VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

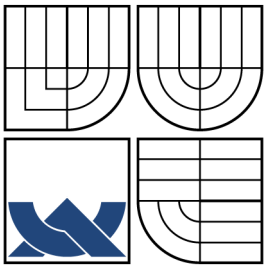
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

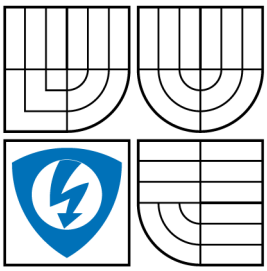
METODY PRO DOPLŇOVÁNÍ CHYBĚJÍCÍCH DAT V OBRAZU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE RADEK VODIČKA
AUTHOR



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

METODY PRO DOPLŇOVÁNÍ CHYBĚJÍCÍCH DAT V OBRAZU

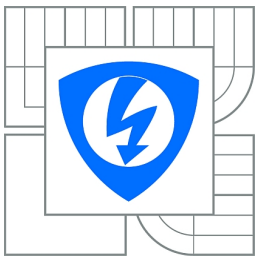
METHODS FOR FILLING MISSING DATA IN IMAGE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
RADEK VODIČKA

VEDOUCÍ PRÁCE
SUPERVISOR
Mgr. PAVEL RAJMIC, Ph.D.

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Radek Vodička	***ID:***	74530
Ročník:	3	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Metody pro doplňování chybějících dat v obrazu

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte z dostupné literatury různé přístupy k doplňování chybějících oblastí v obrazu (angl. tzv. inpainting) - výpadek přenosového kanálu, ostraněné logo, škrábanec či kaňka na fotografii. Shromážděte testovací obrázky a navrhněte postup testování vybraných metod a způsob porovnání. Objektivně a subjektivně porovnejte výsledky různých Vámi implementovaných metod.

DOPORUČENÁ LITERATURA:

[1] G. Aubert and P. Kornprobst, Mathematical Problems in Image Processing. Partial Differential Equations and the Calculus of Variations, Springer-Verlag, New York, 2002.
[2] Masnou, S. Disocclusion: a variational approach using level lines, Image Processing, IEEE Transactions on , vol.11, no.2, pp.68-76, Feb 2002, doi: 10.1109/83.982815
[3] C. Ballester et al. Filling-in by joint interpolation of vector fields and gray levels, IEEE Trans. Image Process., 10 (2001), pp. 1200–1211.
[4] M. Beltramio, G. Sapiro, V. Caselles, and B. Ballester, Image inpainting, in Proceedings of the 27th Annual ACM Conference on Computer Graphics, 2000, pp. 417–424.
[5] MISITI, Michel, et al. Wavelet Toolbox : For Use with MATLAB. The MathWorks Inc., 2000.

Termín zadání:	6.2.2012	***Termín odevzdání:***	31.5.2012

Vedoucí práce: Mgr. Pavel Rajmic, Ph.D.
Konzultanti bakalářské práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

ABSTRAKT

Cílem této práce je popsat základní principy metod inpaintingu. Blíže se zde seznámit s metodou postupné opravy poškozené oblasti z okrajů po bodech a pomocí úrovňových křivek.

V praktické části jsou předvedeny vlastní programy pro opravu dat v obrazu.

Výstupem práce je předvedení obrázků opravených vlastními programy a vyhodnocení, jaká metoda je v dané situaci pro daný typ obrázků nejvhodnější, dále použitelnost a funkčnost jednotlivých programů.

KLÍČOVÁ SLOVA

inpainting, Bertalmio, Masnou, poškození, obraz, vektory, úrovňové křivky, textura, program

ABSTRACT

The aim of this paper is to describe the basic principles of inpainting methods. We take a closer look with method of continual correction for damaged area from its margins in poins and by means of level lines.

The practical part includes showcase of the specific programs for repair of data in the image.

Result of the work is demonstration of images that are corrected by the specific programs and evaluation about which method is the most suitable one in the given situation for the individual type of images as well as usability and functionality of individual programs.

KEYWORDS

inpainting, Bertalmio, Masnou, damage, image, vectors, level lines, texture, program

VODIČKA, Radek *Metody pro doplňování chybějících dat v obrazu*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2012. 39 s. Vedoucí práce byl Mgr. Pavel Rajmic, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Metody pro doplňování chybějících dat v obrazu“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Chci poděkovat svému vedoucímu bakalářské práce *Mgr. Pavlu Rajmicovi, Ph.D.* za vedení, dohled a za poskytnutí informačních zdrojů potřebných k vypracování práce.

Dále bych chtěl poděkovat lidem z mého okolí za konstruktivní pomoc při řešení problémů a za morální podporu.

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Existuje mnoho způsobů, metod, jak zrekonstruovat poškozený obraz nebo jak z obrazu odstranit nevhodnou část. Opravy šumu, doostření nebo vyhlazení obrazu však nejsou za inpaintig označovány v pravém slova smyslu. U inpaintingu totiž nejsou důležitá data, která jsou v oblasti, na kterou je inpainting aplikován.

Slovem inpainting jsou označovány opravy artefaktů, které do obrazu nepatří a jeví se jako poškození např. vepsaný text (u mnoha digitálních fotoaparátů vložené datum), v prašném prostředí vyfoceny částečky prachu, které jsou viditelné jako světlé skvrny, trhliny na fotografiích, škrábance a jiné poškození způsobené na obraze. Ale patří sem i retušování obrazu například odstranění „červených očí“, nevhodného objektu, který by mohl narušit dojem z celku (televizní anténa na zámecké budově), popřípadě odstranění osoby z fotografie třeba kvůli politickým důvodům.

Hlavním cílem je vytvořit upravený obraz, na kterém nepůjde poznat, že na něm proběhly změny.

Dříve bylo opravování obrazů záležitostí restaurátorů, ale dnes, kdy se obrazy digitalizují, již funkce inpainting obsahuje celá řada grafických editorů.

Základní principy.

1. Dynamický obraz
    - Doplnění chybějících dat na snímcích ve filmu.
2. Statický obraz
    - Doplnění chybějící části na základě porovnávání a nalezením podobné oblasti v jiné části obrazu.
    - Dopočítání chybějící části obrazu z okrajů.

1 STANOVENÍ PROBLÉMU

Všechny metody vycházejí z předpokladu, že obraz je jako plocha I, která má rozměr $[M, N]$. Je tedy M sloupců a N řádků. Další věcí, která je společná všem metodám, je oblast, na niž bude inpainting aplikován. Tato část obrazu bude dále označována jako oblast Ω. To jaká oblast má být doplněna záleží na uživateli. Ten ji musí předem definovat. Tento výběr by měl co nejméně zasahovat do prostoru, který se má zachovat, aby případné zkreslení bylo co nejmenší. Nevhodně vybraná oblast Ω může mít zásadní vliv na výsledný obraz.

Ukázka nevhodně vybrané oblasti je vidět na následujícím příkladu (obr. 1.1). Pokud byla chyba označena čtvercovým výběrem (obr. 1.2), lze na obrázku po provedení inpaintingu jednoduše rozeznat, že na obrázku byla provedena oprava (obr. 1.4). Oproti tomu, při pohledu na obrázek, kde výběr kopíruje chybu (obr. 1.3), nejde téměř poznat, že s ním bylo pracováno (obr. 1.5).

Obr. 1.1: Zobrazení chyby na boku psa.[4]

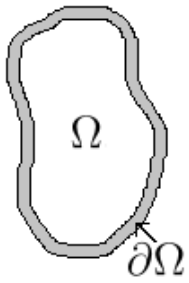
Obr. 1.2: Čtvercový výběr.[4]

Obr. 1.3: Přesný výběr.[4]

Obr. 1.4: Opravená chyba čtvercové oblasti.[4]

Obr. 1.5: Opravená chyba přesně kopírované oblasti.[4]

Dále je důležité okolí oblasti Ω, které je využíváno pro doplnění chybějící části. Toto okolí bude označováno jako $\partial\Omega$ (obr. 1.6). V některých metodách se využívá širší oblast, u některých metod je důležitý pouze přechod mezi známou oblastí $\partial\Omega$ a oblastí neznámou Ω.



Obr. 1.6: Oblasti.

1.1 Filmové snímky

K doplňování chybějících částí snímku ve filmu, je k dispozici mnoho informací. Využívá se předchozího a následujícího snímku. Pokud je však poškozeno několik snímků po sobě, musí se použít některá z metod popsaných níže a aplikovat na každý snímek zvlášť nebo nakombinovat oba způsoby. Tyto principy zde nebudou dále rozváděny.

2 METODY PRO DOPLŇOVÁNÍ CHYBĚJÍCÍCH DAT V OBRAZE

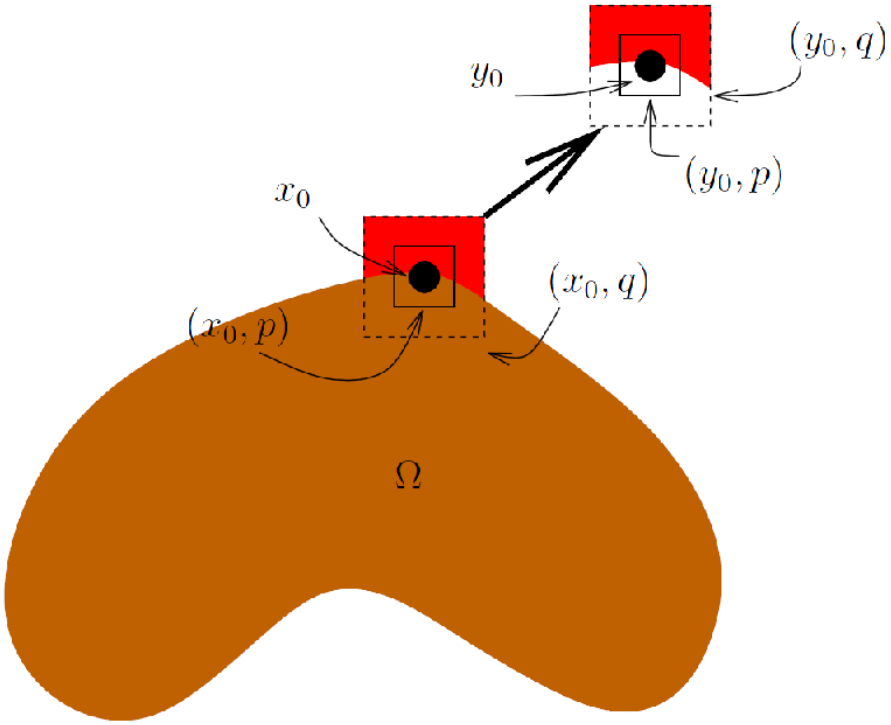
Následující metody využívají pro doplnění chybějící části různé algoritmy. V některých situacích je inpainting subjektivní a výběr oblasti Ω, na kterou bude metoda aplikována, závisí na uživateli. V jiných případech je tento výběr volen bez zásahu uživatele. Po označení oblasti Ω se vyplnění provede automaticky podle zvolené metody.

Pro jednoduchost budou algoritmy vysvětleny pouze na obrázcích ve „škále šedé“. Pokud by se jednalo o obrázek například v barevném modelu RGB, musel by se algoritmus aplikovat na každou barevnou složku zvlášť.

2.1 Implementace textur

Doplňování textur [3] je metoda, kterou lze udělat i manuálně. Princip metody spočívá v nalezení stejné nebo maximálně podobné oblasti Ω. Uživatel musí prohlédnout celý obrázek, aby nalezl požadovanou část, kterou by překryl oblast Ω. Pokud poškozené místo obsahuje více textur, musí být nalezeny všechny textury a navíc musí být doplněn přechod mezi těmito texturami.

Vyhledávání oblasti automaticky se řídí následujícím postupem. Je dán obrázek I, na kterém se nachází neznámá oblast Ω. Na okraji oblasti Ω je dán bod $x_0 \in \Omega$. Jsou zde definovány další dvě oblasti p a q, které mají střed v bodě x_0 tak, že oblast p obklopuje bod x_0 a oblast q obklopuje oblast p (obr. 2.1). Čili prohledává se celý obrázek I, dokud není nalezena oblast p na jiném místě obrázku. Po nalezení se přenese nová oblast q (obklopuje novou oblast p) se středem v bodě y_0. Pro dokonalejší navázání se používá rozšířená oblast q, aby na spojích nedošlo k viditelným přechodům. Souřadnice bodu $x \in p \cap \Omega$ je vektorově posunuta $I(x) = I(x + y_0 - x_0)$. Tímto způsobem je pokryta celá plocha Ω. Výsledky metody jsou závislé na zvolené velikosti p a q, kde $p > 0$ je lepší kopírovat malé oblasti než samotné body a $q > p$ jsou zajištěny plynulé přechody.



Obr. 2.1: Nalezení nové oblasti.[3]

2.2 Vyhodnocení metody implementace textur

Z teoretického hlediska je procházení obrázku a hledání stejné oblasti časově náročné, bez ohledu na to, jestli je činnost prováděna uživatelem manuálně nebo jestli je vykonávána automaticky pomocí programu.

Dobré výsledky aplikace metody implementace textur jsou předpokládány u obrázků, kde se objevují stejné vzory.

2.3 Bertalmiova metoda

Bertalmiova metoda [1] doplňuje chybějící část obrazu po bodech z okraje.

Je tedy definována oblast Ω, která bude vyplněna z okolí $\partial\Omega$. V tuto chvíli není důležité, co se v oblasti Ω nachází. Metoda se snaží intuitivně prodloužit isotopy (křivky spojující body stejné intenzity světla) z jedné strany $\partial\Omega$ na druhou, při zachování úhlu, pod kterým do oblasti Ω vstupují.

Základní kroky algoritmu jsou:

1. Celkový pohled na obraz určuje, jak vyplnit oblast Ω, aby změna nebyla poznat.
2. Struktura okolí $\partial\Omega$ pokračuje do oblasti Ω, přičemž vrstevnice mají zachován směr průchodem oblastí Ω
3. Uvnitř oblasti Ω jsou zachovány barvy z oblasti $\partial\Omega$.
4. Malé detaily jsou potlačeny a ztraceny.

Metoda opakuje krok 2. a 3. dokud se oblast Ω úplně nevytratí.

Základní matematické vztahy, které budou pro výpočty uplatňovány. Ať $I_0(i,j) : [0,M]\times[0,N] \rightarrow R$, kde $[0,M]\times[0,N] \subset N\times N$, je nespojitý 2D obraz ve škále šedé. U digitálního inpaintingu metoda vychází z bodu $I(i,j,n) : [0,M]\times[0,N]\times N \rightarrow R$, pokud $I(i,j,0) = I_0(i,j)$ a $\lim_{n\rightarrow\infty} I(i,j,n) = I_R(i,j)$, kde $I_R(i,j)$ je výstup z algoritmu. Hlavní rovnici algoritmu lze napsat jako:

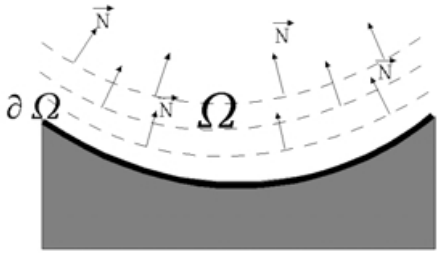
$$I^{n+1}(i,j) = I^n(i,j) + \Delta t I_t^n(i,j), \forall (i,j) \in \Omega, \tag{2.1}$$

kde n je aktuální krok algoritmu. Δt je míra zlepšení v n-tém kroku. (i,j) jsou souřadnice bodu. Ve vzorci (2.1) je tedy $I^{n+1}(i,j)$ vylepšená verze obrazu $I^n(i,j)$ o část $I_t^n(i,j)$. Je třeba určit, jak spočítat $I_t^n(i,j)$ v každém kroku. $L^n(i,j)$ je informace, která se přenáší z hranice $\partial\Omega$ dovnitř oblasti Ω ve směru $\vec{N}^n\ (i,j)$ (obr. 2.2).

Pak platí:

$$I_t^n(i,j) = \delta\vec{L}^n\ (i,j) \cdot \vec{N}^n\ (i,j), \tag{2.2}$$

kde $\delta\vec{L}^n\ (i,j)$ je míra změny informace $L^n(i,j)$. Díky rovnici (2.2) je zajištěno, že informace $L^n(i,j)$ na obrazu a vypočtené změny se šíří ve směru $\vec{N}$. Stojí za povšimnutí, že v ustáleném stavu, to je když $I^{n+1}(i,j) = I^n(i,j)$ a když $\delta\vec{L}^n\ (i,j) \cdot \vec{N}^n\ (i,j) = 0$, se informace L šíří směrem $\vec{N}$. Zbývá ještě vyjádřit informace L, která má být šířena, a směr šíření $\vec{N}$.



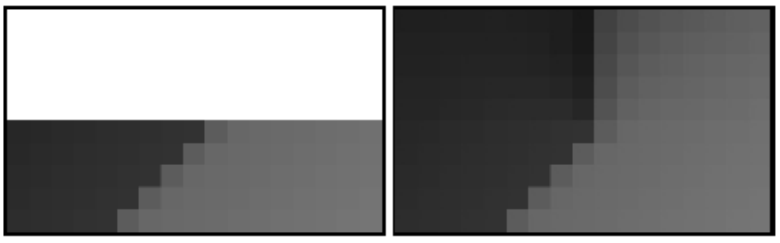
Obr. 2.2: Postupné vyplnění neznámé oblasti.[1]

Cílem metody je, aby vzniklé přechody byly co nejhladší, musí proto informace šíření $L^n(i,j)$ hladce navazovat. Doplňková informace se vypočítá pomocí diskrétního Laplaciánu definovaného jako:

$$L^n(i,j) = \frac{\partial^2 I^n(i,j)}{\partial x^2} + \frac{\partial^2 I^n(i,j)}{\partial y^2}, \tag{2.3}$$

Mohly by být použity složitější výpočty, ale už s touto rovnicí je dosaženo velice uspokojivých výsledků.

Z rovnice (2.3) lze dopočítat změnu $\delta L^n(i,j)$ ve směru $\vec{N}$. Za tímto účelem musí být ještě definován směr $\vec{N}$. Jednou z možností je určit $\vec{N}$ jako normálový vektor směřující od $\partial\Omega$. To znamená, že bod (i,j) v oblasti Ω je dán normálovým vektorem $\vec{N}\,(i,j)$ vycházejícím z $\partial\Omega$ procházejícím bodem (i,j). Předpokladem pro tuto definici je, že se isotopy šíří kolmo k hranici $\partial\Omega$. Na obrázku s neznámou oblastí a s vyplněnou oblastí od kraje (obr. 2.3) je znázorněno, že metoda nebere v úvahu předchozí směr isotop a doplnění není ideální. Pokud totiž isotopy přicházejí v nějakém směru



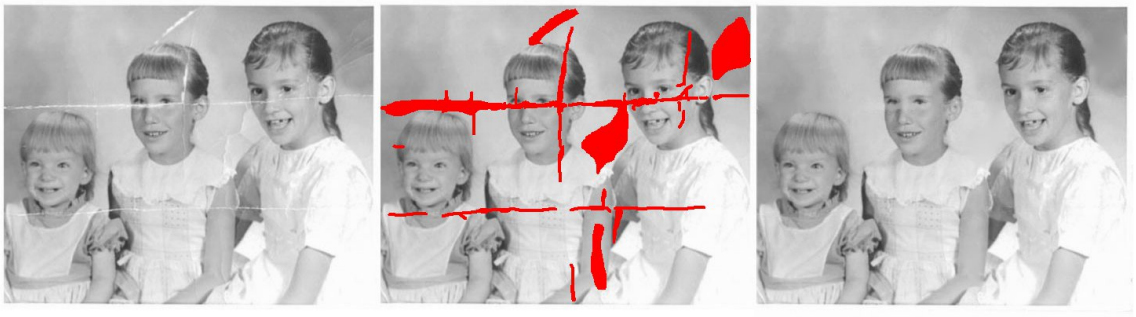
Obr. 2.3: Nedokonalé vyplnění neznámé oblasti.[1]

$\vec{N}$, je nejlepší volbou směr isotop zachovat i uvnitř Ω. Proto Bertalmiova metoda používá vícenásobný odhad směru isotopického pole pro každý bod (i,j). Sklon vektoru $\nabla I^n(i,j)$ udává směr s největší prostorovou změnou, zatím co 90° otočení $\nabla\perp I^n(i,j)$ je směr nejmenší prostorové změny a z toho důvodu vektor $\nabla\perp I^n(i,j)$ udává směr

isotopy. Pole $\vec{N}$ udává vícenásobný odhad $\vec{N}\ (i,j,n) = \nabla\perp I^n(i,j)$, který je ze začátku hrubý, ale postupně dosahuje požadované spojitosti v $\partial\Omega$. Místo pevného pole $\vec{N}$, jež by vyžadovalo znát směry isotop od začátku, je získávána informace postupně a pro každý krok vyplnění zvlášť. Je tak zajištěno, že isotopa může průchodem oblasti Ω měnit směr, protože je ovlivňována i ostatními isotopami.

2.4 Vyhodnocení Bertalmiovy metody

Na obrázku 2.4 vlevo je původní poškozený obrázek. Na prostředním obrázku jsou vyznačena místa, která se budou opravovat a vpravo je obrázek, kde již byl inpainting aplikován. Při důkladném zkoumání jsem narazil na rozmazané úseky opravovaných částí. Na levé ruce prostřední holčičky u loktu a na hřebenu nosu pravé holčičky jsou vidět neostré přechody prolnutí dvou oblastí. I přes tyto dva úseky je celkový dojem z opravy velice dobrý.



Obr. 2.4: Původní poškozený obrázek (vlevo), zvýraznění poškozených oblastí pro opravu (uprostřed), obrázek po odstranění poškozených částí (vpravo).[1]

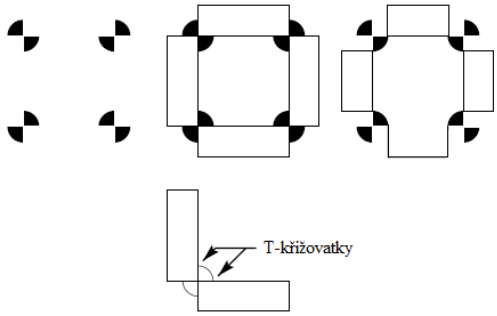
Na druhém obrázku 2.5 je ukázáno odstranění objektu z původního obrázku. Vlevo je tedy původní obrázek a vpravo je obrázek, ze kterého byl odstraněn mikrofon, jenž byl nevhodně umístěn. Spodní kabel mikrofonu je odstraněn bez následků, ale samotný mikrofon a tyč, na které je mikrofon uchycen, po sobě zanechala rozmazanou oblast.

Obr. 2.5: Původní obrázek s mikrofonem (vlevo), obrázek s odstraněným mikrofonem (vpravo).[1]

Teoreticky je tedy Bertalmiova metoda vhodná spíše k opravám jemných nesrovnalostí typu linek, než k opravám rozsáhlejších oblasti. Tento předpoklad byl potvrzen i ve filmu *Kuky se vrací*, který byl natočen s loutkami na vodících nitkách, které byly elektronicky odstraněny právě Bertalmiovou metodou.

2.5 Masnouova metoda

Masnouova metoda [2] se snaží napodobit schopnost lidí, domyslet si, že objekt je překryt jiným předmětem. V přírodě je objekt málokdy zcela viditelný. Naše vnímání však za určitých podmínek dokáže objekt rekonstruovat a vnímat jej jako celek. V první části obrázku 2.6 vidíme čtyři černé mašličky. V druhé části jsou zobrazeny čtyři černé kruhy částečně překryté obdélníky. Ve třetí části je vyznačeno, že by se mohlo jednat o bílý kříž, který je na černém čtverci. Tato schopnost lidského vnímání rekonstruovat částečně skryté objekty byla vědecky studována. Pokračování úrovňových křivek objektů je hlavní myšlenkou Masnouovy metody. Výpočet těchto křivek se provádí od tzv. T-křižovatek, což jsou body, ve kterých je objekt přerušen. Znázornění T-křižovatek vidíme na obrázku 2.6 ve čtvrté části.



Obr. 2.6: Iluze obrázku a T-křižovatky.[2]

Princip spočívá v propojení T-křižovatek, se stejnou úrovní šedé, s minimální délkou křivky s minimální oscilací. Aproximace Eulerovy elastiky

$$\int (1 + k^2)\mathrm{d}s, \tag{2.4}$$

kde s je délka oblouku a k je délka podél okraje. Rovnice (2.4) je efektivní, pokud je obraz více členitý a je zde méně T-křižovatek. Automatické doplnění mezi nimi pak není příliš obtížné, jednoznačně se vybere, co k čemu patří. Pokud se, ale v průběhu obrázku změní úroveň šedé, může s tím mít takto popsaná metoda obtíže. V reálu je změna odstínu naprosto běžná a proto nelze metodu v tomto znění použít.

Popis je tedy zaměřen na rozšířenou metodu, jež vychází z předchozího principu, ale umožňuje obnovit i více poškozené obrázky. Je dána neznámá oblast Ω, která se má vyplnit, s okrajem $\partial\Omega$ na obrázku I. T-křižovatky jsou průsečíky mezi $\partial\Omega$ a úrovňovými křivkami vstupujícími do oblasti Ω z okolí. Každé úrovňové křivce procházející $\partial\Omega$ je přiřazena pozitivní nebo negativní orientace v závislosti na orientaci ∇I ostatních křivek na obrázku I. Pro každou úroveň šedé t zastoupenu

v Ω je definována křivka $(L_i^t)_{i\in I(t)}$ odpovídající nastavení úrovňové křivky. Protože úrovňové křivky procházejí Ω stojí za povšimnutí, že jsou sudé T-křižovatky spojeny s $(L_i^t)_{i\in I(t)}$ jako T-křižovatky s pozitivní a negativní orientací. Potom však problém spočívá v nalezení optimálního nastavení křivek $(\Gamma_j^t)_{j\in J(t)}$, kde $J(t) = I(t)/2$, propojením dvou T-křižovatek se stejnou úrovní, stejnou orientací a minimální energií:

$$E = \int_{-\infty}^{+\infty} \sum_{j\in J(t)} \left(\int_{\Gamma_j^t} (\alpha + \beta \left|k\right|^p) ds + \varphi \right), \tag{2.5}$$

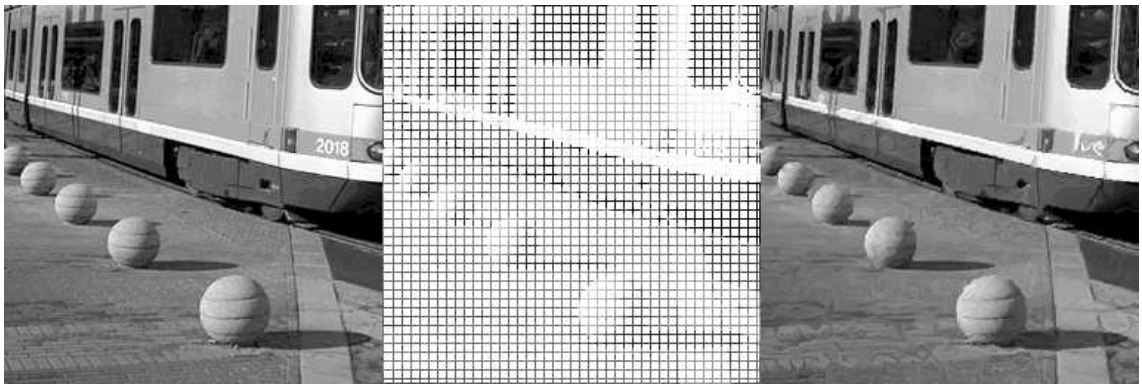
kde α,β jsou kladné konstanty a φ označuje součet, na obou koncích křivky Γ_j^t, úhlů mezi směrem této křivky a směrem související úrovňové křivky. Parametr p lze chápat jako zakřivení objevující se v energetické Eulerově elastice. Metoda popsaná v tomto dokumentu poskytuje optimální řešení pro $p = 1$, což odpovídá pokračování úrovňových křivek polygonální cestou. Z toho vyplývá, že se nebude jednat o přímku, ale křivku, která bude v oblasti Ω různě zahnuta.

2.6 Vyhodnocení Masnouovy metody

Na prvním obrázku 2.7 jsou opraveny bílé skvrny. Teprve po bližším zkoumání jsem narazil na jednu malou chybičku, u opravy rtu není spodní linka tolik výrazná.

Obr. 2.7: Obrázek s chybnými oblastmi (vlevo), opravený obrázek (vpravo).[2]

Na druhém obrázku 2.8 je ukázáno, že Masnouovu metodu lze aplikovat i na obrázky, které jsou značně poškozeny a stačí pouze nástin původního obrázku k doplnění.



Obr. 2.8: Původní obrázek (vlevo), obrázek zobrazený jako čtvercová síť 5x5 (uprostřed), rekonstruovaný obrázek (vpravo).[2]

Oprava prvního obrázku je velice zdařilá a vzhledem k poškození druhého obrázku i ta se dost povedla. Masnouova metoda se jeví jako nejuniverzálnější pro drobné retuše i pro opravy rozsáhlejších oblastí.

3 PRAKTICKÁ IMPLEMENTACE

Pro praktickou ukázku jsem naprogramoval v programovacím jazyce MATLAB dva různé programy na opravu obrázků, které svým principem vychází z již popsaných metod.

Pro programování a testování jednotlivých programů jsem použil notebook Asus řady A6000 (Intel Pentium M procesor 1.8GHz, 1.5GB RAM).

3.1 MATLAB

Název MATLAB vznikl zkrácením slov **MAT**rix **LAB**oratory (volně přeloženo „maticová laboratoř“), což odpovídá skutečnosti, že klíčovou datovou strukturou při výpočtech v matlabu jsou matice.

MATLAB je interaktivní programové prostředí a skriptovací programovací jazyk čtvrté generace. Umožňuje počítání s maticemi, vykreslování 2D a 3D grafů funkcí, implementaci algoritmů, počítačovou simulaci, analýzu a reprezentaci dat i vytváření aplikací včetně uživatelského rozhraní.

Původně byl jazyk určen pro matematické účely, ale časem byl upraven, byly přidány nové funkce a rozšíření, rozrostl se různými směry a dnes je využitelný v široké paletě aplikací.[5]

Všechny programy jsem vytvořil v programovacím jazyce MATLAB v. 7.10.0(R2010a). Ačkoliv jsme neměl v minulosti možnost s tímto programem pracovat, musím kladně ohodnotit nejen prostředí, ale hlavně funkce programu. Spousta vnitřních funkcí i podstata Matlabu, že každá proměnná je matice, mi programování velice ulehčily.

3.2 Obrázky

Obrázky jsou ve škále šedé. Jako vstupní i výstupní formát je použit JPEG. Jedná se o formát se ztrátovou kompresí. Tento formát jsem použil z důvodu jeho rozšířenosti. Většina digitálních fotoaparátů ukládá obrázky právě do formátu JPEG. Jako demonstrační obrázek, na kterém budou ukázány změny různých parametrů jednotlivých programů a jejich vliv na výsledek, je část látky s provázkovou strukturou (obr. 3.1).

Obr. 3.1: Originální obrázek.

3.2.1 Označení chyby

Pro označení chyby jsem vycházel z předpokladu, že obrázky ve škále šedé mají hodnotu všech tří barevných složek stejnou. V grafickém editoru jsem vyznačil „poškozenou“ oblast modrou barvou. Vyzkoušel jsem všechny tři barevné hloubky, ale protože formát JPEG při uložení „rozmazává“ barvu do blízkého okolí a protože lidské oko je nejméně citlivé právě na modrou barvu, vyhodnotil jsem její použití za nejlepší. Na obrázku 3.2 lze pozorovat barevnou „mlhu“ kolem vyznačených oblastí.

Obr. 3.2: Zobrazené rozšířené okolí chyby.

Pro úpravu vstupních obrázků jsem používal grafický editor GIMP 2.6.11. S grafickými editory podobného typu jsem pracoval již v minulosti a tento jsem si vybral kvůli jeho dostupnosti a jednoduchosti.

3.2.2 Nalezení chyby

Obrázek se do matlabu načítá jako trojvrstvá matice, kde každá vrstva reprezentuje jednu barevnou hloubku z barevného modelu RGB.

Program prochází obrázek od horního levého rohu po řádcích k dolnímu pravému rohu a vyhledává předdefinovanou chybu srovnáním dvou vrstev matice obrázku

(jedna z nich musí být modrá) a porovnáním jejich hodnot. Pokud jsou hodnoty rozdílné, není v daném bodě splněn předpoklad, že všechny tři vrstvy obrázku ve škále šedé jsou stejné, znamená to, že bod je špatný a musí se opravit. Vzhledem k tomu, že při ukládání formátu JPEG se chyba projeví i do okolí kolem vyznačené oblasti a jednotlivé matice nejsou stejné ani u obrázku, kde není žádná chybová oblast vyznačena, srovnávání matic má nastavenu hodnotu rozdílu parametrem ICH=13. Při menších hodnotách programy opravují i oblasti, které nejsou označeny jako neznámé. Tuto hodnotu jsem zvolil po několika pokusech, kdy chyba ještě není vidět.

3.2.3 Doba běhu programu

Ke zjištění, jak dlouho programy jednotlivé obrázky zpracovávají, je použita vnitřní funkce v MATLABU, která uloží do proměnné aktuální datum a čas.

Při spuštění programu je ihned po načtení obrázek uložen. Při ukládání je do názvu obrázku vložen aktuální datum a čas. Po provedení všech opravných funkcí je obrázek opět uložen stejným způsobem. Výstupem programu jsou tedy dva obrázky, jeden neopravený s časovým údajem začátku a druhý opravený s koncovým časem. Odečtením časů získáme délku běhu programu.

Pro ukázku je zobrazen obrázek 3.3 se zvýrazněnou chybou (oblast je vybrána náhodně), který se uloží při startu programu s počátečním časem. V závorce je uveden přesný výstupní název.

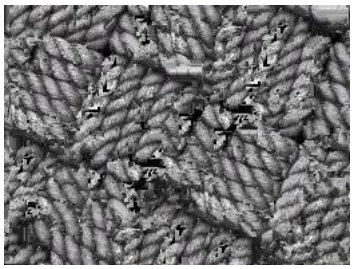
Obr. 3.3: Obrázek se zvýrazněnou chybou uložen při spuštění programu s časem počátku. `vystup_20120514T183633.jpg`

Počáteční číslice znázorňují rok, měsíc, den a za písmenem `T` je uveden čas ve formátu hodiny, minuty, vteřiny. Tento postup jsem použil u obou programů.

3.3 První program - Implementace textur

V prvním programu jsem použil princip implementace textur.

Po nalezení neznámého bodu (souřadnice $[k, l]$) program prohledá oblast kolem tohoto bodu. Při hledání neznámého bodu jsou porovnávány barevné hloubky (vrstvy) obrázku, protože je chyba vyznačena na modré, musí být jedna z vrstev modrá. Rozdíl vrstev jednotlivých bodů, kdy jsou ještě považovány za stejné, je dán parametrem ICH=13. Tuto hodnotu jsem určil po několika pokusech (obr. 3.4, 3.5, 3.6 v závorce je uveden čas, jak dlouho oprava trvala). Při použití menší hodnoty program opravuje i oblasti, kde chyba nebyla vyznačena. Při použití větší tolerance může program vyznačenou oblast opravit jen částečně.



Obr. 3.4: ICH=3 (6:14:34)

Obr. 3.5: ICH=13 (2:04:30)

Obr. 3.6: ICH=35 (01:13:04)

Po nalezení neznámého bodu je prohledávána oblast 7x7 bodů(48 možných známých okolních bodů). Potřebný počet potřebných známých bodů udává parametr $BODY$=24 (obr. 3.7, 3.8, 3.9 v závorce je uveden čas, jak dlouho oprava trvala). Při menší hodnotě jsou výsledky dost neuspokojivé, protože na poškozenou oblast je přenesena oblast, která se s ní dostatečně neshoduje. Při použití větší hodnoty se může vyskytnout problém, že tolik známých bodů v okolí neznámého bodu není a že program pak část obrázku vůbec neopraví.

Obr. 3.7: BODY=15 (0:53:26)

Obr. 3.8: BODY=24 (2:04:30)

Obr. 3.9: BODY=39 (02:29:37)

Po nalezení požadovaného počtu bodů program prohledává obrázek opět z levého horního rohu po řádcích k pravému dolnímu rohu a vyhledává stejnou oblast bodů jako je okolí neznámého bodu.

Tolerance mezi body nalezenými v okolí neznámého bodu a v nově nalezené oblasti udává parametr IK=6 (obr. 3.10, 3.11, 3.12 v závorce je uveden čas, jak dlouho oprava trvala). Při použití menší hodnoty parametru se výsledek na kvalitě nijak neprojevil, ale čas potřebný pro opravu se zvýšil. Při použití větší hodnoty parametru se sice čas zkrátil, ale výsledek se zhoršil.

Obr. 3.10: IK=1 (2:39:08)

Obr. 3.11: IK=6 (2:04:30)

Obr. 3.12: IK=15 (00:59:05)

Při nalezení takové oblasti se přenesou hodnoty všech bodů z oblasti 7x7 bodů (49 bodů) na původní oblast. Přenáší se průměrná hodnota vrstev barvy červené a zelené. Staré hodnoty se přepíší bez ohledu, zda byly známé nebo ne. Obrázek se takto prohlíží několikrát, ale vždy se zvýší hodnota parametru IK a postupně se zvedá až k 50.

3.3.1 Spuštění programu

První program se spouští z prostředí MATLABU najetím do pracovního adresáře `MATLAB_1`, kde jsou uloženy všechny potřebné funkční soubory. Spuštění programu se provede voláním scriptu `textury`.

3.3.2 Načtení a uložení obrázků

Obrázek pro načtení musí být uložen v adresáři `vstup`, který je v pracovním adresáři MATLABU. Jméno souboru musí být `vstup.jpg`. Výstupní obrázek je uložen do adresáře `vystup` v pracovním adresáři MATLABU a je pojmenován `vystup_`*datum*`T`*čas*.

3.3.3 Průběh programu

Po načtení souboru je volán skript pro uložení, aby byl zaznamenán údaj o čase spuštění. Obrázek je zaveden do trojvrstvé matice $IMAGE$. Do proměnné MAX jsou uloženy rozměry této matice - počet řádků, počet sloupců a počet vrstev. Proměnná MAX slouží v programu pro vymezení velikosti obrázku.

Je nastaven parametr IK=6, udává toleranci mezi známými body a hledanými body. Program vstupuje do cyklu, který bude vykonávat, dokud nebude parametr IK=50. Tím je zaručeno několik opakování opravných průběhů obrázku, vždy s větší tolerancí hledaných bodů.

Program prochází obrázek od levého horního rohu, souřadnice [1, 1], po řádcích do pravého dolního rohu, souřadnice jsou uloženy v proměnné MAX na prvních dvou pozicích (počet řádků, počet sloupců). K postupu při hledání jsou jako souřadnice použity proměnné [k, l]. Hledání poškozeného (neznámého) bodu se provádí kontrolou rozdílu dvou vrstev matice $IMAGE$, dvě barevné hloubky obrázku, z nichž jedna musí být třetí (představuje modrou barvu). Rozdíl hodnot je dán parametrem ICH=13. Při malých hodnotách parametru ICH program opravuje oblasti, ve kterých chyba nebyla zvýrazněna, při vyšších hodnotách jsou už okem pozorovatelné šumy. Pokud je rozdíl vrstev větší, je na souřadnicích [k, l] neznámý bod.

Program prohledá okolí neznámého bodu, matice 7x7 bodů se souřadnicemi neznámého bodu uprostřed (pokud je bod na kraji obrázku je tato matice zmenšena). V okolní oblasti je hledán počet bodů uveden v parametru $BODY$=24, u nichž je rozdíl dvou vrstev matice $IMAGE$ menší než ICH, tzn. tyto body jsou známé. Hodnota 24 byla opět zvolena po několika pokusech. Při menší hodnotě program pracuje rychleji, ale výsledky nejsou tak dobré, při vyšších hodnotách se stávalo, že oblast zůstala neopravena, protože v okolí tolik známých bodů nebylo nalezeno.

Počet známých bodů je zaznamenán do proměnné PZB. Vždy při nalezení známého bodu je uložena do nulové matice Ax (9x9 bodů, kde střed představuje neznámý bod na souřadnicích [k, l]) na příslušných souřadnicích hodnota 1.

Po prohledání okolí a nalezení alespoň 24 bodů (může být nalezeno i více) program prochází obrázek od začátku, z levého horního rohu po řádcích k pravému dolnímu rohu, dokud není nalezena stejná oblast bodů. Pohyb po obrázku je nyní zaznamenáván do proměnných [m, n] (řádky, sloupce). Tyto proměnné jsou vždy po opravě oblasti vynulovány, tím je zajištěno, že při zjištění dalšího neznámého bodu bude

obrázek k nalezení známé oblasti prohledáván postupně od začátku. Pokud je na souřadnicích [m, n] bod, jehož rozdíl dvou vrstev matice $IMAGE$ (opět jedna z nich musí být třetí - modrá, testuje se jestli jde o známý bod) vyhovuje parametru ICH, jsou otestovány okolní body podle matice Ax. Bod na souřadnicích [m, n] tvoří střed matice. Porovnávány jsou body, kterým odpovídá hodnota 1 v matici Ax. Tyto body musí vyhovovat podmínce rozdílu dvou vrstev matice (jedna musí být třetí - modrá) a parametru ICH a rozdílu druhých vrstev s body v okolí neznámého bodu na souřadnicích [k, l] (na stejné pozici podle matice Ax) a parametru IK. Úspěšný počet takto nalezených bodů je zaznamenán do proměnné PNB a do nulové matice Ay (9x9 bodů, kde střed představuje neznámý bod na souřadnicích [m, n]) je vždy na pozici nalezeného bodu zapsána hodnota 1. Aby bylo novou oblast možno považovat za adekvátní, musí hodnota proměnné PNB odpovídat minimálně hodnotě parametru $BODY$=24. Pokud není nalezeno takové množství, vynuluje se matice Ay a proměnná PNB a hledání pokračuje po souřadnicích [m, n] dál. V případě, že se v celém obrázku nenajde taková sekvence bodů, jsou posunuty souřadnice [k, l] o bod ke konci obrázku, a proces ověřování a případné hledání známého bodu se opakuje.

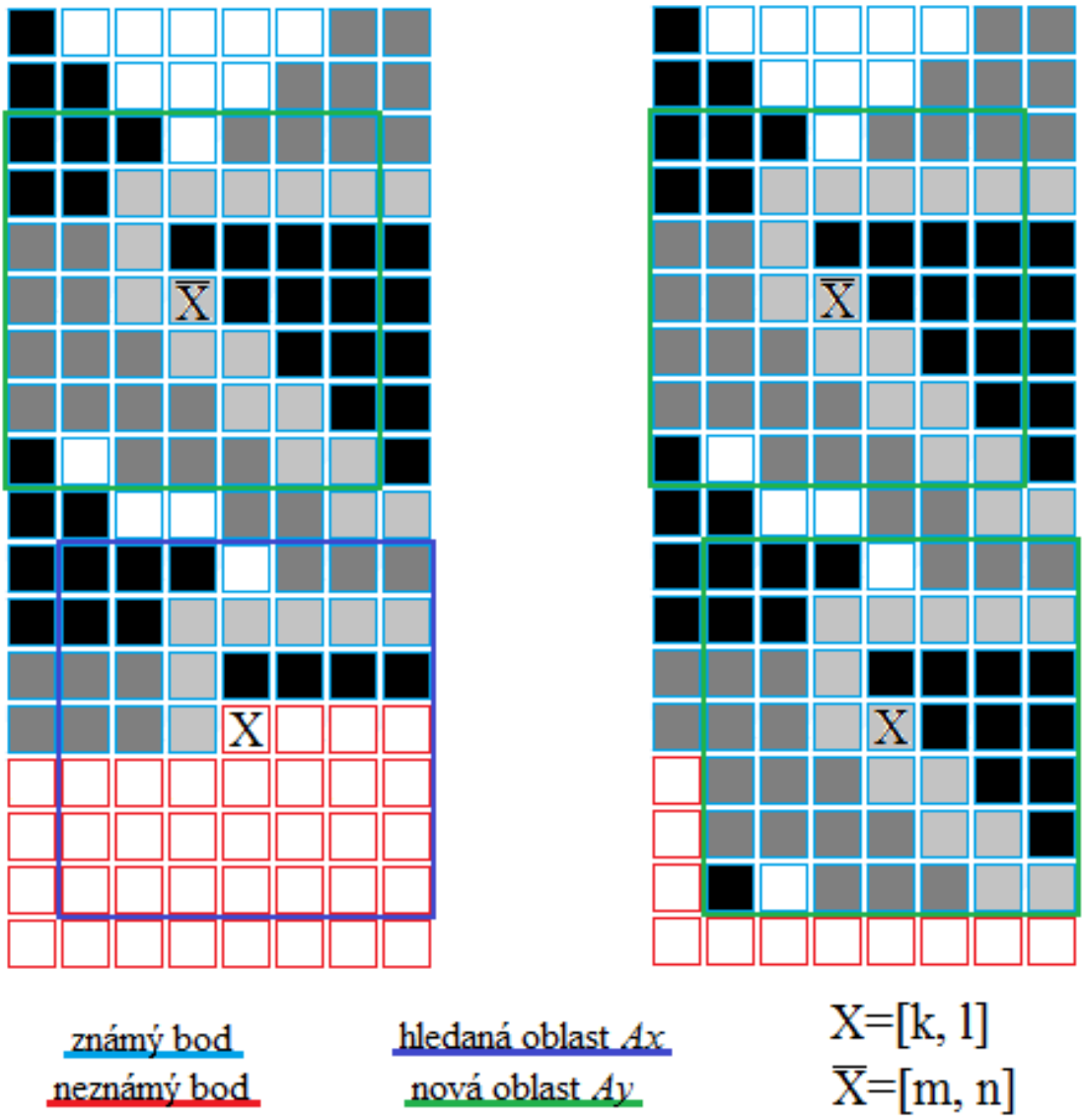
3.3.4 Přenesení nalezené oblasti

Program přenese průměrnou hodnotu vrstev 1 a 2 bodů z nově nalezené oblasti o rozměrech 7x7 bodů, kde středem jsou souřadnice [m, n], na všechny tři vrstvy bodů v původní oblasti o rozměru 7x7 bodů, kde středem jsou souřadnice [k, l] (obr. 3.13).

Pak jsou vynulovány matice Ax, Ay, souřadnicím [m, n] jsou přiřazeny hodnoty počátku [1, 1] a pokračuje se na souřadnicích [k, l] v hledání neznámých bodů.

Po průchodu celým obrázkem je o jedničku zvednuta hodnota parametru IK, jsou změněny hodnoty souřadnic [k, l] na [1, 1] a celý průběh programu se opakuje.

Program je ukončen, když parametr IK dosáhne hodnotu 50. Před ukončením je ještě výsledný obrázek uložen, jméno souboru obsahuje aktuální čas, v tomto případě čas konce.

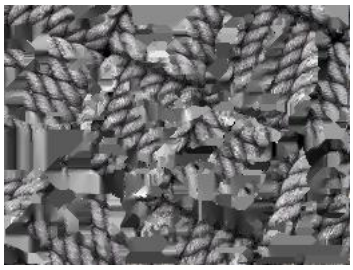


Obr. 3.13: Vyznačené oblasti Ax a Ay a jejich přenesení.

3.4 Druhý program - Bertalmiova metoda

Ve druhém programu jsem vycházel z principu Bertalmiovy metody.

Program projde celý obrázek, hledání probíhá z levého horního rohu po řádcích do pravého spodního rohu a najde v něm všechny neznámé oblasti. Pro nalezení neznámého bodu se opět využívá hodnota rozdílu dvou vrstev, z nichž jedna musí reprezentovat modrou barvu. Tato hodnota je určena parametrem ICH=13 (obr. 3.14, 3.15, 3.16 v závorce je uveden čas, jak dlouho oprava trvala). Při nižších hod-



Obr. 3.14: ICH=3 (0:00:13)

Obr. 3.15: ICH=13 (0:00:02)

Obr. 3.16: ICH=25 (0:00:01)

notách byly programem označeny jako špatné body i ty, které nebyly předem stanoveny. Při vyšších hodnotách se u tohoto programu chyba neprojevila tak markantně jako u předchozího, ale právě ze zkušenosti s předešlým programem jsem se přiklonil k hodnotě parametru ICH stejné jako v něm.

Při nalezení neznámého bodu, je prohledáno jeho nejbližší okolí (8 bodů). Pokud je nalezen pouze jeden známý bod, je jeho hodnota bez dalšího výpočtu přiřazena neznámému bodu. V případě, že body jsou stejné, hodnota rozdílu bodů, kdy jsou považovány za stejné je opět stanovena parametrem ICH=13, je neznámému bodu přiřazena průměrná hodnota těchto bodů. Jestliže není bod takto doplněn, je prohlídnuto širší okolí, oblast 5x5 bodů. Hodnota bodu je určena podle směru odkud se k bodu přistupuje. V okolí bodu jsou hledány oblasti stejných bodů a jejich hranice, podle kterých se hodnota určí. Pokud není nalezena žádná sekvence bodů, která by se dala k tomuto účelu použít, je hodnota neznámého bodu, vypočítána jako průměr nejbližších známých bodů.

Po doplnění neznámého bodu se pokračuje ve směru hodinových ručiček postupně do středu neznámé oblasti. Takto jsou doplněny všechny neznámé oblasti, dokud není opraven celý obrázek.

3.4.1 Spuštění programu

Druhý program se spouští z prostředí MATLABU najetím do pracovního adresáře `MATLAB_2`, kde jsou uloženy všechny potřebné funkční soubory. Spuštění programu se provede voláním scriptu `bertalmio`.

3.4.2 Načtení a uložení obrázků

Obrázek pro načtení musí být uložen v adresáři `vstup`, který je v pracovním adresáři MATLABU. Jméno souboru musí být `vstup.jpg`. Výstupní obrázek je uložen do adresáře `vystup` v pracovním adresáři MATLABU a je pojmenován `vystup_`*datum*`T`*čas*.

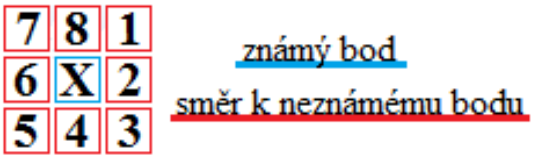
3.4.3 Průběh programu

Po načtení souboru je volán skript pro uložení, aby byl zaznamenán údaj o čase spuštění. Obrázek je zaveden do trojvrstvé matice $IMAGE$. A do proměnné MAX jsou uloženy rozměry této matice - počet řádků, počet sloupců a počet vrstev. Proměnná MAX slouží v programu pro vymezení velikosti obrázku.

Program projde celý obrázek z levého horního rohu po řádcích k pravému spodnímu rohu a zaznamená do nulové matice Ax, která má stelný rozměr jako jedna vrstva matice $IMAGE$, body, kde rozdíl dvou vrstev 2 a 3 je větší než parametr ICH=13. Pokud bude rozdíl větší, uloží na aktuální souřadnice hodnotu -1. Tím budou v matici Ax uloženy neznámé oblasti. Zároveň do matice Ay, která má stejné parametry jako matice Ax, je uloženo okolí těchto oblastí. Při ukládání bodu do matice Ay, je zkontrolováno okolí 3 bodů na všechny strany od aktuální pozice a pokud jsou na těchto souřadnicích známé body, rozdíl vrstev 2 a 3 je menší než ICH, je vložena hodnota -1. Souřadnice bodů jsou ukládány do proměnných [k, l]. Hodnota parametru ICH je zvolena z ohledem na skutečnost, že při nižších hodnotách program může vyhodnotit za špatnou i oblast, kde není chyba vyznačena. To je pravděpodobně způsobeno ukládáním použitého formátu JPEG.

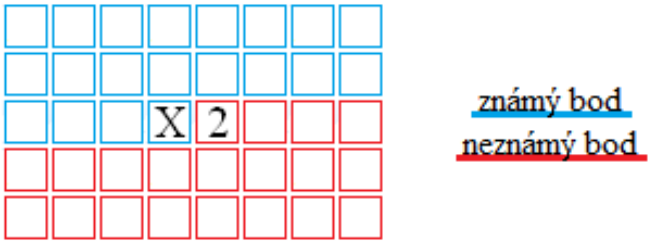
V dalším kroku jsou souřadnice [k, l] nastaveny na počáteční bod [1, 1]. Program prochází matici Ax z levého horního rohu po řádcích do spodního pravého rohu, dokud není nalezena hodnota -1, tzn. vyznačená chyba.

Neznámá oblast je vyplňována podle směru hodinových ručiček od okraje ke středu. Přihlíží se ke směru, odkud se k neznámému bodu přišlo. Pokud je znám tento údaj, je vymezena oblast, ze které se neznámý bod dopočítává. Program rozlišuje 8 směrů (obr. 3.17). Při nalezení prvního neznámého bodu je zvolen směr 2 (při-

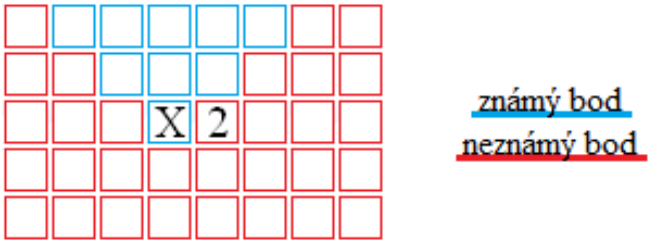


Obr. 3.17: Číslování jednotlivých směrů.

stupuje se po řádcích zleva doprava). Na následujícím obrázku 3.18 je znázorněna oblast, ze které je neznámý bod odvozen. Při vyplňování dalších bodů, ale může dojít i na situaci, kde se známé body v okolí hledaného bobu téměř nevyskytují (obr. 3.19).



Obr. 3.18: Předpoklad známé oblasti pro určení hodnoty neznámého bodu s vyznačením směru pohybu. Známá oblast je velká.



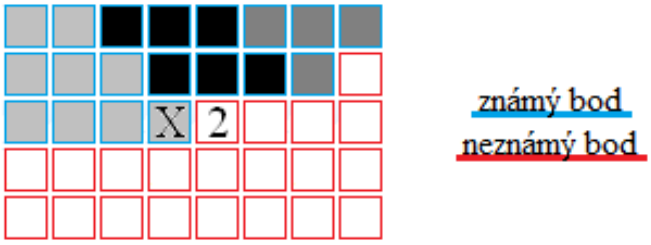
Obr. 3.19: Předpoklad známé oblasti pro určení hodnoty neznámého bodu s vyznačením směru pohybu. Známá oblast je malá.

Při nalezení neznámého bodu je prohledáno nejbližší okolí (1 bod všemi směry). Známé hodnoty bodů jsou zaznamenány do matice Aa, před ukláním jsou všechny hodnoty matice nastaveny na hodnotu -1, rozměr matice je 3x3 bodů se středem v právě nalezeném neznámém bodu. Počet nalezených bodů se ukládá do proměnné PZB. Hodnoty jednotlivých proměnných jsou zaznamenávány do proměnné

typu pole *okoli*. Z takto uložených proměnných je určena maximální hodnota (proměnná *velke*), minimální hodnota (proměnná *male*) a průměrná hodnota (proměnná *prumer*).

Pokud je v okolí nalezen pouze jeden známý bod, je neznámému bodu přidělena jeho hodnota. V případě, že je nalezeno více bodů a rozdíl hodnot v proměnných *velke* a *male* je menší než ICH, tzn. všechny okolní známé body jsou stejné, je neznámému bodu přidělena hodnota v proměnné *prumer*.

Je-li nalezeno více bodů, ale jejich hodnoty jsou rozdílné, je mezi nimi hledána sekvence stejných bodů, podle které by se dal neznámý bod určit. Program nesmí jen kopírovat hodnotu nejbližšího známého bodu, ale musí zkontrolovat širší okolí (matice *Ab* o rozměru 5x5 bodů se středem v neznámém bodě) jestli není neznámý bod na přelomu dvou objektů (křivek) rozdílné barvy a podle toho určit, jakou hodnotu neznámému bodu přiřadit (obr. 3.20).



Obr. 3.20: Předpoklad známé oblasti pro určení hodnoty neznámého bodu s vyznačením směru pohybu. Známá oblast naznačuje výslednou hodnotu neznámého bodu.

Jestliže se ve zkoumané oblasti *Ab* nenajde žádná sekvence znaků, je neznámému bodu přiřazena hodnota z proměnné *prumer*. Po té je ještě v matici Ax změněna hodnota na souřadnicích $[k, l]$ na nulu.

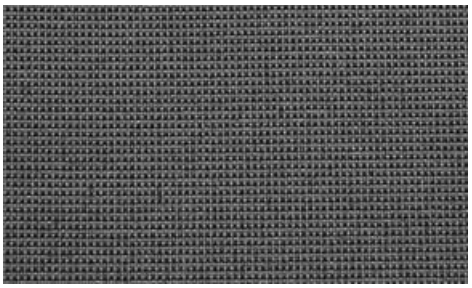
Po opravě neznámého bodu program prohlédne jeho okolí podle pořadí uvedeného na obrázku 3.17 a posune souřadnice $[k, l]$ na nový neznámý bod. Ten je reprezentován hodnotou 1 v matici Ax. Poté se celý proces s určením hodnoty neznámému bodu opakuje. V případě, že již žádný neznámý bod není nalezen jsou souřadnice $[k, l]$ nastaveny na počátek obrázku (souřadnice $[1, 1]$) a vyhledávání neznámé oblasti se opakuje od začátku dokud není celý obrázek opraven.

4 VYHODNOCENÍ PROGRAMŮ

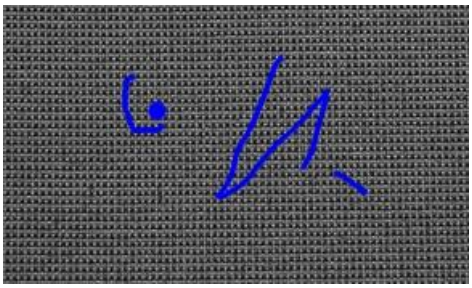
V této části je ukázáno několik obrázků, které byly opraveny zmiňovanými vlastními programy.

4.1 První obrázek - pravidelný rastr

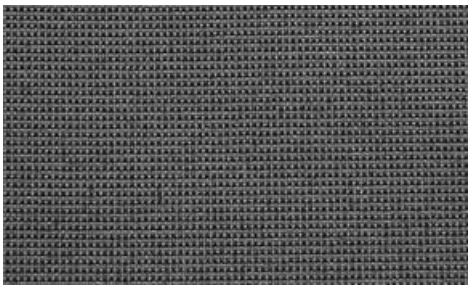
Jako první jsem vybral obrázek blíže neurčitého rastru, jehož vzor se pravidelně opakuje (obr. 4.1), na druhém obrázku 4.2 je znázorněna náhodná vada. Na obrázku 4.3 je chyba odstraněna prvním programem. Oprava je velice zdařilá a ani po důkladném prozkoumávání obrázku jsem nenarazil na nesrovnalosti. Na dalším obrázku 4.4 je zobrazen rozdíl původního a opraveného obrázku a jak je vidět obrázky jsou téměř shodné. Opravení trvalo přes 40 minut. Na obrázku 4.5 je vada opravena druhým programem. Tady se ovšem oprava příliš nepovedla, což je vidět i na zobrazeném rozdílu obrázků (obr. 4.6), který je větší než v předešlém případě. Oprava druhému programu trvala pouze 2 vteřiny. U obrázků tohoto typu je lepší použít první program, zde je výsledek u druhého programu tak nezdařilý, že jeho použití je vyloučeno.



Obr. 4.1: Původní obrázek.

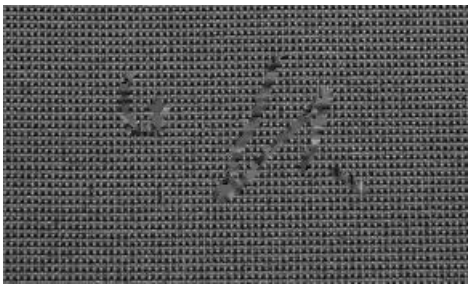


Obr. 4.2: Obrázek s vyznačenou chybou.



Obr. 4.3: Obrázek opravený prvním programem (0:40:45).

Obr. 4.4: Neopravená chyba u opravy prvním programem.



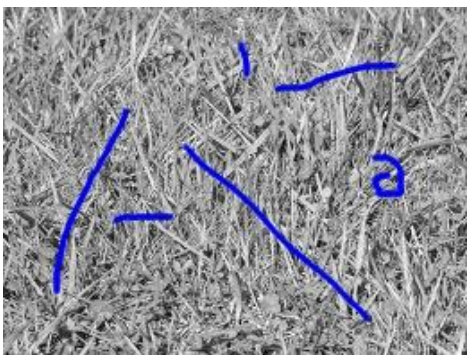
Obr. 4.5: Obrázek opravený druhým programem (0:00:02).

Obr. 4.6: Neopravená chyba u opravy druhým programem.

4.2 Druhý obrázek - tráva

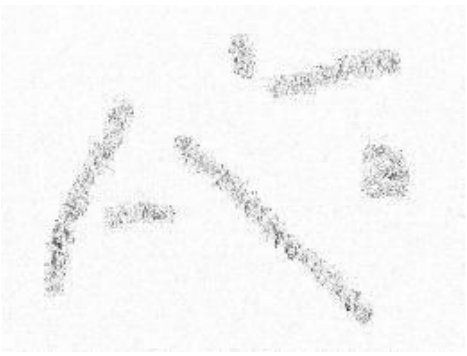
V druhém případě jsem uvedl obrázek trávy (obr. 4.7). Vedle je obrázek s vyznačenou vadou (obr. 4.8). Zde na opraveném obrázku prvním programem (obr. 4.9) i na obrázku opraveným druhým programem (obr. 4.11) je výsledek přijatelný. Při důkladnějším prohledání, lze ale nalézt v opravené části neostré oblasti, které oba programy zanechaly. Na obrázku 4.10 je chyba z opravy prvním programem a na obrázku 4.12 je chyba z opravy druhým programem. Ačkoliv neopravená chyba u druhého programu je menší než u prvního, výsledný obrázek je u prvního programu o trochu lepší. Tento jev připisuji metodě prvního programu, kde více vznikají ostré přechody barev, které zde vypadají přirozeněji, než vyhlazené přechody barev u výsledného obrázku druhého programu. Když však srovnáme čas, který pro opravu potřeboval první program zhruba 2 a 1/4 hodiny a druhý program 2 vteřiny, je na individuálním úsudku, pro který program se rozhodnout, jestli obětovat čas za nepatrně lepší výsledek. Dle mého názoru jsou oba programy použitelné.

Obr. 4.7: Původní obrázek.



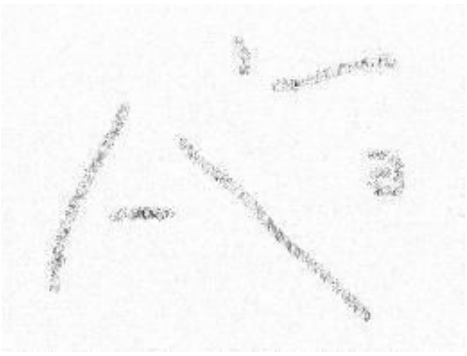
Obr. 4.8: Obrázek s vyznačenou chybou.

Obr. 4.9: Obrázek opravený prvním programem (2:16:43).



Obr. 4.10: Neopravená chyba u opravy prvním programem.

Obr. 4.11: Obrázek opravený druhým programem (0:00:02).

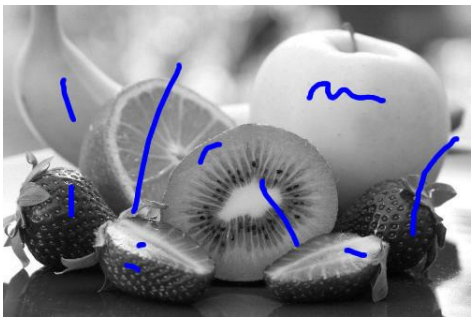


Obr. 4.12: Neopravená chyba u opravy druhým programem.

4.3 Třetí obrázek - ovoce

Pro poslední srovnání použitelnosti programů jsem vybral obrázek ovoce (obr. 4.13). Na dalším obrázku je opět zvýrazněna vada (obr. 4.14). Na obrázku po provedení korekce prvním programem (obr. 4.15), jsou opravené oblasti zřetelně pozorovatelné. Na obrázku 4.16 sice chyba nepůsobí rozsáhle, ale na výsledném obrázku jsou tyto oblasti vnímány negativně. Obrázek opravený druhým programem (obr. 4.17), je mnohem kvalitnější, což dokazuje i chyba zobrazená na obrázku 4.18. Čas potřebný prvním programem (30 minut) je opět výrazně větší než při použití druhého programu (4 vteřiny). U takových obrázků je vhodnější použít druhý program.

Obr. 4.13: Původní obrázek.



Obr. 4.14: Obrázek s vyznačenou chybou.

Obr. 4.15: Obrázek opravený prvním programem (0:30:04).

Obr. 4.16: Neopravená chyba u opravy prvním programem.

Obr. 4.17: Obrázek opravený druhým programem (0:00:04).

Obr. 4.18: Neopravená chyba u opravy druhým programem.

5 ZÁVĚR

Cílem bakalářské práce bylo objasnit pojem inpainting.

V teoretické části jsem na základě uvedených zdrojů popsal podstatu tří metod, kterými lze inpainting aplikovat. Dále jsem teoreticky vyhodnotil jednotlivé metody.

V praktické části jsem naprogramoval dva programy vycházející z teoretických popisů metod.

První program vychází z metody implementace textur. Dle teoretických předpokladů je oprava programem, který vychází z této metody, časově náročná. Výsledná opravená oblast má, pokud není dokonale navázána na původní, ostré hrany, proto bych program doporučil na členité obrázky, kde se opakují stejné nebo podobné sekvence bodů.

V druhém programu jsem použil princip Bertalmiovi metody doplňování neznámých oblastí dopočtem ze známého okraje oblasti. Program mne překvapil rychlostí s jakou obrázky opravoval. Jistě by se ještě daly rozšířit podmínky pro sekvence bodů při dopočtu neznámého bodu, čímž by se výsledek zlepšil. Okraje opravených oblastí jsou často rozmlžené. Program bych doporučil pro méně členité obrázky.

Oba programy si lépe poradí s menšími vadami typu škrábanců než při vyplňování větších spojitých ploch.

LITERATURA

[1] Bertalmio, M.; Sapiro, G.; Caselles, V.; Ballester, C. *Image inpainting.* Proceedings of SIGGRAPH'00, New Orleans, USA,417-424, July 2000.

[2] Masnou, S. *Disocclusion: a variational approach using level lines.* IEEE International Conference on Image Processing, Chicago, USA, 1998.

[3] Masnou, S. *A two-step method for texture/geometry inpainting.* Laboratoire Jacques-Luis Lions, Université Paris 6

[4] Prosuch, J. *Automatické retušovanie poškodených obrázkov.* Bakalářská práce, Bratislava 2010

[5] MATLAB [online]. Wikipedie, otevřená encyklopedie [cit. 2012-05-12]. Dostupné na internetu: <http://cs.wikipedia.org/wiki/Matlab>