitola 8

Obsah přiloženého DVD

```
DVD
 |-- Kod                        - Zdrojové kódy
 |   |--projekt                  - Kompletní projekt ve Visual studiu
 |   |--bakalarka                - Zkompilovaný program
 |-- Obrazky
 |   |--obrazky                   - Vstupní obrázky
 |   |--obrazkyTGA                - Výstupní obrázky ve formátu TGA
 |   |--obrazkyPNG                - Výstupní obrázky ve formátu PNG
 |   \--graf.png                  - Graf závislosti času výpočtu na velikosti obrazu
 |-- Text
 |   |--latex                     - Zdrojové soubory textu práce
 |   \--bakalarka.pdf             - Text práce v PDF
 |   \-manual.pdf                 - Stručný manuál k programu
 \-- readme.txt                 - Informace o obsahu DVD
```