Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Rekonstrukce povrchu pomocí scanneru

Plzeň 2013 Ondřej Nedvěd

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 25. června 2013

Ondřej Nedvěd

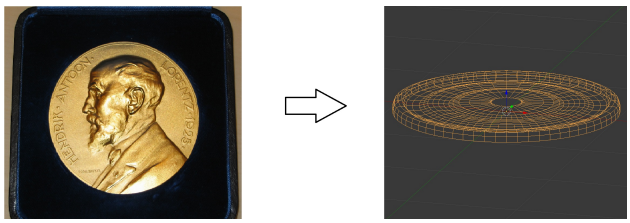
Abstract

Tato práce se zabývá procesem skenování téměř plochých objektů pomocí stolního skeneru a následným získáním prostorového modelu. Patří sem v první fázi rekonstrukce normálového pole a v druhé fázi integrace povrchu. Teoretická část této práce se podrobněji zabývá matematickými vztahy jak pro získání normálového pole, tak vztahy pro integraci povrchu. Ověření funkčnosti, tedy praktická implementace, je popsáno v druhé části práce. Přiložená aplikace umí spočítat a vizualizovat normálové pole i výsledný povrch.

Obsah

1 Úvod

Motivací pro tuto práci byl fakt, že získání prostorového modelu pomocí 3D skeneru je sice známý, nicméně samotný hardware a software pro tento proces je nákladnější než použití klasického stolního skeneru, který je běžně dostupný. Oproti konvenčnímu postupu má však stolní skener svá omezení. Jelikož je původně určen pro papírové dokumenty, lze pomocí něj získat pouze obraz takových předmětů, jejichž povrch je téměř plochý (mince, medaile, pečeti aj.), navíc svými rozměry nesmí přesáhnout okraje skeneru. Na druhou stranu, současné, běžně dostupné stolní skenery mají ve srovnání s 3D skenery srovnatelné rozlišení a lze tedy předpokládat, že lze dosáhnout na vhodných objektech srovnatelných výsledků.



Obrázek 1.1: Lorentzova medaile (foto: Brienanni (c), 2006)

Praktické uplatnění vidím především v oboru archeologie, kde by bylo možné například vzácné mince či jiné drobné předměty naskenovat a kopii v elektronické podobě 3D modelu zaslat kolegům, případně vytisknutou na 3D tiskárně.

2 Teoretická část

2.1 Získání normálového pole

Tento oddíl vychází především z [1] a [2]. První článek z uvedených pojednává o celém procesu získání normálové mapy, druhý pak popisuje proces detekce prostorových změn (konkrétně rýh a přehybů) na fotografiích, naskenovaných stolním skenerem. Technika použití čtyřech snímků, použitá v tomto článku a v mé práci, vychází z [3].

Samotný proces získání normálového pole se skládá z více částí.

2.1.1 Skenování

Tato fáze značeně omezuje výběr předmětů, které je vůbec možné skenovat. Sklo běžného kancelářského skeneru má rozměry o málo větší než stránka formátu A4, je náchylné na poškrábání ostrými hranami a navíc se pod větší vahou prohne či praskne. Je nutností tedy dále předpokládat, že skenujeme minci či objekt velikostí a vahou podobný minci. Pro demonstraci některých problémů však mohou být v textu použity i jiné objekty.

Pro dobrý výsledek je potřeba provést čtyři skeny, navzájem vůči sobě pootočené, vždy o 90°. Snímky označme jako S_0, S_{90}, S_{180} a S_{270}. Při skenování potřebujeme zaznamenat pouze intenzitu světla dopadajícího na snímač, takže skener je vhodné nastavit tak, aby předával pouze jasovou informaci. Ta nechť je označena jako I_0, I_{90}, I_{180} a I_{270}. Je zřejmé, že tyto čtyři skaláry jasu jsou získány v každém[1] místě povrchu. Aplikace pak umí načítat i barevné snímky, uložené v barevném formátu RGB[2]

Skenování je vhodné provádět v co možná nejvyšším přirozeném rozlišení skeneru. Interpolovaná rozlišení by mohla zanést do snímku předem neznámou chybu v závislosti na metodě interpolace.

Ještě před dalším postupem je nutné všechny čtyři naskenované snímky otočit do pozice 0°. Rotace o pravý úhel nijak nenaruší naskenovaný obraz,

[1] Přesněji každém diskrétním.
[2] Viz literatura [8], strana 35.

takže nedojde ke ztrátě informace. Aby bylo u kulatých mincí docíleno co nejpřesnějšího otočení o 90°, je vhodné je připevnit na čtvercovou podložku.

2.1.2 Odstranění případného šumu

Každé zařízení převádějící analogový signál, v tomto případě světlo, na digitální záznam generuje v signálu (obrazu) šum. Bylo tedy nutné zjistit, zda jsou metody výpočtu normálového pole a rekonstrukce povrchu z normálového pole náchylné na zašumění obrazu. Vzhledem k tomu, že nebylo předmětem zkoumat problematiku obecně, byly porovnány výsledky pro neupravené snímky a snímky, na které byl použit patřičný filtr pro odstranění šumu.

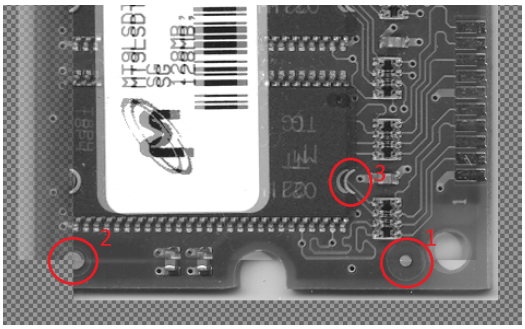
Typicky se v obrazu vyskytuje Gaussovský šum, který lze odstranit vhodnou konvoluční maskou. Při výpočtu normálového pole byl výsledek výpočtu prakticky stejný jak pro zašuměný vstup, tak pro vstup po filtraci. Následná integrace již vychází z vypočteného normálového pole, na které šum nemá významný vliv. Lze tvrdit, že není třeba v programu provádět odstranění šumu z naskenovaného obrazu. Pokud by přesto uživatel trval na odstranění šumu, může vždy použít libovolný externí nástroj a načíst až filtrovaný obraz.

2.1.3 Registrace či „usazení“

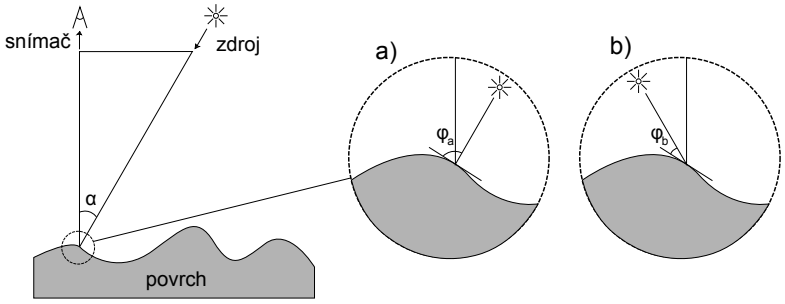
V průběhu zpracování práce se ukázalo, že není možné použít libovolný typ skeneru. Je nezbytně nutné, aby snímač byl v každém okamžiku kolmo k desce skeneru a osvětlení pod konstantním úhlem. V opačném případě nastává problém překrývání vzdálenějších ploch blízkými, to je zřetelně vidět na obrázku 2.1. Prakticky existují především dva typy skenerů a pouze v jednom případě lze provést usazení naskenovaných snímků na sebe.

První případ je skener, kdy kolmo ke sklu skeneru (v každém místě) směřuje osvětlení a snímač se nachází pod určitým úhlem α. Výhodou tohoto rozložení je fakt, že se v obrazu nenacházejí stíny, o kterých bude zmínka dále. Skener s takovouto konfigurací však způsobuje obrazový překryv, zmíněný výše. I když je obvykle obraz zaostřený ve všech místech (včetně vzdálenějších ploch) lépe než v následujícím případě, nelze takto naskenované snímky pro rekonstrukci použít.

Ve druhém případě se kolmo ke sklu skeneru nachází snímač a osvětlení je



Obrázek 2.1: Posun obrazu v závislosti na hloubce. 1, 2 – Přesné zarovnání na vzdálenější úrovni. 3 – Překryv obrazu způsobený různými pozicemi v jednotlivých snímcích na úrovni skla skeneru.



Obrázek 2.2: Schéma umístění snímače a zdroje světla ve skeneru.

vrženo pod úhlem α. Pro představu viz obrázek 2.2. Výhodou je především fakt, že nedochází k žádným posunům obrazu, nevýhod je však hned několik. Ač je tato práce zaměřena na téměř ploché objekty[3], byla provedena i měření na předmětech s větší hloubkou, řádově do 5 mm. Právě zde se projevilo rozmazání objektu přímo úměrné hloubce. To je zřejmě způsobeno konstantním zaostřením na povrch skla skeneru a zároveň velmi malou hloubkou ostrosti snímače. Dalším problémem mohou být stíny, které se vytvářejí v místech, kde se na objektu nachází příkřejší plochy než je α. Usazovat takto naskenované snímky je tedy vhodné provádět na základě těch ploch, které leží přímo na skle skeneru. Právě z důvodu, že v podstatě každý objekt na okrajích vrhá stíny, je automatická registrace obrazu přinejmenším nesnadná, stíny je nutné nejprve detekovat, následně odstranit a pak teprve obraz usadit. Toto nepovinné rozšíření automatické registrace naskenovaných nebylo implementováno a bylo tedy ponecháno ruční usazování, kde stíny nepředstavují tak

[3] I když v zadání není specifikována přesná hodnota, přibližná maximální hloubka předmětu se pohybuje okolo 1 mm.

velkou překážku.

2.1.4 Odstranění deformací obrazu

Jelikož při práci bylo k dispozici omezené množství skenerů a navíc pouze jeden z dostupných měl vhodným způsobem usazenou kameru (viz výše), nemusí tato informace platit obecně, avšak je vhodné či až nutné s deformací naskenovaného obrazu počítat a korigovat ji. Pro prozkoumání možností deformace, jejích typů a rozsahu, byl použit běžně používaný postup, srovnání obrazu vygenerované šachovnice s velikostí čtverce jednoho palce[4]. To proto, že rozlišení všech těchto zařízení je udáváno v horizontálních a vertikálních bodech na jeden palec – DPI.

Nelineární deformace

Bylo testováno zakřivení ve směru pohybu skenovací hlavy i v kolmém směru. Konkrétně bylo zjišťováno, zda v obrazu je či není znatelný „náběh“ motorku skeneru, poháněnějícího snímací hlavu (rozjezd nemůže být okamžitý) a to především na okrajích obrazu. Avšak ukázalo se, že pohyb hlavy skeneru je rovnoměrný již od první snímané pozice na okraji skla skeneru.

Druhý potenciální problém mohlo být zpoždění odesílání dat. Případ, kdy intenzita snímaná na levé části skeneru je zapsána na jinou vertikální pozici[5] než intenzita snímaná na druhé straně hlavy. Předpoklad je, že data jsou podávána sériově po bodech a ne po celých řádcích a mohlo by tak dojít ke zkosení obrazu. I tato možnost však vyla vyvrácena za pomoci měření.

Ostatní nelineární jevy jakou je například známý efekt „poduškovitosti“ či „soudkovitosti“ se taktéž neprojevily. Pro detailní popis (nejen) těchto deformací obrazu na základě optických vad, například čoček ve snímací soustavě, doporučuji literaturu [4].

Vyloučením všech nelineárních deformací obrazu se práce nejen částečně ulehčila, avšak také lze navíc předpokládat, že nedojde k dalšímu zbytečnému rozmazání obrazu při jejich korekci.

[4] $1\ inch = 2.54\ cm$

[5] Hlava skeneru je brána jako horizontálně orientovaná.

Lineární deformace

Důvodem lineárních deformací, které se ve snímcích projevily, je především fakt, že rychlost pohybu hlavy skeneru není zkalibrovaná tak, aby počet bodů na palec v horizontálním směru[6], tedy podél hlavy skeneru, odpovídal vertikálnímu počtu bodů na palec.

Tato deformace nemusí být konstantní ve smyslu poměru horizontálního a vertikálního rozlišení obrazu, pro každý skener se přesné poměry mohou lišit. Pouze z výsledného obrazu lze zjistit tento poměr s přesností na jeden obrazový bod a nelze tedy hovořit o přesných měřeních této deformace, nicméně při opakovaných pokusech s šachovnicovým vzorem byla na zvoleném skeneru při přepočtu z celého obrazu zjištěna přibližná střední hodnota chyby, pohybující se okolo dvou obrazových bodů oproti poměru stran 1 : 1. Přesněji řečeno místo rozlišení 600×600 se rozlišení pohybovalo okolo 600×598. Určitě je možné pořídit přesně zkalibrovaný skener, nicméně pro účely psaní této práce by to byly nadbytečné náklady a bylo nutné se spokojit s faktem, že obraz bude korigován.

Na úkor drobného rozmazání obrazu byla tedy tato chyba odstraněna ve výsledných snímcích škálováním v jednom směru. Pro konkrétní příklady výsledných upravených obrazů viz výsledky na straně 34. Všechny snímky musely být před výpočtem korigovány.

2.1.5 Výpočet výsledné normály v každém bodě

Až doposud se práce zaobírala přípravou snímků pro samotnou rekonstrukci, ta je nezbytně nutná pro další postup. Zde tedy předpokládejme snímky bez deformace, usazené a mající pouze informaci o intenzitě v rozsahu hodnot $0,0$ až $1,0$.

Vzorce a postupy uvedené v tomto oddílu lze s podrobnějším vysvětlením nalézt v [1]. Jelikož článek v době psaní této práce ještě nebyl vydán, budou uvedeny a použity pouze prakticky ověřené vzorce.

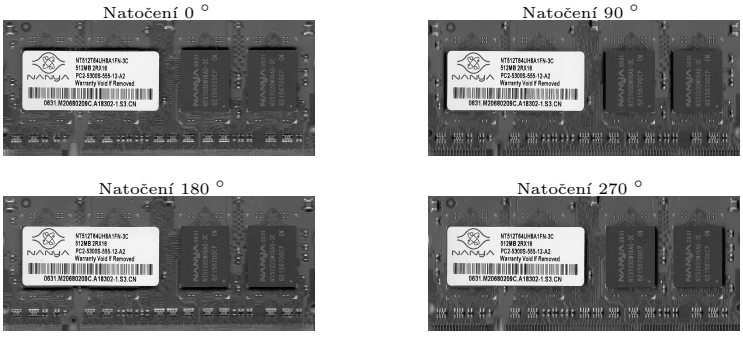
Pro jednoduchost lze prozatím předpokládat, že naskenovaný objekt je téměř plochý (jeho hloubka nepřesahuje 1 mm), neobsahuje ostré hrany a neobsahuje strmé plochy, které by v naskenovaném obrazu vrhaly stín. Máme

[6]Toto rozlišení je pevně dáno usazením snímače/snímačů na hlavě skeneru.

tedy čtyři snímky pro čtyři natočení, jak bylo řečeno v oddílu 2.1.1. Všechny čtyři snímky jsou navzájem usazené a mají přesně stejné rozměry. Tak je tomu i na obrázcích 2.3.

Obrázek 2.3: Naskenované snímky dvacetikoruny, očekávaný případ vstupních dat.

Abych mohl demonstrovat zmíněné problémy, vybral jsem k porovnání snímky RAM modulu, obsahující ostré hrany, stíny i strmé plochy.



Obrázek 2.4: Naskenované snímky RAM modulu, prakticky nejhorší případ vstupních dat.

Dle [1] lze k řešení dospět výpočtem následující soustavy pomocí metody nejmenších čtverců:

$$\begin{bmatrix} 0 & \tan\alpha & 1 \\ 0 & -\tan\alpha & 1 \\ -\tan\alpha & 0 & 1 \\ \tan\alpha & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \begin{bmatrix} I_0 \\ I_{180} \\ I_{90} \\ I_{270} \end{bmatrix} \tag{2.1}$$

Výsledný vzorec (resp. vzorce) pro výpočet normály $[n_x, n_y, n_z]$ v bodě, v němž známe intenzity $I_0 \ldots I_{270}$, je tedy:

$$n_x = \frac{(I_{270} - I_{90})}{2\tan\alpha}, n_y = \frac{(I_0 - I_{180})}{2\tan\alpha}, n_z = \frac{(I_0 + I_{90} + I_{180} + I_{270})}{4}. \tag{2.2}$$

Úhel α je právě úhel svíraný vektory směru osvětlovací diody v hlavě skeneru a směru, ze kterého na snímač dopadá odraz. Úhel α nemusí být ve specifikaci skeneru udán, nicméně lze jej snadno spočíst z délky vrženého stínu na rovnou podložku. Důkaz funkčnosti vzorce byl proveden pouze empiricky. Výsledky je možné shlédnout na straně 34.

V průběhu ověřování funkčnosti postupu se projevily spíše pozitivní vlastnosti metody. Především se ukázalo, že Gaussovský šum, přítomný v podstatě v jakémkoli digitálním snímku získaném ze skeneru, nemá na výpočet normálového pole prakticky žádný vliv. Při porovnání výsledků vypočítaných ze snímků po odstranění šumu a snímků obsahujících šum byly rozdíly zanedbatelné.

Na snímcích objektů s větší hloubkou (například zmíněné RAM moduly) lze pozorovat rozmazání v daném místě, závislé na vzdálenosti odpovídající „plošky“ od skla skeneru. Toto rozmazání je u téměř plochých objektů minimální a bylo by možné jej vynechat z bodu zájmu, nicméně ani u „hlubších“ objektů se neprojevil závažný vliv rozmazání naskenovaných snímků na výsledné normálové pole. Efekt se projevuje větším vyhlazením detailů. Zde by bylo vhodné srovnání s profesionálním skenerem, u kterého by nedocházelo k žádnému rozmazání ve větších hloubkách. Ten však nebyl k dispozici a bylo nutné se spokojit s předpokladem, že výsledné rekonstruované normálové pole má správný průběh a že srovnání bude možné provést pouze vizuálně.

2.1.6 Fyzikální a matematický náhled

Po četných konzultacích s několika odborníky v oboru matematiky[7] bylo uznáno, že již samotná rekonstrukce normálového pole ze čtyř „plochých“ snímků je z čistě matematického hlediska nemožná a je nutné zauvažovat nad fyzikálním náhledem na věc. Jednoduše – slovy – lze výše uvedený vzorec pro výpočet normál vysvětlit následovně.

Pokud světlo dopadá na plochu pod ostrým až nulovým úhlem vůči normále, je jas dané plochy maximální. Naopak pro téměř kolmý úhel je jas minimální až nulový. Výsledný vzorec kombinuje osvětlení plochy předmětu „zleva“ a „zprava“ do jedné složky (x) a obdobně ze zbylých dvou směrů (y). Úhel α odpovídá odklonu směru osvětlení od svislice.

2.2 Rekonstrukce povrchu

Rekonstrukcí povrchu je zde míněna úloha, kdy na vstupu je normálové pole, následné jsou stanoveny okrajové podmínky pro tuto časově neměnnou úlohu a je hledána taková integrace povrchu, která bude dle vybrané metody nejlepší. Přesněji řečeno, jelikož se po naskenování pohybujeme v diskrétním světě, hledáme v každém skenerem zaznamenaném místě na pozici $[x, y]$ jeho hloubkovou informaci tak, aby celku odpovídalo výše vypočítané normálové pole.

2.2.1 Okrajové podmínky

Počet okrajových podmínek samozřejmě závisí na vybrané metodě, nicméně jejich volbu lze provést na základě faktu, že známe skenovaný předmět a dokážeme například určit, která místa objektu se dotýkají skla či která místa jsou na stejné hloubkové hladině.

Jelikož je předmět naskenován velmi detailně[8], je nesnadné určit s přesností na jeden bod, která část okraje leží přímo na skle. Není neobvyklé, že na předmětu mohou být vrypy, na první pohled přehlédnutelné. Bylo nutné vymyslet automatizovaný či alespoň částečně automatický přístup hledání

[7] Doc. RNDr. Miroslav Lávička, Ph.D., Doc. Ing. Marek Brandner, Ph.D.

[8] I obyčejné, běžné skenery umí skenovat s rozlišením 600 bodů na palec.

okrajových podmínek. Ve výsledné implementaci je umožněno ruční zadání okraje po částech i automatické dopočtení na základě zadání výšky v rozích snímků.

První možností je tedy ruční zadání okrajových hodnot, kdy si uživatel rozdělí okraj snímku na jednotlivé úseky, kterým přiřadí konstantní výšku. Vzhledem k tomu, že není možné automaticky zjistit rozlišení načtených vstupních snímků, musí být hodnoty zadávány v jednotkách, kdy jedna jednotka je jeden obrazový bod. Pro minci o průměru jednoho palce, naskenované v rozlišení 600 DPI je jeden milimetr hloubky roven přibližně 24 jednotkám.

Druhou variantou je téměř plně automatizovaný přístup, který byl také implementován. Celý postup sestává ze dvou kroků:

1. Uživatel stanoví hloubkové hodnoty v rozích obrazu.
2. Program dopočte řešením okrajové úlohy prvního řádu hodnoty po celé délce okraje.

Ještě další je možnost poloautomatizovaná, kdy jsou nalezeny ploché oblasti (x a y složky normálového pole jsou v takovýchto místech blízké nulovým hodnotám) a teprve poté je jim uživatelem přisouzena hloubková informace. I když nápad vypadal zprvu schůdně, realizace není v tomto bodě reálná, jelikož se neobejde bez stanovení ϵ koeficientu, určujícího maximální naklonění normály, které ještě bude považováno za nulové. Navíc pro výpočet je potřeba znát hodnoty vnějšího okraje, což tento přístup nezaručí.

2.2.2 Řešení pomocí radiálních bázových funkcí

Úvod do RBF

Interpolace pomocí radiálních bázových funkcí (dále jen RBF) je metoda, používaná například pro případy, kdy na vstupu jsou neuspořádaná data, mezi kterými chceme interpolovat. Pro více detailů viz [5] či jinou literaturu, zabývající se tímto tématem.

My potřebujeme využít RBF pouze jako prostředek, který umí pracovat se všemi body zároveň a dalo by se tedy předpokládat, že má globální charakter a může poskytnout dobré výsledky. Začněme tedy od slovní definice:

Radiální bázová funkce je taková funkce, jejíž hodnota závisí pouze na vzdálenosti od zvoleného počátku c.

$$\Phi(r) = \Phi(\|x - x_i\|) \tag{2.3}$$

Norma nemusí být vždy Euklidovská vzdálenost, avšak v našem případě je vhodná právě tato. x_i je i-tý bod ve snímku a počítány jsou vzdálenosti od všech ostatních bodů x. Bázové funkce mohou být různé, nicméně mezi nejpoužívanější se řadí především polyharmonické spliny:

$$\Phi(r) = r^k, \text{ kde } k \in \{1, 3, 5, \ldots\}, \tag{2.4}$$

$$\Phi(r) = r^k \cdot \ln(r), \text{ kde } k \in \{1, 3, 5, \ldots\}. \tag{2.5}$$

Speciálním případem je *Thin Plate Spline* (TPS) funkce:

$$\Phi(r) = r^2 \cdot \ln(r) \tag{2.6}$$

Předpis pro základní interpolaci[9] pomocí RBF má tvar:

$$f(x) = \sum_{i=0}^{N} \omega_i \Phi(\|x - x_i\|) \tag{2.7}$$

ω_i označuje lineární kombinaci hodnot vybrané radiální bázové funkce.

Odvození řešení

Ještě před tím než bude uvedeno odvození, je nutné získat z vypočteného normálového pole x a y složky gradientu, určující parciální derivace, respektive diference[10]. Známe-li $\mathbf{n} = [n_x, n_y, n_z]$, pak:

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} = -\frac{n_x}{n_z} \tag{2.8}$$

Zřejmě využijeme základního předpisu pro RBF interpolaci a k tomu ještě jeho derivaci, tedy:

$$f(x) = \sum_{i=0}^{N} \omega_i \Phi(r(x)), \tag{2.9}$$

[9] V některých zdrojích se lze setkat s pojmem RBF aproximace, avšak jelikož výsledná $f(x)$ prochází všemi zadanými body, jedná se skutečně o interpolaci.

[10] Jelikož se při výpočtech pohybuji v diskrétním světě.

$$f'(x) = \sum_{i=0}^{N} \omega_i \Phi'(r(x)) \cdot r'(x). \tag{2.10}$$

Dále budeme pro jednoduchost předpokládat rovinný případ x, y, viz obrázek 2.5. V krajních bodech známe $f(x_0) = y_0$, $f(x_N) = y_N$ a ve všech ostatních diskrétních bodech známe $f'(x)$, respektive normálu. Jelikož máme k dispozici první derivace, řešíme úlohu prvního řádu a k té postačí jedna okrajová podmínka, tedy $f(x_0) = y_0$. Navíc derivace normy vzdálenosti je v tomto případě rovna $r'(x) = 1$, což zjednoduší zápis.

Pro přehlednost výslednou soustavu zapišme maticově, detailnější vysvětlení následuje poté:

$$\begin{bmatrix} \Phi(\|x_0 - x_0\|) & \Phi(\|x_0 - x_1\|) & \cdots & \Phi(\|x_0 - x_N\|) \\ \Phi'(\|x_1 - x_0\|) & \Phi'(\|x_1 - x_1\|) & \cdots & \Phi'(\|x_1 - x_N\|) \\ \Phi'(\|x_2 - x_0\|) & \Phi'(\|x_2 - x_1\|) & \cdots & \Phi'(\|x_2 - x_N\|) \\ \vdots & \vdots & \vdots & \vdots \\ \Phi'(\|x_N - x_0\|) & \Phi'(\|x_N - x_1\|) & \cdots & \Phi'(\|x_N - x_N\|) \end{bmatrix} . \begin{bmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \\ \vdots \\ \omega_N \end{bmatrix} = \begin{bmatrix} f(0) \\ f'(1) \\ f'(2) \\ \vdots \\ f'(N) \end{bmatrix} \tag{2.11}$$

Celkem máme $N+1$ neznámých (ω_0 až ω_N) a N rovnic. Řešením regulární soustavy s pravou stranou získáme vektor ω, který lze použít pro rekonstrukci a to již standardním postupem:

- Výpočet radiální funkce pro hledaný bod x, ke kterému hledáme funkční hodnotu – $\mathbf{\Phi_q}$,
- vynásobení vektorem ω,
- výsledkem je funkční hodnota v hledaném bodě x.

Při řešení takovéto soustavy lze využít symetričnosti a pozitivní definitnosti odpovídající matice. Velkou nevýhodou je zřejmá velikost soustavy a zaplnění matice[11]. Pro představu při výpočtu povrchu ze snímků o velikosti 600×600 bodů je nutné řešit soustavu $360\,000 \times 360\,000$.

[11] Tato matice je hustá, není pásová.

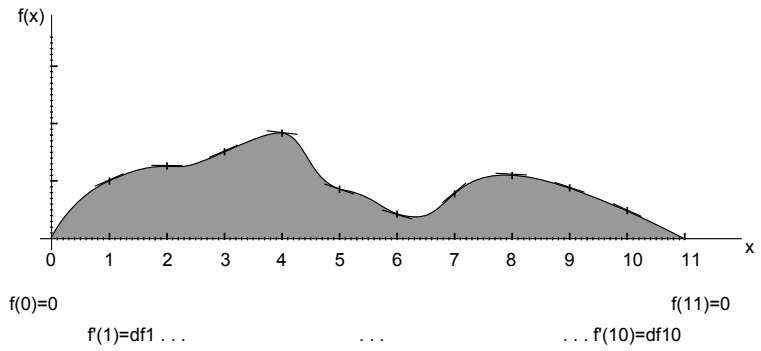
Závěr RBF

Především vzhledem k výpočetnímu času by byla tato volba pro větší objekty neúnosná. Bylo tedy nutné najít rychlejší, efektivní metodu integrace povrchu.

Na základě dalších konzultací s odborníky v oboru matematiky bylo rozhodnuto použití soustavy diferenciálních rovnic od řešení pomocí RBF bylo upuštěno.

2.2.3 Řešení pomocí soustavy rovnic prvního řádu

Na dvourozměrném případě se postup bude lépe vysvětlovat. Tento případ je tedy znázorněn na obrázku 2.5. Povrch objektu $f(x)$ je řekněme diskrétní funkce s definičním oborem obecně $\mathcal{D} \in \mathcal{Z}$ z podoboru celých čísel, jelikož krok je vždy roven jedné jednotce. Lze tedy soudit, že je vhodné se nad problémem zamyslet právě již v diskrétní podobě a nepředpokládat spojitý problém, i když by řešení bylo obdobné.



Obrázek 2.5: Na krajích známe výškovou hodnotu, v ostatních místech známe jen normálu.

Po dalších měřeních bylo rozhodnuto, že pro řešení bude vhodné použít dvě okrajové podmínky, konkrétně $f(0) = y_0$ a $f(N) = y_N$. Takovéto zadání totiž vychází z centrální diference, která má „vyhlazovací“ vlastnost a navíc by nemělo docházet k tak velké distribuci chyby při výpočtu soustavy.

Vyjděme tedy z definice centrální diference:

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{f(x+h) - f(x-h)}{2 \cdot h}. \tag{2.12}$$

Krok h je roven jedné, takže lze rovnou zapsat:

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \frac{f(x+1) - f(x-1)}{2}, \tag{2.13}$$

$$2 \cdot f'(x) = f(x+1) - f(x-1). \tag{2.14}$$

Výsledná soustava rovnic tedy bude vypadat následovně:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ -1 & 0 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 0 & 1 & \cdots & 0 \\ \vdots & & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix} . \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ \vdots \\ f(N) \end{bmatrix} = \begin{bmatrix} y_0 \\ 2 \cdot f'(1) \\ 2 \cdot f'(2) \\ \vdots \\ y_N \end{bmatrix} \tag{2.15}$$

Na rozdíl od matice pro RBF je tato pásová a lze ji tedy řešit rychleji za pomoci algoritmů pro řešení řídkých matic.

Řešení soustavy rovnic

Pro řešení soustavy rovnic lze použít jednak přímé metody a nebo iterační metody. Mezi přímé metody patří především *Gaussova eliminační metoda* a *LU-rozklad*. [11] k iteračním patří například *Gauss-Seidelova metoda* nebo *Jacobiova metoda*.[11]

Stabilnější, nicméně algoritmicky i implementačně složitější možností je *Singulární rozklad (SVD)*[12]. V čase $O(M^3)$ nalezne řešení, což je nepřívětivé pro velké soustavy, v řádu $M \approx 1000$. Tato bývá implementována ve většině výpočetních knihoven a je vhodné s k možnosti použití knihovny přiklonit namísto vlastní implementace.

Pro případ rekonstrukce povrchu pomocí skeneru nejsou však přímé metody příliš vhodné, jelikož u nich není možné předem stanovit maximální chybu, ke které mají konvergovat.

[12]Singular Value Decomposition

Nejschůdnějším přístupem v tomto bodě tedy bylo přenechat řešení soustavy řídké matice na vybrané výpočetní knihovně, viz implementační část. Knihovna iterativně minimalizuje chybu až ke stanoveným mezím a výsledky tohoto řešení jsou uspokojivé, viz strana 42.

2.2.4 Řešení pomocí soustavy rovnic druhého řádu

Jako další metoda bylo testováno řešení zadaného problému jako úloha druhého řádu. Zůstává zatím otázkou, zda má smysl hledat toto řešení, když výše zmíněná metoda funguje. Důvodem je pouze možnost srovnání.

Z definice si nejprve odvoďme druhou diferenci:

$$f''(x) = (f'(x))' \tag{2.16}$$

a tedy:

$$\frac{\mathrm{d}^2 f}{\mathrm{d}x^2} = \frac{\frac{f(x+h+h)-f(x-h+h)}{2h} - \frac{f(x-h+h)-f(x-h-h)}{2h}}{2h} \tag{2.17}$$

a po úpravě:

$$\frac{\mathrm{d}^2 f}{\mathrm{d}x^2} = \frac{f(x+2h) - 2f(x) + f(x-2h)}{4h^2}. \tag{2.18}$$

Pokud opět dosadíme $h = 1$, pak lze dostat vztah mezi $f(x)$ a $f'(x)$ následovně:

$$2 \cdot f'(x+1) - 2 \cdot f'(x-1) = f(x+2) - 2f(x) + f(x-2). \tag{2.19}$$

Z maticového zápisu je patrná soustava rovnic:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & -2 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & -2 & 0 & 1 & \cdots & 0 \\ \vdots & & & & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \cdots & 0 \end{bmatrix} \cdot \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \\ f(4) \\ \vdots \\ f(N) \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ 2 \cdot (f'(x_3) - f'(x_1)) \\ 2 \cdot (f'(x_4) - f'(x_2)) \\ \vdots \\ y_{N-1} \\ y_N \end{bmatrix} \tag{2.20}$$

Použitím této metody výpočtu bylo dosaženo v měřitelných testech závěru, že její výsledky jsou (pokud zanedbáme strojovou nepřesnost) prakticky totožné s výsledky úlohy prvního řádu. Implementace do přiloženého programu tedy nebyla provedena a ve výsledcích na straně 34 lze najít pouze testy na křivkách s analyticky zadaným průběhem, kde je možno přesně změřit chybu výpočtu.

Důvod shodnosti výsledků lze snadno vysvětlit. Pokud máme již v zadání, tedy jako vstupní data, první diferenci, pak druhou získáme pouze z té první a výsledky tedy musí být totožné. Ze stejných dat musíme vždy dosáhnout stejných výsledků. Nicméně se potvrdila korektnost předchozích úvah, což je důležité.

3 Realizační část

Program přiložený k této práci byl vyvíjen pro běhové prostření .NET verze 2.0, jelikož je podporováno drtivou většinou operačních systémů ať již v nativní podobě od společnosti Microsoft ° či v otevřené podobě Mono[1]. V průběhu vývoje aplikace se tato volba potvrdila jako dobrá, nebylo nutné řešit problémy netýkající se aplikace samotné (chyby knihoven etc.).

Program umí snímky získat ze skeneru či načíst ze souboru. Dále je implementován celý výpočet normálového pole a jeho vizualizace a samozřejmě výpočet integrace povrchu a plošná vizualizace výškové mapy. Jak normály tak průběh je možno sledovat v řezu na zvoleném místě. Pro další detaily viz uživatelskou dokumentaci.

3.1 Fáze skenování

V původním návrhu aplikace bylo předpokládáno, že uživatel použije vlastní software k získání snímků ze skeneru, nicméně běhové prostředí .NET podporuje službu *Windows Image Acquisition*[2], která práci značně usnadnila a skenování bylo možno zabudovat přímo do přiloženého programu.

Použití tohoto rozhraní následuje. Voláním:

```
Image img = Scanner.getImage();
```

se spustí obslužná procedura k získání obrazu ze skeneru, která zaobaluje volání knihovních funkcí. Zde je nejprve uživateli zobrazen výběr zařízení, podporujících WIA verze 2.0. Poté je zjištěno nastavení skeneru a jsou nastaveny výchozí hodnoty, pokud to ovladač podporuje. Následně se spustí program ovladače skeneru, který je jiný pro každé zařízení. Uživatel si zde může nastavit jiné hodnoty rozlišení, výřezu a jiné specifické hodnoty. Tento zobrazený dialog již vrací programu přímo hodnoty naskenovaného obrazu a není nutné další zpracování. Nevýhodou je, že uživatel může nastavit libovolné parametry skenování a v dalších výpočtech je nutné počítat s možností, že obraz ze skeneru bude například barevný a může mít i jiné než přednastavené rozlišení.

[1] Multiplatformní běhové prostředí .NET od společnosti Novell, Inc.

[2] Dále jen WIA.

Celý proces je tedy nezávislý na operačním systému i na použitém zařízení. Je však na zvážení uživatele, zda je právě jeho zařízení vhodné pro rekonstrukci povrchu. Pořád je nutné mít na paměti, že skener musí mít snímač kolmo ke skenovanému předmětu a osvětlení pod konstantním úhlem vůči hlavě skeneru.

Dalším poměrně nepříjemným omezením je fakt, že kancelářské skenery jsou navrženy pro skenování dokumentů a že výstup má omezené rozlišení jasových hodnot. Z pohledu aplikace tedy nelze předpokládat, že počet úrovní jasu bude větší než 256.

I pokud chce uživatel snímek načíst ze souboru, musíme předpokládat převod z barevného systému RGB[3], který se ustálil pro práci s obrazem v počítačích jakožto většinový, na hodnoty jasu. Jelikož je pro zpracování vyžadována pouze jasová informace, musíme brát v potaz především tři možnosti dodaného obrazu:

1. Barevný obraz, který má 3 byty pro každý pixel, 1 byte na složku.
2. Šedotónový obraz, který má také 3 byty na každý pixel, ale všechny tři obsahují stejnou hodnotu – jas.
3. Šedotónový obraz, který má 1 byte na každý pixel, určující jas.

V prvním případě byl pro převod použit transformační vektor $[0.299, 0.587, 0.114]$[4], kterým lze barevný RGB obraz transformovat na achromatický obraz. V druhém případě lze buď detekovat tento typ uložení obrazu a následně vybrat pouze jednu libovolnou složku nebo použít stejný postup jako pro RGB obraz. Poslední případ není třeba komentovat, hodnoty není třeba převádět.

3.2 Úprava obrazu před výpočtem

Ještě před samotným výpočtem je nutné zajistit, aby všechny snímky byly navzájem identicky umístěny. Jelikož již při skenování musí být snímky vůči sobě natočeny přesně o 90° a tím pádem i objekt musí být vždy umístěn (co se natočení týče) přesně na daný úhel, stačí snímky vzájemně posunout tak, aby objekt na nich byl v zákrytu.

[3]Sled bytů pro červenou, zelenou a modrou barvu.
[4]Zdrojem je literatura [9].

Celý postup přípravy je tedy následující:

- Naskenování čtyřech snímků pro jednotlivá natočení $0°, 90°, 180°, 270°$.
- Otočení každého snímku o zápornou hodnotu jemu odpovídajícímu natočení (bezztrátové).
- Posunutí snímků po x a y tak, aby obrazy naskenovaného objektu byly v zákrytu.
- Oříznutí snímků podle největší společné části, tedy aby v žádném snímku nechyběla data na okraji.

Registrace bez asistence uživatele byla v původním plánu, i když nebyla v povinném zadání práce, avšak vývoj postupu pro korektní registraci by byl časově příliš nad rámec této práce. Automatická vzájemná registrace snímků by byla pohodlná pro uživatele, ovšem lze zauvažovat nad některými překážkami. Postup by se skládal z následujících kroků:

- Konvoluce hranovým operátorem pro nalezení hran a zlomů, podle kterých lze snímky snáze zarovnat.
- Určení vzájemně si odpovídajících hran.
- Určení obecné transformace tak, aby byly snímky v zákrytu.
- Aplikace transformace a oříznutí všech snímků.

Právě v určení vzájemnosti hran tkví problém. Pro stejnou fyzickou hranu na objektu se totiž může při různých úhlech nasvícení detekovat hrana v snímku na odlišné pozici. To způsobí, že by ve výsledku byl snímek deformovaný. Důvodem posunutí hran je pozice odlesků a stínů, ty se mění v závislosti na natočení objektu; přesněji každé plošky objektu vůči osvětlení. Z toho vyplývá, že v přiloženém programu automatické usazení snímků implementováno není a může tak být předmětem práce navazující na tuto.

Je všeobecně známo a prakticky vyzkoušeno, že běhové prostředí .NET verze 2.0[5] neumí efektivně pracovat s jednotlivými obrazovými body a pro práci na nižší programové úrovni je nutné použít klauzuli `unsafe`. Postup při přechodu od objektu obrazu `Image` k poli '`float *`' následuje.

[5]Ve vyšších verzích až do 4.0 tomu není jinak.

Nejprve jsou uzamčena obrazová data. Ačkoli se aplikace může jevit jako jednovláknová, v praxi k datům může přistupovat vláken více a je nutné dbát na správnou synchronizaci.

Dále je odpovídající kus kódu uzavřen do „nebezpečného“ oddílu `unsafe { ...}`. Slovo `unsafe` spíše varuje programátora, že pracuje přímo s byty v paměti a musí vědět co dělá, jinak kód uvnitř nijak nebezpečný není. Tímto se běhovému prostředí sdělí, aby objekty použité uvnitř bloku nepřesouval na jinou adresu paměti. Je rovněž známo, že v různých běhových prostředích se mohou objekty aplikace přesouvat na jiné adresy i za běhu.

Uvnitř bloku lze používat ukazatele do paměti a samotné kopírování včetně případných transformací běží již dostatečně rychle.

Jako příklad je níže uveden kus kódu, který kopíruje kladnou část rekonstruovaných normál do obrazu. Právě jejich výpočtem se zabývá následující oddíl této práce. Postup je nicméně vždy analogický.

```
Bitmap nb = new Bitmap((int)sourceImages[0].width,
                       (int)sourceImages[0].height);
BitmapData bd = nb.LockBits(new Rectangle(0, 0, nb.Width, nb.Height),
                            ImageLockMode.WriteOnly,
                            PixelFormat.Format24bppRgb);

unsafe {
  byte* lnptr = (byte *)bd.Scan0;
  byte* ptr;
  int w = 0;
  for (int i = 0; i < nb.Height; i++) {
    ptr = lnptr;
    for (int j = 0; j < nb.Width; j++) {
      if (positive) {
        *ptr++ = outputz[w] > 0.0f ? (byte)(outputz[w] * 255.0) : (byte)0;
        *ptr++ = outputy[w] > 0.0f ? (byte)(outputy[w] * 255.0) : (byte)0;
        *ptr++ = outputx[w] > 0.0f ? (byte)(outputx[w] * 255.0) : (byte)0;
        w++;
      }
```

```
      else {
        *ptr++ = outputz[w] < 0.0f ? (byte)(-outputz[w] * 255.0) : (byte)0;
        *ptr++ = outputy[w] < 0.0f ? (byte)(-outputy[w] * 255.0) : (byte)0;
        *ptr++ = outputx[w] < 0.0f ? (byte)(-outputx[w] * 255.0) : (byte)0;
        w++;
      }
    }
    lnptr += bd.Stride;
  }
}
nb.UnlockBits(bd);
```

Nevýhodou takovéhoto kódu je jeho ladění. Nastane-li zde chyba, samotná aplikace nemusí reagovat běžným výpisem chyby, v tomto případě je chování aplikace nepředvidatelné. Proto je nutné zvážit každý příkaz, napsaný v tomto bloku kódu. Typickou chybou je nastavení uzamčení obrazu „pouze pro čtení“ a následný zápis do proměnné `ptr`. Přiložený kód byl samozřejmě pro předejití takovýmto chybám několikrát kontrolován.

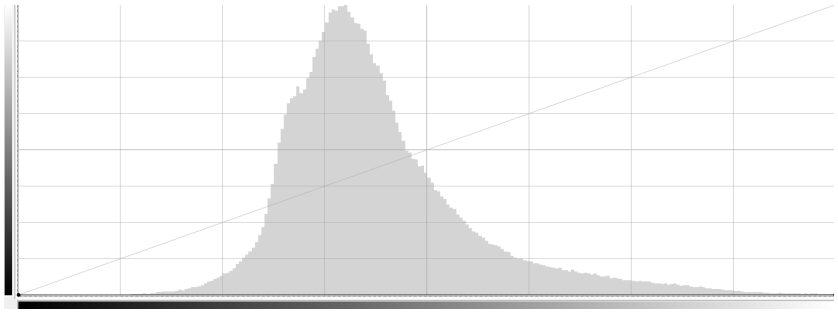
Důležitější je ovšem fakt, že díky těmto blokům `unsafe` program pracuje značně rychleji než při použití přístupu per-pixel a stále není nutné se přiklánět k externím knihovnám.

Pro veškeré výpočty nepracujeme s polem celočíselných hodnot. Ihned po načtení jsou obrazová data převedena do hodnot s plovoucí řádovou čárkou – `float`. Každá hodnota interpretuje jas, kdy 0 značí žádné nasvícení a 1 plné nasvícení. Zpětný převod do obrazu je prováděn pouze pro účely zobrazení výsledků uživateli.

3.2.1 Spektrální analýza vstupních dat

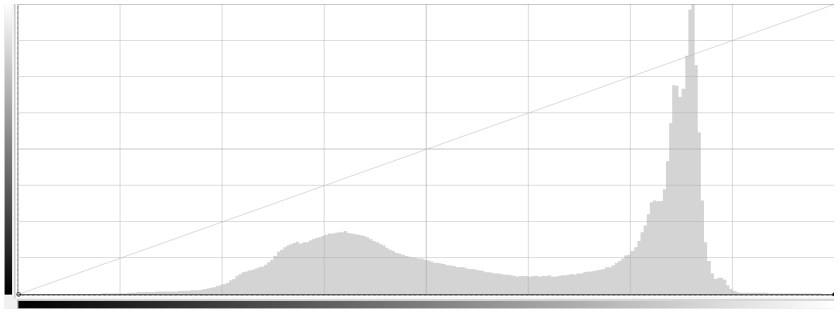
Stále ještě před samotným výpočtem normálové mapy je vhodné zajistit co možná největší rozsah vstupních hodnot. Můžeme předpokládat, že skener nepředá žádnou hodnotu blížící se minimu a maximu, tedy hodnoty blízké 0 a 1. V aplikaci tedy byla implementována volba přemapování původních hodnot tak, aby byl využit maximální rozsah. Spodní hodnota byla stanovena jako $0,01$ pro případ, kdy by mohlo díky malým hodnotám dojít k nestabilitě výpočtů.

Jako příklad můžeme předpokládat dva případy. Jednak výřez mince a pak také celou minci včetně podkladu. V prvním případě, obrázek 3.1, není využita celá spodní část spektra a částečně i horní kus. Přemapování tedy může přinést lepší výsledky.



Obrázek 3.1: Histogram výřezu obrazu naskenované dvacetikoruny, bez okrajů.

V druhém případě (obrázek 3.2) se jedná o snímek s „bílými“ okraji. Mince byla při skenování podložena na její spodní části (vzadu z pohledu snímače) bílou podložkou. Dalo by se tedy čekat, že v uvedeném histogramu bude i největší hodnota, kterou je skener schopen zaznamenat. I přesto lze i zde interval přemapovat na nové hodnoty a zvýšit tím potenciálně přesnost výpočtu.



Obrázek 3.2: Histogram celého obrazu dvacetikoruny, včetně okrajů.

Při srovnání výsledků skenovaných mincí nebyl po rekonstrukci znatelný rozdíl. Nelze však vyloučit případy, kdy právě tato volba v aplikaci zvýší citlivost a napomůže lepším výsledkům.

3.3 Implementace výpočtu normálového pole

Jako výstupní datový formát byla použita tři pole typu float, každé z nich pro jednu složku výsledné normály ve všech místech naskenovaného snímku. Tento formát byl vybrán záměrně ze dvou důvodů. Jednak je používán jako plovoucí datový typ na grafických kartách a v případě budoucí paralelizace na GPU[6] by již nebylo nutné dalšího převodu a jako druhý důvod je objem paměti zabrané pro výpočty. Pro obraz s rozměry $1\,024 \times 1\,024\ px$ (objekt o přibližné velikosti 1 až 2 palce) tato tři pole spotřebují 12 MB paměti, což je stále únosné a lze tedy pracovat i s většími objekty.

Výpočet normálového pole je, jakkoli je odvození vzorce složité, poměrně implementačně jednoduchý a lze jej zapsat následovně:

```
int x = 0;
double twoTanAm1 = 1.0 / (2.0 * Math.Tan(alpha));
for (int i = 0; i < sourceImages[0].height; ++i) {
  for (int j = 0; j < sourceImages[0].width; ++j) {
    outputx[x] = (float)(twoTanAm1 * (sourceImages[3].bmp[x] -
        sourceImages[1].bmp[x]));
    outputy[x] = (float)(twoTanAm1 * (sourceImages[0].bmp[x] -
        sourceImages[2].bmp[x]));
    outputz[x] = (float)(0.25 * (sourceImages[0].bmp[x] +
        sourceImages[2].bmp[x] + sourceImages[1].bmp[x] +
        sourceImages[3].bmp[x]));
    ++x;
  }
}
```

Celý kus kódu provede totéž, co je napsáno v kapitole 2.1.5 na straně 8 odpovídajícími vzorci. Úhel α pro použitý skener je $\frac{\pi}{6}$.

3.4 Integrace normálového pole

Právě tato část programu i celkové práce byla na jedné straně přínosná a na druhé straně také přinesla poměrně velké množství skrytých omezení, o

[6] Graphical Processing Unit

kterých se bez implementace není snadné nedozvědět. Ta budou uvedena v posledním pododdílu této části. Nejprve tedy k výběru okrajových podmínek.

3.4.1 Výběr okrajových podmínek

Pro matematika je fráze „Nechť jsou dány okrajové podmínky. . . “ prakticky samozřejmá, nicméně zde bylo nutné se zabývat tím, jak zvolit ty nejvhodnější okrajové podmínky tak, aby aplikace byla stále uživatelsky přívětivá a aby rekonstrukce jednoho předmětu dopadla při stejných parametrech opakovaně vždy stejně. Nabízejí se dvě alternativy. Buď nechat uživatele stanovit hodnoty celého okraje snímku nebo nechat stanovit pouze hodnoty v rozích a hodnoty celého okraje dopočítat. Jelikož rozlišení snímku nelze přímo zjistit, bylo vhodnější se přiklonit k variantě, kdy jedna jednotka je rovna jednomu obrazovému bodu. Uživatel si tedy musí spočíst, kolik bodů je tedy řekněme jeden milimetr na původním objektu a to z rozlišení, které sám nastavil.

Ruční určení okrajových podmínek

Prvně musíme zmínit, v jakých rozměrech se uživatel pohybuje. Pro průměrný případ, kdy rozlišení skeneru (symetricky v obou směrech) je 600 bodů na palec, se jeden milimetr rovná přibližně 24 bodům.

Předpokládejme, že uživatel má již naskenovaný snímek, například výřez mince, a ví, že spodní okraj raznice (nejhlubší místo na minci) se nachází v hloubce nula. Nyní potřebuje tuto informaci sdělit programu. Problém však je, že okraj výřezu nemusí vždy být celý na úrovni nulové hloubky. Implementováno tedy bylo rozdělení okraje na více částí, kdy ke každé části je možné přiřadit patřičnou hloubkovou informaci.

Takovýto okraj samozřejmě nebude hladký, nicméně použitý výpočet pomocí centrální diference hodnoty nacházející se dále od okraje vyhladí a povrch bude hladší. Navíc je doporučeno zvolit pro výpočet větší okraj tak, aby bylo možné výsledný objekt snáze oříznout.

Poloautomatické určení okrajových podmínek

V tomto případě uživatel určí pouze hloubkovou informaci v rozích a hodnoty celého okraje se dopočítají pomocí řádkové, respektive sloupcové, integrační metody – viz dále. Takto spočítané hodnoty jsou rozhodně hladší než v předchozím případě a dalo by se předpokládat, že i výsledný povrch bude přesnější. Toto tvrzení se nepotvrdilo, výsledky obou možností zadání okrajů podávají srovnatelné výsledky.

3.4.2 Navrhované metody rekonstrukce

Celkem byly brány v potaz tři metody rekonstrukce povrchu z normálového pole, které jsem již zmínil v teoretické části. První (RBF) byla zavržena pro její výpočetní náročnost a dále tedy ne byla uvažována. Zbylé dvě byly před praktickým použitím otestovány na analytických datech. Bylo nutné vybrat takové hodnoty, ze kterých lze vypočíst absolutní chybu výpočtu. Abychom mohli zvolit tu nejvhodnější variantu k implementaci do programu, bylo použito prostředí *Octave*[7]. To obsahuje optimalizovaný řešitel soustav lineárních rovnic, který by jinak muselo být implementováno. Navíc umí toto prostředí vybírat v závislosti na soustavě vhodnou metodu řešení.

Úloha prvního řádu

Za pomoci prostředí *Octave* bylo otestováno řešení soustavy rovnic se dvěma okrajovými podmínkami, odvozené z centrální diference (oddíl 2.2.3). Vstupní „povrchy“, byly funkce s předvidatelným chováním. Konkrétně byly testovány polynomy, e^x a sinusovky, respektive jejich kombinace. Výsledky byly překvapivě přesné a absolutní chyba byla na hranici strojové přesnosti. Byly brány v potaz polynomy různých stupňů i sinusovky s různou periodou. Výsledky byly pokaždé uspokojivé, a proto byla tedy tato metoda ustanovena jako primární pro implementaci.

Abychom se ujistili, jak velké absolutní chyby lze takto dosáhnout, byla provedena jednoduchá měření, kdy byly srovnány hodnoty analyticky zadané funkce a hodnoty vypočtené pomocí této metody. Funkce byly voleny tak, aby odpovídaly případným tvarům průřezu objektů, které lze do skeneru vložit.

[7] Otevřená varianta Matlabu.

Dokonce ani u prudce oscilujících sinusovek, jejichž charakteristika se blíží hranici Nyquistova teorému[8], se pro 100 vzorků absolutní chyba nedostala přes $10^{-}14$. Na druhou stranu je nutné brát v potaz, že vstupní data můžou být velmi různorodá a rozhodně většího rozsahu. Po převodu zpět do datového typu `float` je však tato chyba stále menší než oficiálních[9] 7 desetinných míst.

Úloha druhého řádu

Pro úlohu druhého řádu byl postup prakticky stejný jako v předchozím případě, takže netřeba detailně popisovat stejný postup. Praktickým rozdílem byl pouze počet okrajových podmínek. Při použití centrální diference potřebujeme čtyři okrajové podmínky, což není problém, pokud je průběh vstupních dat předem znám (analyticky zadaný předpis funkce), ovšem implementačně by bylo nutné, aby okraj skenovaného předmětu byl vždy zcela plochý, bez rýh či škrábanců. Především proto, že v porovnání s předchozí metodou byly výsledky této varianty srovnatelné až téměř identické, bylo stanoveno, že nebude nutné implementovat do přiloženého programu obě možnosti.

3.4.3 Omezení pro výpočet výsledného povrchu

Některá omezení byla ihned zřejmá, jiná se projevila až v průběhu měření. Uveďme tedy jejich seznam s komentáři.

Zkreslení obrazu byla předpokládána různá, jak bylo již dříve zmíněno. Pro mírná nelineární zkreslení by vyplývalo omezení se pouze na malé předměty, na kterých by zkreslení nebylo patrné. Jelikož se však neprojevila, je možné skenovat předměty po celém snímaném povrchu skeneru bez omezení. Pro lineární zkreslení (roztáhnutí) se provede korekce obrazu odpovídajícím způsobem.

Hloubka předmětu je dána již v zadání jako velmi malá, je však na místě uvést konkrétní hodnoty. Byly skenovány předměty s hloubkou až do pěti milimetrů. Omezení je dáno dvěma hlavními faktory. Jednak klesající intenzitou

[8] Vzorkovací frekvence $> 2\times$ nejvyšší frekvence signálu.
[9] Dle standardu IEEE 754.

osvětlení se zvětšující se vzdáleností od zdroje – to by mohlo ovlivnit výpočet normálového pole. Druhým faktorem je rozostření či rozmazání obrazu v závislosti na vzdálenosti od skla skeneru. Optická soustava má obecně velmi malou hloubku ostrosti a ve vzdálenosti větší než 5 *mm* od skla skeneru je již obraz značně rozmazaný. Optimální maximální hloubka se tedy pohybuje okolo tří milimetrů.[10]

Tvar předmětu může být libovolný. Až do fáze výpočtu normálového pole (včetně) nezáleží ani na tvaru a ni na velikosti skenovaného předmětu. Při rekonstrukci povrchu byla nicméně odhalena jistá omezení. Metoda rekonstrukce totiž předpokládá, že normálové pole z předchozího výpočtu odpovídá přesně skutečnosti. Problém však nastane v případě, kdy ostrá hrana na předmětu vytvoří stín. Typicky jsou tyto stíny přítomny na okrajích, kdy je stín vržen na podložku a pro rekonstrukci je předložen obraz předmětu i s „bílým okrajem“. V místě, kde se nachází stín se vypočtou nekorektní normály a při výpočtu povrchu pak dochází k velkým chybám, i když vizuálně je lze místy přehlédnout. Nejvhodnějším předmětem co do tvaru je tedy obdélníkový předmět s plochým okrajem. Samozřejmě problém nenastane ani u libovolného obdélníkového výřezu.

Rekonstrukce mincí funguje dobře, ovšem je nutné brát v potaz, že korektní výstup je pouze v místech neovlivněných stíny, tedy primárně od středu před okraj mince.

3.5 Použitá metoda rekonstrukce

V závislosti na předchozích pokusech a měřeních bylo rozhodnuto do programu implementovat řešení pomocí soustavy rovnic prvního řádu, vycházející z definice první centrální diference. Jednotka byla stanovena jako jeden obrazový bod, což v průměrném skeneru odpovídá přibližně $\frac{1}{24}$ mm. Jednotka je stejná pro všechny směry.

Výhodou použité metody je fakt, že vstupní matice je vždy genericky stejná a vždy ji lze sestavit pro libovolně velký obraz.

Nabízí se dva možné přístupy. První, řádkový přístup, sestaví pro snímky

[10]Samozřejmě zde hraje roli kvalita skeneru a k dispozici byl pouze běžný, neprofesionální stolní skener.

o velikosti $W \times H$ pro každý řádek různou soustavu rovnic. Druhý přístup sestaví „globální“ soustavu rovnic pro všechny hodnoty vstupu. V přiložené aplikaci jsou přístupné obě varianty, aby bylo možné porovnat výsledky.

3.5.1 Řádkový přístup

Soustava rovnic pro tuto možnost vychází přímo z vztahu z oddílu 2.2.3 na straně 14. Matice pro celý obraz o rozměrech $W \times H$ tedy vypadá následovně:

$$\begin{bmatrix} \mathcal{R}_0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \mathcal{R}_1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \mathcal{R}_2 & 0 & \cdots & 0 \\ \vdots & & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \mathcal{R}_{H-1} \end{bmatrix} . \begin{bmatrix} f(0,0) \\ f(0,1) \\ f(0,2) \\ \vdots \\ f(H-1, W-1) \end{bmatrix} = \begin{bmatrix} \mathcal{B}_{r[0]} \\ \mathcal{B}_{r[1]} \\ \mathcal{B}_{r[2]} \\ \vdots \\ \mathcal{B}_{r[H-1]} \end{bmatrix} \tag{3.1}$$

Hodnoty $\mathcal{R}_i$ na diagonále jsou pak jednotlivé matice soustav pro jednotlivé řádky vstupních dat, identické se vztahem v oddílu 2.2.3 na straně 14 a hodnoty $\mathcal{B}_i$ jsou odpovídající vektory pravých stran. Řešit je možno soustavu jako celek nebo separátně, každou podsoustavu zvlášť. Výsledek je identický a jeho vizualizaci lze najít ve výsledcích na straně 34.

3.5.2 Plošný přístup

V tuto chvíli je vhodné znovu zformulovat úlohu. Na vstupu máme normálové pole, tedy v každém bodě známe gradient – vektor parciálních derivací. Pro každý bod tedy máme dva vztahy – řádkový a sloupcový:

$$f(x_{i,j+1}) - f(x_{i,j-1}) = 2 \cdot \frac{\mathrm{d}f}{\mathrm{d}x}, \tag{3.2}$$

$$f(x_{i+1,j}) - f(x_{i-1,j}) = 2 \cdot \frac{\mathrm{d}f}{\mathrm{d}y}. \tag{3.3}$$

Vzhledem ke schopnostem použité výpočetní knihovny (níže) byla sestavena přeurčená soustava rovnic a řešena ve smyslu metody nejmenších čtverců.

$$\left[\begin{array}{cccccc} \mathcal{I} & 0 & 0 & 0 & \cdots & 0 \\ 0 & \mathcal{R}_1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \mathcal{R}_2 & 0 & \cdots & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & \mathcal{R}_{H-2} & 0 \\ 0 & 0 & 0 & 0 & \cdots & \mathcal{I} \\ \hline -\mathcal{I} & \mathcal{C}_1 & \mathcal{I} & 0 & \cdots & 0 \\ 0 & -\mathcal{I} & \mathcal{C}_2 & \mathcal{I} & \cdots & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & \cdots & -\mathcal{I} & \mathcal{C}_{W-2} & \mathcal{I} \end{array}\right] \cdot \left[\begin{array}{c} f(0,0) \\ f(0,1) \\ f(0,2) \\ \vdots \\ \vdots \\ f(H-1,W-1) \\ \hline f(1,0) \\ f(1,1) \\ \vdots \\ f(H-2,W-1) \end{array}\right] = \left[\begin{array}{c} \mathcal{B}_{r[0]} \\ \mathcal{B}_{r[1]} \\ \mathcal{B}_{r[2]} \\ \vdots \\ \vdots \\ \mathcal{B}_{r[H-1]} \\ \hline \mathcal{B}_{c[1]} \\ \mathcal{B}_{c[2]} \\ \vdots \\ \mathcal{B}_{c[H-2]} \end{array}\right] \tag{3.4}$$

Soustava je znázorněna za pomoci submatic. $\mathcal{I}$ je jednotková matice a ve spodní části celé soustavy znázorňuje vedlejší diagonály, mající význam vždy „hodnota nad, respektive pod“. Hodnoty $\mathcal{B}_{r[i]}$ a $\mathcal{B}_{c[i]}$ jsou vektory:

$$\mathcal{B}_{r[i]} = [f(i,0), 2 \cdot f'_x(i,1), \cdots, 2 \cdot f'_x(i,W-2), f_x(i,W-1)]^T, \tag{3.5}$$

$$\mathcal{B}_{c[i]} = [f(0,i), 2 \cdot f'_y(1,i), \cdots, 2 \cdot f'_y(W-2,i), f_y(W-1,i)]^T. \tag{3.6}$$

Oba mají rozměr W a první a poslední hodnota je vždy okrajová. Matice $\mathcal{R}_i$ je stejná jako v předchozím případě a má tvar:

$$\left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & \cdots & 0 \\ -1 & 0 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 0 & 1 & \cdots & 0 \\ \vdots & & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{array}\right] \tag{3.7}$$

Matice $\mathcal{C}_i$ je v tomto případě (první centrální diference) nulová. Hodnoty -1 a 1 jsou posunuty o W doleva a doprava a tvoří již zmíněné vedlejší diagonály. S hodnotou vprostřed (na diagonále) se nepočítá, proto je právě ttato matice prázdná – nulová.

3.5.3 Výpočetní knihovna

Při výběru vhodné výpočetní knihovny bylo postupováno dle osvědčených požadavků.

- Musí být stále vyvíjena,
- musí existovat kontakt na autora / autory,
- musí existovat dostatečná dokumentace,
- knihovna je psána ve stejném jazyce jako implementovaný program.

Dále byl přidán požadavek na práci s řídkými maticemi. Po poměrně krátkém hledání byla zvolena jako nejvhodnější knihovna `Alglib`. Práce s ní je jednoduchá, existuje v nekomerční i komerční variantě a výpočty jsou pro tyto účely dostatečně rychlé.

Řídkou matici je nejprve nutné deklarovat:

```
alglib.sparsematrix m;
```

Ještě před jejím vytvořením potřebujeme znát její přesný tvar, tedy rozměry a počet nenulových hodnot na každém řádku. Matice se vytvoří následovně:

```
alglib.sparsecreatecrs(výška, šířka, popis_řádků, out matice);
```

První parametr udává počet řádků matice, druhý počet sloupců. Třetí parametr je popis s počty nenulových hodnot na každém řádku a poslední parametr je výstupní, sem je předána reference na vytvořenou matici. Samotné naplnění je prováděno příkazem `alglib.sparseset(matice, i, j, hodnota);` .

Vektor pravých stran je jednorozměrné pole typu `double[]`, předané při spuštění výpočtu. Výpočet samotný je již jednoduchý:

```
alglib.linlsqrcreate(výška, šířka, out lsstate);
alglib.linlsqrsetcond(lsstate, epsilon_A, epsilon_B, max_iterací);
alglib.linlsqrsolvesparse(lsstate, matice, pravá_strana);
alglib.linlsqrreport report;
alglib.linlsqrresults(lsstate, out výsledek, out report);
```

V paměti je vytvořen prostor pro popis výpočetní metody a její parametry, zde soustava řešená ve smyslu nejmenších čtverců. Dále jsou nastaveny zastavovací podmínky, tedy maximální velikost absolutní chyby a maximální počet iterací. Algoritmus se zastaví v případě, že je splněna libovolná podmínka. Výsledek je předán opět parametrem jako pole `double[]`.

Po ukončení práce s maticí je nutné uvolnit paměť, kterou zabírala, voláním `alglib.sparsefree(matice);`.

3.6 Vizualizace výsledků

3.6.1 Normálové pole

Aplikací vypočtené výsledky je možno vizualizovat několika způsoby. Normálová mapa je primárně zobrazena po složkách jako obraz, kdy záporné hodnoty jsou znázorněny jako hodnoty 0 až 0, 5 a kladné jako 0, 5 až 1. Praktická ukázka je ve výsledcích. K lepšímu zjištění skutečného průběhu v řezu je implementován i *slicer*, který zobrazuje pro přehlednost vždy jen průměrný směr normály z určitého okolí.

Při exportování dat do souboru jsou uloženy i vypočtené normály a přiřazeny odpovídajícím vrcholům prostorového modelu. Výsledné pole je tedy možné zobrazit i v externích aplikacích.

3.6.2 Výšková mapa

I výškovou mapu lze zobrazit třemi způsoby. Obraz zobrazený v náhledu po dokončení integrace je nejjednodušší možno vizualizací. Tmavší odstín šedé vyjadřuje plochy blízké minimu a světlý odstín značí vystouplé plochy, tedy blíže ke snímači skeneru. Druhou možností je zobrazení průběhu v řezu,

v aplikaci je i možné zobrazit zároveň průřez a k němu náležící normály. Finální možností je export integrovaného modelu do formátu Wavefront .OBJ společnosti *Wavefront Technologies*[10]. Model je pak dále možné upravovat v jiném softwaru.

4 Metody 3D tisku

Aby byl celý proces ucelen, je nutné se zmínit o metodách 3D tisku, jelikož výsledky této práce byly vytisknuty na 3D tiskárně tak, jaky bylo předesláno v úvodu. Informace k této kapitole byly čerpány z [12].

Vývoj 3D tisku lze datovat již od druhé poloviny 20. století. Konkrétně roku 1986 byla představena metoda *Stereolitografie*. Jedná se o techniku využívající tuhnutí určitého polymeru pod ultrafialovými paprsky. Z této metody vycházejí další, využívající podobných principů.

Mezi další patří metoda zvaná *Fused Deposition Modeling (FDM)*. Používá tavný plast v podobě plastových „drátů“. Tento materiál je roztaven v hlavě tiskárny a ve vrstvách nanášen na podložku, čímž vzniká model. Výhodou této metody jsou nízké náklady na samotný tisk. Nevýhodou je poměrně velká hrubost výsledného modelu.

Vybraná metoda pro tisk výsledků této práce se nazývá *Three Dimensional Printing (3DP)* a používá velmi jemný prášek, spojovaný po vrstvách speciálním lepidlem. Bylo použito zařízení *ZPrinter® 650*.

Mezi další lze zařadit například *Ballistic particle Manufacturing* – model je sestavován „po jednotlivých voxelech“ z roztaveného materiálu. Tato metoda je však časově velmi zdlouhavá. Zmiňme ještě metodu společnosti *Sanders Inc.*, kdy je model vytvářen po vrstvách z materiálu podobnému vosku. Nevýhodou této metody je nemožnost přímého tisku modelu, obsahujícího části nepodepřené v ose z, typicky například ruce modelu člověka.

Hlavním důvodem výběru metody tisku jemným práškem, spojeným lepidlem byla dostupnost tiskárny. Modely byly tisknuty s jemností $0,8\ mm$ a jejich kvalita je uspokojivá.

5 Výsledky

V průběhu celé práce byla provedena mnohá měření ať již přesnosti či rychlosti výpočtu. Budou uvedeny však pouze ty výsledky, u kterých můžeme předpokládat, že jsou přínosné či jiným způsobem zajímavé. V grafech jsou zobrazeny převážně výřezy z datových množin, aby bylo dosaženo jejich lepší čitelnosti.

5.1 Měření přesnosti metod

Nelze uvést přesnou hodnotu maximální odchylky pro obecná vstupní data, budou zde tedy uvedeni pouze zástupci očekávaných dat. Pro kompletnost výsledků by byl zapotřebí přesný 3D model objektu, který by bylo možné naskenovat a rekonstruovat, nicméně takový model nebyl při zpracování této práce k dispozici. Potvrzení analýzy metod bylo empiricky ověřeno vždy na analyticky zadaných funkcích, pro které je předem známa její přesná derivace.

5.1.1 Soustava prvního řádu

Při použití soustavy rovnic prvního řádu bylo dosaženo velmi příznivých výsledků. Předem je nutno podotknout, že vstupní data jsou z jistého pohledu optimální, jelikož na rozdíl od reálných dat je ve všech místech vždy známa přesná hodnota derivace/normály.

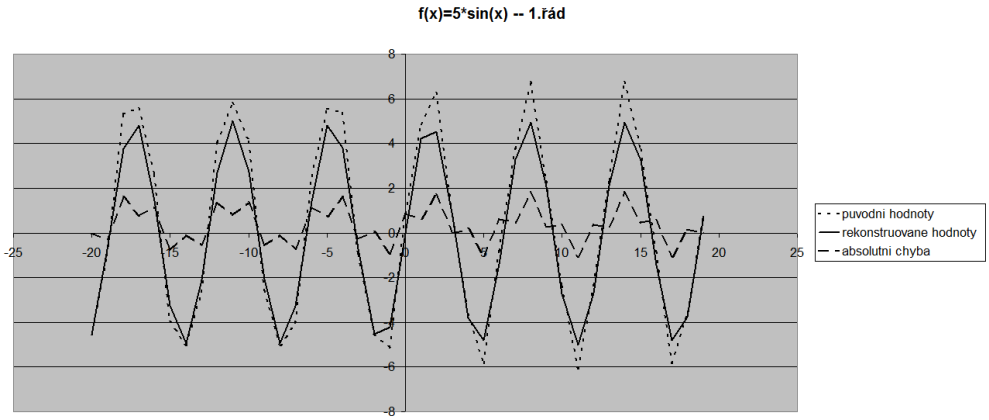
Výsledky pro graf zobrazující rekonstrukci funkce x^2 není vhodné vykreslit do grafu, hodnoty by se totiž zcela překrývaly. I při použití několika set vstupních hodnot (po diskretizaci) byla naměřená chyba menší než strojová přesnost počítače[1] a hodnoty vycházely zcela přesně, tedy s nulovou chybou. Zvolené množství vstupních hodnot odpovídá přibližně rozměrům mince.

Toto ovšem nebylo přesvědčivým důkazem o absolutní přesnosti a bylo nutné najít taková vstupní data, pro která by byla chyba největší. Jelikož se vždy jedná o vzorkovaná data, bylo nasnadě použít na vstup sinusovku, využít znalosti Nyquistova teorému a přiblížit se hranici vzorkovací frekvence.

[1] Dle IEEE 754 je přesnost datového typu `double` přibližně 10^{-16}

Výsledek je možno shlédnout v grafu 5.1. Pokud vezmeme v potaz, jakému tvaru povrchu by odpovídala tato funkce, pak jsou i tyto výsledky příznivé. Jedna jednotka je přibližně $\frac{1}{24}$ milimetru. Tato křivka tedy představuje povrch, na němž jsou na jednom milimetru čtyři zákmity a to pravidelné. I takovýto povrch však dle předpokladu nemusí být těžké nalézt, jemně vroubkovanou minci si lze představit. Přesto je výsledek uspokojivý.

Chyba dosahující dvou jednotek – tedy ‚velmi zhruba, téměř jedné desetiny milimetru – je přijatelná především proto, že osciluje kolem téměř nulové hodnoty a jelikož metoda používající centrální diferenci má vyhlazovací účinek, bude takovýto téměř šum vyhlazen. Jedná se o opravdu krajní, nejhorší případ a lze mu předejít i vhodnou konvoluční maskou v přípravné fázi.



Obrázek 5.1: *Metoda:* soustava rovnic, *Varianta:* prvního řádu, *Funkce pro vstupní data:* $\sin(x)$

5.1.2 Soustava druhého řádu

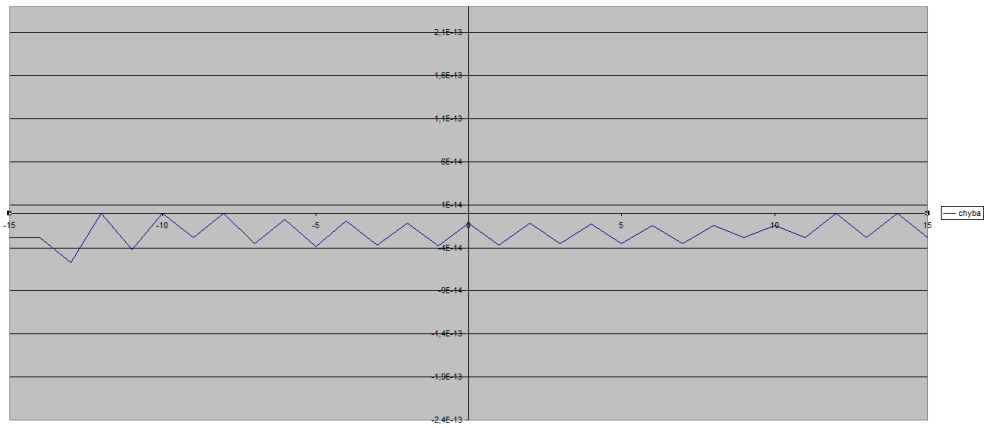
Soustava rovnic druhého řádu byla použita především pro potvrzení korektnosti předchozí metody, jelikož vstupní data jsou diference prvního řádu a hlubší význam použití této možnosti jako primární není. Byla tedy sledována spíše chyba při výpočtu. Grafy původních dat a rekonstruovaných dat se vizuálně překrývaly, ale absolutní chyba již nebyla rovna nule.

Na grafu 5.2 je znázorněna absolutní chyba při použití vstupních dat získaných vzorkováním funkce x^2. Je zřetelně vidět, že chyba neosciluje kolem nulové hodnoty a je jednoznačně záporná. Rekonstruované hodnoty jsou tedy v tomto případě níže než původní hodnoty. Na druhou stranu chyba v absolutní hodnotě se stále pohybuje v řádu 10^{-14}, což je stále přijatelné.

Pro otestování komplikovanějšího povrchu byla použita aplikace Wolfram Alpha[2], kde pro vyjádření komplikovanějšího povrchu byla vygenerována následující funkce:

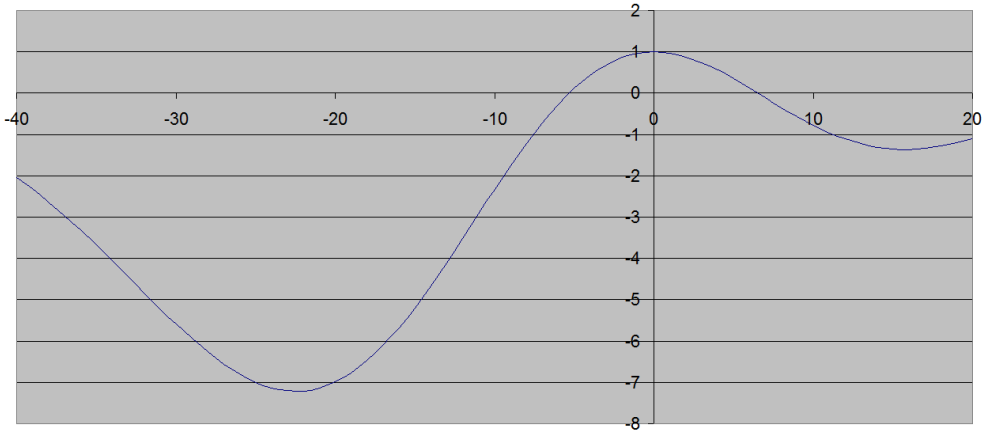
$$e^{-((0.05 \cdot x)^2)} \cdot (0.001x^3 - 0.03x^2 + 1) \tag{5.1}$$

Část jejího průběhu lze pro představu shlédnout v grafu 5.3. Velikost chyby při rekonstrukci je viditelná na grafu 5.4. I zde se chyba v absolutní hodnotě pohybuje v řádu 10^{-14} a lze tedy předpokládat, že pokud budou hodnoty normálového pole správné, bude i reprodukovaný povrch poměrně přesně odpovídat původnímu objektu.

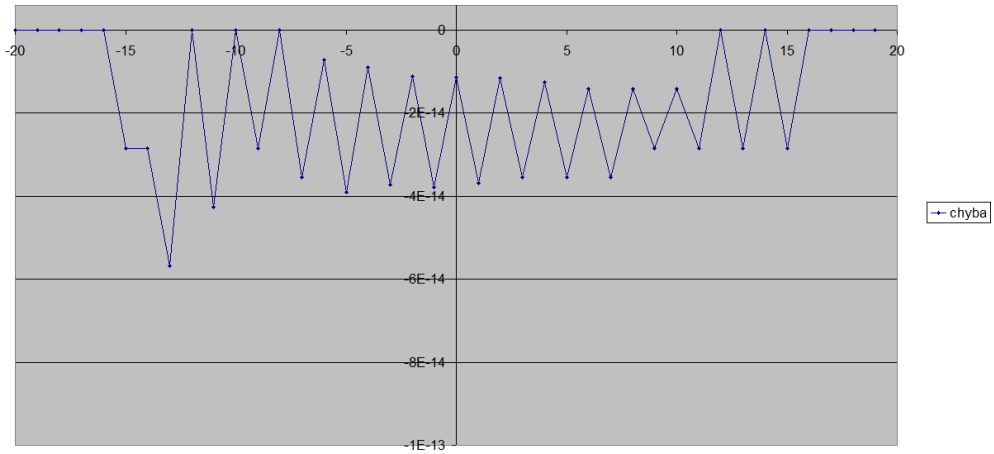


Obrázek 5.2: *Metoda:* soustava rovnic, *Varianta:* druhého řádu, *Funkce pro vstupní data:* x^2

[2]On-line projekt výpočetního softwaru společnosti Wolfram, dostupný na http://www.wolframalpha.com/ .



Obrázek 5.3: $e^{-((0.05 \cdot x)^2)} \cdot (0.001x^3 - 0.03x^2 + 1)$



Obrázek 5.4: $e^{-((0.05 \cdot x)^2)} \cdot (0.001x^3 - 0.03x^2 + 1)$

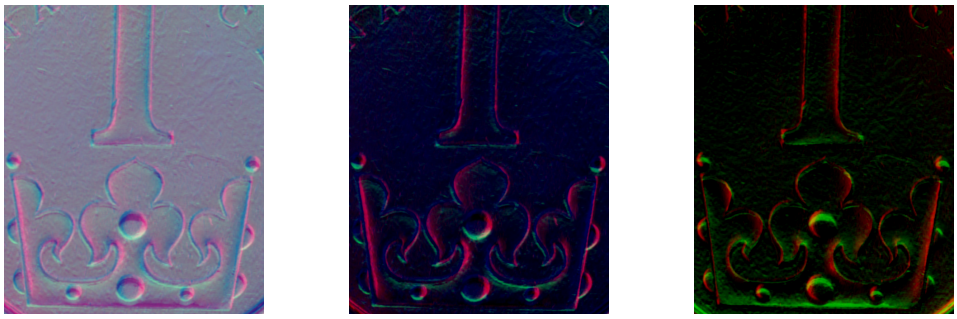
5.2 Normálové pole

V tomto bodě je znovu nutné uvést fakt, že k rekonstruovaným datům nebyly k dispozici skutečné hodnoty normál ani jiný popis povrchu.

Výsledky normálového pole vypadají na první pohled velmi přesvědčivě a dalo by se říci, že jsou téměř bezchybné. Po dalším zkoumání a především na základě výpočtu výsledného povrchu se však projevily i některé nedostatky. Na obrázku 5.5 je vlevo vizualizace vypočteného normálového pole detailu mince 1 Kč. Jednotlivé barevné složky postupně odpovídají složkám normály v daném bodě. Jelikož však normála může obsahovat i záporné hodnoty, je střední hodnota stanovena na 50% šedou. Uprostřed jsou zobrazeny pouze kladné hodnoty (záporné jsou nulové) a vpravo pak pouze záporné hodnoty (kladné jsou nulové). Převážně modrá barva na prostředním výřezu značí, jak se dalo očekávat, plochy kolmé ke sklu skeneru.

Na obrázku 5.7 ve výsledcích na straně 40 jsou zobrazeny normálové mapy čtyřech mincí. Je patrné, že výsledek výpočtu nezávisí na zbarvení ani lesku materiálu objektu, který chceme skenovat. Výsledek je vždy stejně kvalitní, alespoň pro mince, které byly v rámci této práce skenovány.

Byl objeven negativní limit celého procesu, který se projevuje právě již ve fázi výpočtu normál, avšak není patrný na první pohled. Na obrázku 5.5 se v levém a pravém dolním rohu a u kulatých výběžků před výpočtem na některých snímcích[3] vyskytoval stín a normály v těchto místech jsou tím pádem nesprávné. Na vizualizaci normálového pole se tento fakt neprojevuje příliš zřetelně, nicméně v následujícím oddílu bude zřejmý.



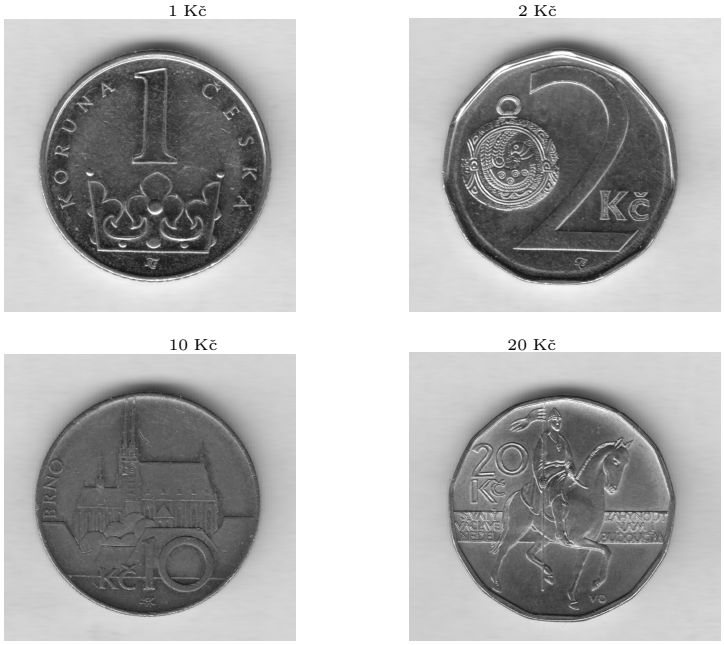
Obrázek 5.5: Detail koruny na minci jedné Koruny české.

[3]Záleží na natočení vůči zdroji světla.

5.3 Rekonstruovaný povrch

Pro snadné srovnání budou uvedeny vždy výsledky pro čtyři mince. Jejich snímky před zpracováním lze shlédnout v prvním pododdílu této sekce. Dále následují rekonstruované normálové mapy a na závěr obě varianty rekonstrukce povrchu. Řádkový přístup i plošný.

5.3.1 Vstupní data



Obrázek 5.6: Naskenované snímky mincí.

Pro znázornění byly vybrány čtyři české mince tak, aby obsahovaly různé druhy povrchů.

Na minci s hodnotou 1 Kč je vyražen zjednodušený obraz Svatováclavské koruny, obsahující uvnitř pozvolné, hladké přechody.

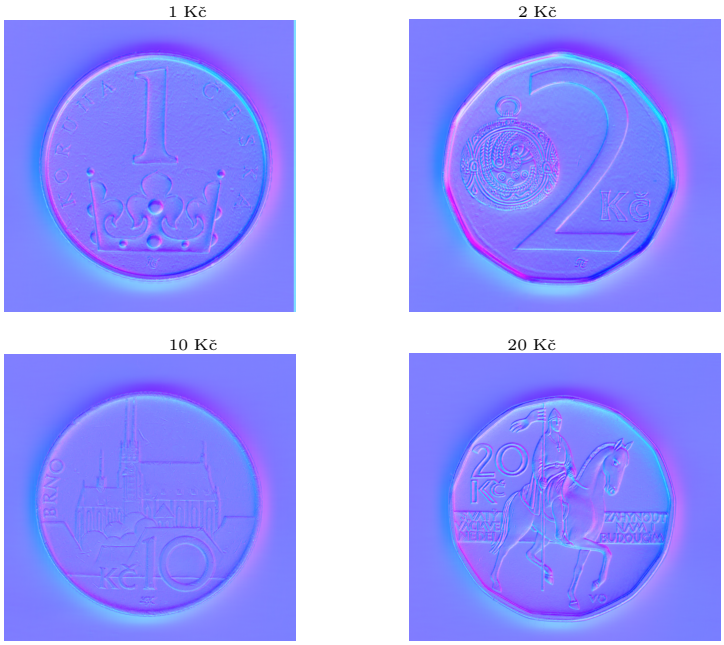
Dvoukoruna naopak obsahuje dvě rozdílné, rovné hladiny (vršek číslovky 2 a spodní plochu). Navíc obraz Gombíku z dob Velké Moravy poskytuje příklad povrchu s jemnými detaily.

Katedrála svatého Petra a Pavla znázorněná na desetikoruně je složena

z více vrstev, které se od sebe liší jen malou výškou. Je tedy ideální k určení rozlišovací schopnosti metody.

Podobizna svatého Václava na koni na dvacetikoruně spolu s nápisem a číslovkou tvoří komplikovaný různorodý povrch, který by měl prokázat použitelnost metody na prakticky libovolné mince.

5.3.2 Normálové mapy



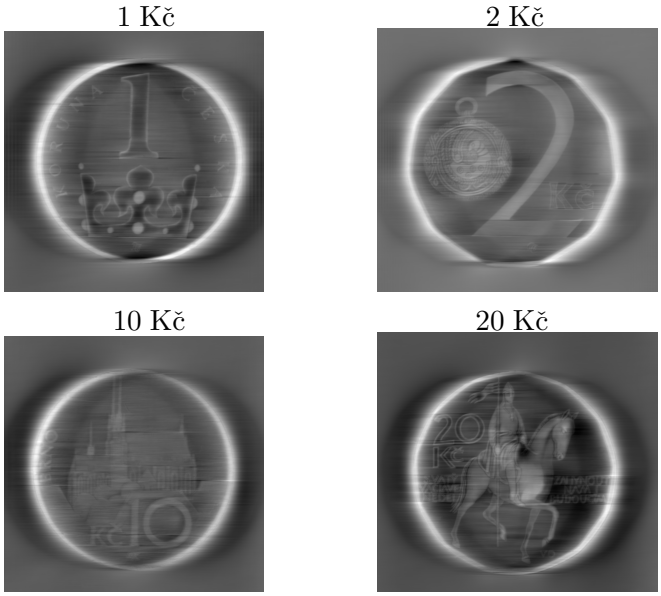
Obrázek 5.7: Vypočtené normálové mapy pro mince.

Tyto normálové mapy byly spočteny s roztažením intervalu vstupních jasů tak, aby byl využit celý interval 0 až 1 a s normálami normalizovanými na $nz = 1$.

Jak již bylo předesláno, lze si zde kolem okrajů všech mincí a kolem kulatých výstupků mince 1 Kč všimnout nekorektních normál.

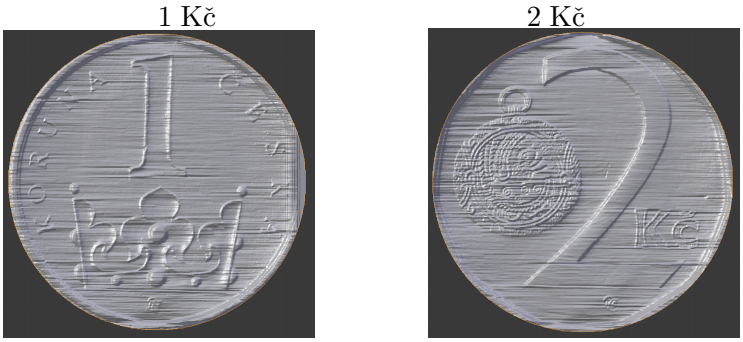
5.3.3 Řádková varianta integrace

Řádková integrace je prováděna separátně pro každý řádek. V případě bezchybných vstupních dat by zřejmě rozdíl oproti plošné integraci byl minimální, nicméně zde lze snadno pozorovat, že jelikož každý řádek obsahuje jinou kumulativní absolutní chybu, vytváří jednotlivé řádky jemné změny výšek ve vertikálním směru.



Obrázek 5.8: Vypočtené výškové mapy mincí po integraci. Světlejší hodnoty znamenají vyšší hodnoty.

Efekt vzniknuvší řádkovým přístupem je stejně zřetelný jak na výškové mapě na obrázku 5.8 tak na 3D modelu na obrázku 5.9.



Obrázek 5.9: Exportované modely mincí po integraci.

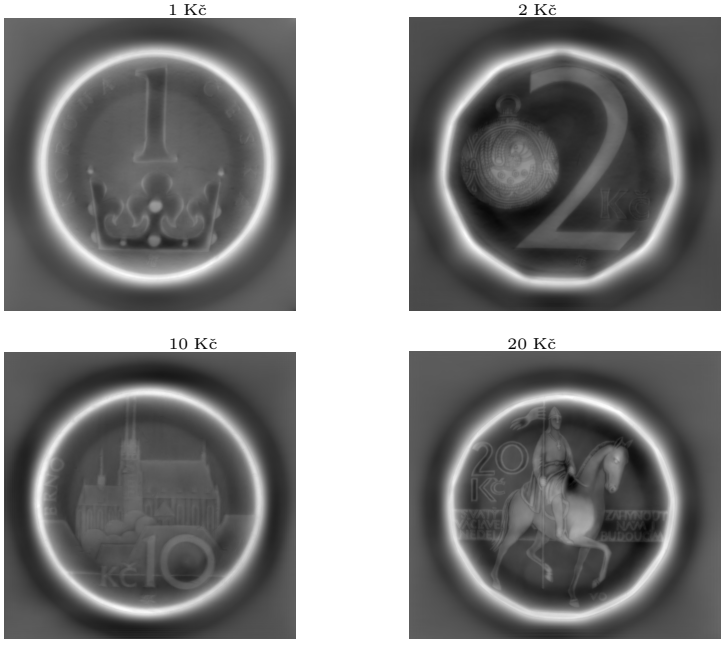
5.3.4 Plošná varianta integrace

Tato technika rekonstrukce minimalizuje chybu metodou nejmenších čtverců, tím pádem je patrné poměrně velké zlepšení oproti řádkovému přístupu. Samozřejmě pro korektní srovnání by byl nutný originální model mince či alespoň raznice. Ten však není veřejně přístupný. I přesto lze odhalit slabá místa rekonstrukce.

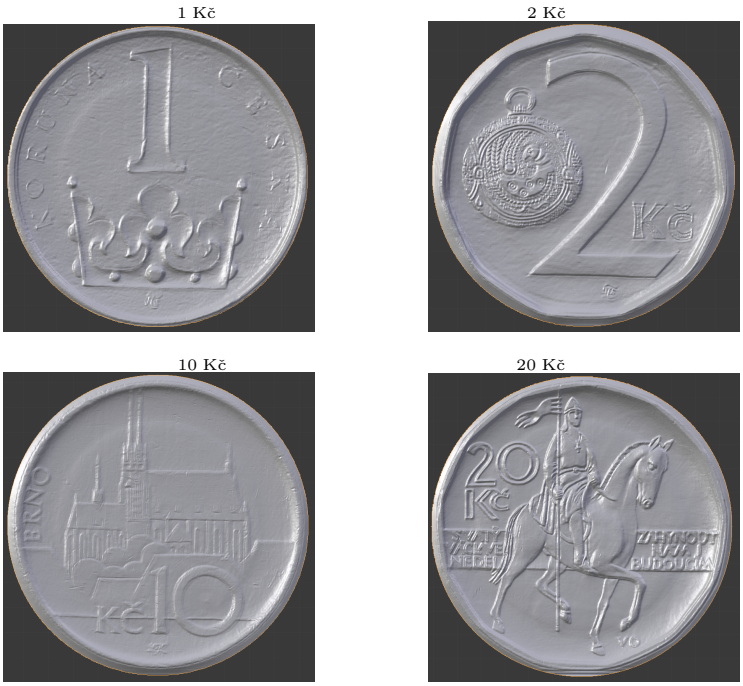
Na výškových mapách můžeme především na dvacetikoruně pozorovat tmavý pruh okolo okraje. Ten je způsoben snahou minimalizovat nekorektní normály v místě, kde se nacházel stín.

Na obrázcích modelů mincí (po rekonstrukci) jsou rovněž patrné hladké přechody v místech, kde má být ostrá hrana, tedy opět v místech, kde se nacházel stín.

I při detailnějším pohledu se dá konstatovat, že výsledky jsou uspokojivé.



Obrázek 5.10: Vypočtené výškové mapy mincí po integraci. Světlejší hodnoty znamenají vyšší hodnoty.



Obrázek 5.11: Exportované modely mincí po integraci.

6 Závěr

Cílem této práce bylo prostudovat dosavadní metody věnující se tématu rekonstrukce pomocí skeneru, následně ověřit správnou funkčnost algoritmu pro výpočet normálového pole a vybrat či vymyslet a ověřit integrační metodu pro získání 3D povrchu.

První fáze této práce tedy byla věnována hledání podobných metod. Proces detekce a opravy poškozených fotografií poskytl základní princip, ze kterého vychází metoda získání normálového pole. Technika pro výpočet povrchu z naskenovaných snímků nalezena nebyla, ani jí podobná.

V žádném z článků nebyly zmíněny problémy se stíny, vrženými ostrými hranami, nicméně u téměř plochých objektů nebylo nutné takovéto případy uvažovat a hledat další specializovaný algoritmus pro jejich detekci odstranění.

V následující fázi byla vytvořena aplikace, kterou se ověřila správnost a korektnost metody rekonstrukce normálového pole. To se podařilo úspěšně a následovala nejtěžší část práce, kterou bylo nalezení vhodného postupu pro integraci normálového pole tak, aby vznikla výšková mapa, respektive prostorový model. Po zavrhnutí metody využívající radiální bázové funkce a po četných konzultacích nejen s odborníky na dané téma byl sestaven výpočetní postup a ten následně implementován.

Výsledky jsou uspokojivé a lze tvrdit, že práce byla úspěšná. Pokud to bude možné, bude k práci k dispozici i 3D model mince, vytisknutý na 3D tiskárně.

A Uživatelská dokumentace

Prerekvizity

Aplikaci je možné spustit v 32 bitovém režimu nebo 64 bitovém režimu. Před spuštěním aplikace je nutné se ujistit, že je na stroji instalováno běhové prostředí .NET – standardně bývá dodáváno již jako součást operačního systému nebo jako jeho aktualizace. Pro běh je zapotřebí verze 2.0 nebo vyšší.

Spuštění aplikace

Aplikace se spustí poklikáním na `SuRe.exe`, spuštěna je jednak konzole, kde můžou být zobrazena některá chybová hlášení, následně i samotné ovládací okno aplikace.

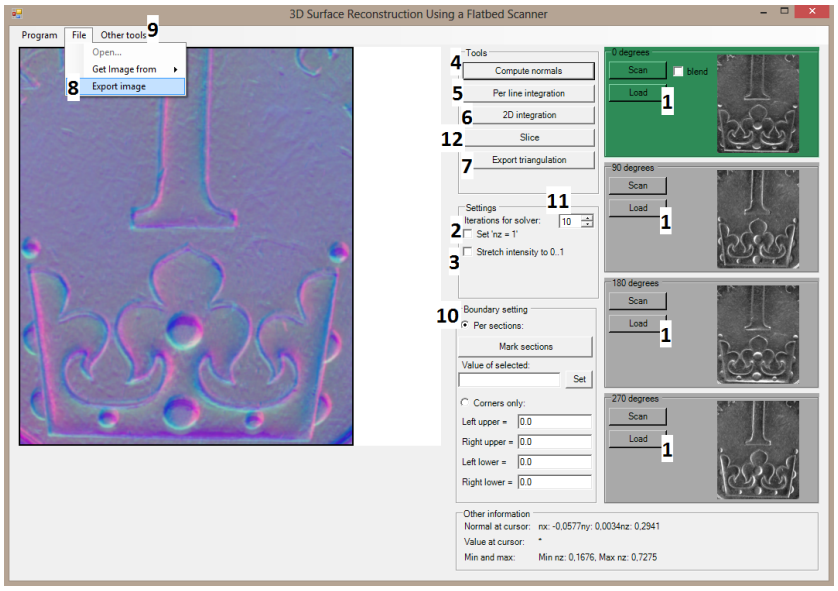
Ovládání

Načtení vstupních snímků se provádí vždy kliknutím na (1) u každého natočení zvlášť. Zobrazit snímky na plátně je možné kliknutím přímo na ně.

Výpočet normál se řídí dvěma volbami. (2) udává, zda se má *'nz'* složka zvolit jako rovná jedné, (3) pak způsobí využití plného intervalu 0 až 1 pro další výpočty z hodnot intenzit. Samotný výpočet započne stisknutím tlačítka (4).

Řádková integrace je možná až po vypočtení normálového pole. Její spuštění se provede stisknutím (5).

2D integrace (6) je rovněž možná až po vypočtení normálového pole. Je pravděpodobné, že výpočet bude trvat delší dobu, je tedy nutné počkat na



Obrázek A.1: Uživatelské rozhraní.

jeho dokončení. Výsledek v podobě výškové mapy je zobrazen na plátně.

Exportovat lze buď vypočtený model s normálami do formátu Wavefront OBJ (7) nebo libovolný výstup zobrazený na plátně (8).

Srovnání řádkové a plošné integrace se provede stisknutím (9), jsou provedeny obě integrace a výstup je vypsán v textové podobě do souboru. Na každém řádku je vždy rozdíl obou hodnot postupně po řádcích pro všechny body.

Okraj včetně typu zadání se nastavuje v (10), čísla je možno zadávat i jako desetinné hodnoty s desetinnou tečkou.

Nastavení počtu iterací při výpočtu soustavy se volí v (11).

Nástroj řezu (12) je zapínací a vypínací opakovaným stisknutím. Výběr řezu se provede kliknutím na plátno.

B Instalace programu

Jelikož společnost Microsoft přestala v použitém vývojovém prostředí[1] podporovat standardní instalátor a přenechala tuto možnost na externí společnosti, je aplikace dodávána jako samorozbalovací archiv a samozřejmě i samostatně.

Běhové prostředí není součástí, jelikož jeho distribuce by byla v rozporu s licenčními podmínkami. Je však standardně dostupné přes systém aktualizací OS Windows.

Jsou dodány vždy dvě verze, pro 32 bitové operační systémy **SuRe_setup_32bit.exe** a pro 64 bitové **SuRe_setup_64bit.exe** .

Dle lokálního systému tedy stačí vybrat správný samorozbalovací archiv, který se zeptá, kam aplikaci instalovat.

[1] Microsoft Visual Studio 2012

C Programátorská dokumentace

Program je dodáván jako projekt pro vývojové prostředí Microsoft Visual Studio 2012, ve kterém byl vyvíjen.

Seznam souborů a jejich popis

- **Border.cs** – Obsahuje metody spojené se zadáváním, výpočtem a vykreslováním okrajových podmínek, zadaných uživatelem.
- **MainForm.*** – Obsahuje zdrojový kód pro ovládání uživatelského rozhraní a obslužné procedury pro všechna tlačítka.
- **MyImage.cs** – Třída obsažená v tomto souboru zaobaluje objekt obrazu, reprezentovaný hodnotami typu `float`, tedy již jas přepočtený na interval 0 až 1.
- **Program.cs** – Tento soubor slouží jako vstupní bod aplikace, je v něm vytvořen hlavní formulář a zpracovávána fronta zpráv systému.
- **Scanner.cs** – Zaobaluje získání obrazu ze skeneru pomocí technologie Windows Image Acquisition.

Důležité procedury a funkce

- `minimizeConsole, maximizeConsole` Vyvolání zobrazení nebo skrytí konzole přiřazené k programu.
- `initPreviews` Inicializace obrazových polí pro načtení náhledů vstupních snímků.
- `drawNormals` Vypočte a vykreslí kladné či záporné části normál, dle parametru.
- `btnNormalsBoth` Vypočte a vykreslí normálové pole, zde je vizualizace pro celý rozsah normál – $0,5$ šedá barva je nulová složka normály. Normály jsou pak dostupné v polích output[x|y|z]

- `loadBitmap` Dle parametru (index 0 až 3) zobrazí dialog otevření souboru a načtený obrázek ustanoví jako odpovídající podklad pro rekonstrukci.
- `createPreview` Vytvoří a vykreslí náhled pro zvolený index (0 až 3) vstupního snímku.
- `drawPreviewOnCanvas` Zobrazí zvolený vstupní obraz na plátno.
- `readyForNormals` Vrátí `true`, pokud je vše připraveno pro výpočet normálového pole.
- `drawNzHistogram` Vykreslí histogram zastoupení normál 'nz' do nového okna.
- `drawSlice` Vykreslí řez normálovým polem.
- `drawSliceHeigth` Vykreslí řez výsledným modelem.
- `btnIntegrate` Provede řádkovou integraci, výsledek je zapsán do proměnné `surface`.
- `btnIntegrate2D` Provede plošnou integraci povrchu, výsledek je zapsán do proměnné `surface`.
- `autoComputeBoundary` Vypočte okraje na základě rohových hodnot.
- `btnExportTris` Exportuje již hotový model, tedy vrcholy s normálami, do souboru, jehož název vybere uživatel na základě dialogové výzvy.

Literatura

[1] Rongjiang Pan, Vaclav Skala. Normal Map Acquisition of Nearly Flat Objects Using a Flatbed Scanner. 2013. [cit. 2013-03-14].

[2] PINTUS, Ruggero, Thomas MALZBENDER, Oliver WANG, Ruth BERGMAN, Hila NACHLIELI a Gitit RUCKENSTEIN. Photo Repair and 3D Structure from Flatbed Scanners. HP Technical Reports [online]. 2009, č. 37, s. 11 [cit. 2012-10-03]. Dostupné z: http://hpl.hp.com/techreports/2009/HPL-2009-37.pdf

[3] BARSKY, Svetlana, Maria PETROU a Ioannis IVRISSIMTZIS. The 4-Source Photometric Stereo Technique for Three-Dimensional Surfaces in the Presence of Highlights and Shadows. IEEE Transactions on Pattern Analysis and Machine Intelligence [online]. 2003, roč. 25, č. 10, s. 1239-1252 [cit. 2012-10-03]. Dostupné z: http://cseweb.ucsd.edu/classes/wi06/cse252a/hw2/BarskyAndPetrou.pdf

[4] SCHRÖDER, Gottfried. Technická optika. Praha: SNTL, 1981, 158 s. ISBN 01-240-8650-0.

[5] BUHMANN, M. Radial basis functions: theory and implementations. Cambridge: Cambridge University Press, 2003, x, 259 s. ISBN 978-0-521-10133-2

[6] KHREICH, Wael, Eric GRANGER, Ali MIRI a Robert SABOURIN. On the memory complexity of the forward and backward algorithm. Pattern Recognition Letters [online]. roč. 31, č. 2, s. 91-99 [cit. 2013-02-08]. ISSN 01678655. DOI: 10.1016/j.patrec.2009.09.023. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0167865509002578

[7] Singulární rozklad matice a jeho použití. Brno, 2012. Dostupné z: http://is.muni.cz/th/211561/prif_m/Vodrazkova_DP.pdf. Diplomová práce. Masarykova univerzita. Vedoucí práce Mgr. Jiří Zelinka Dr.

[8] SKALA, Václav. Světlo, barvy a barevné systémy v počítačové grafice. 1. vyd. Praha: Academia, 1993, 130 s., [4] s. barev. fotogr. ISBN 80-200-0463-7.

[9] DOBEŠ, M., Zpracování obrazu a algoritmy v C#. Technická literatura BEN, ISBN 978-80-7300-233-6, Praha, 2008.

[10] WAVEFRONT TECHNOLOGIES. Object Files (.obj). [online]. 2007 [cit. 2013-05-09]. Dostupné z: http://www.martinreddy.net/gfx/3d/OBJ.spec

[11] KUČERA, Radek. Numerické metody. 1. vyd. Ostrava: VŠB - Technická univerzita, 2006, 132 s. ISBN 80-248-1198-7.

[12] THAYER, Jeffrey S. MASSACHUSETTS INSTITUTE OF TECHNOLOGY. Competitive Strategic Advantage Through Disruptive Innovation. Massachusetts, USA, 1996. Dostupné z: http://dspace.mit.edu/bitstream/handle/1721.1/10954/35749974.pdf

[13] FRANCESCO ABATE, Andrea, Maria DE MARSICO, Stefano RICCIARDI a Daniel RICCIO. Normal maps vs. visible images: Comparing classifiers and combining modalities. [online]. [cit. 2012-10-03]. DOI: 10.1016/j.jvlc.2009.01.004. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S1045926X09000056

[14] GRIMM, Cindy, William D. SMART, Kenneth I. JOY a Hila NACHLIELI. Shape classification and normal estimation for non-uniformly sampled, noisy point data. Computers [online]. 2011, roč. 35, č. 4, s. 904-915 [cit. 2012-10-03]. ISSN 00978493. DOI: 10.1016/j.cag.2011.03.036. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0097849311000756

[15] YOON, Mincheol, Yunjin LEE, Seungyong LEE, Ioannis IVRISSIMTZIS a Hans-Peter SEIDEL. Surface and normal ensembles for surface reconstruction. Computers [online]. 2011, roč. 35, č. 4, s. 904-915 [cit. 2012-10-03]. ISSN 00978493. DOI: 10.1016/j.cad.2007.02.008. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0010448507000516