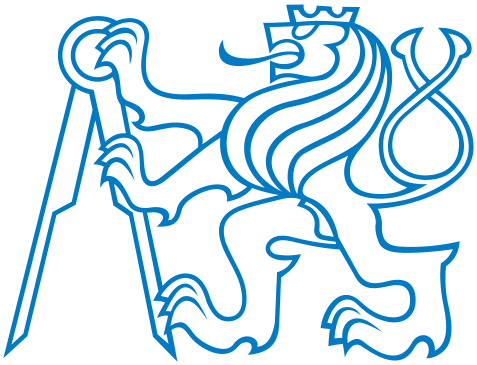


ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra počítačové grafiky a interakce

Vytváření hloubkových map ve vysokém rozlišení

Depth-map super-resolution

Diplomová práce

Studijní program: Otevřená Informatika

Studijí obor: Počítačová grafika a interakce

Vedoucí práce: Ing. David Sedláček, Ph.D.

Tomáš Kříž

Praha 2019

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kříž** Jméno: **Tomáš** Osobní číslo: **382817**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Studijní obor: **Počítačová grafika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Vytváření hloubkových map ve vysokém rozlišení

Název diplomové práce anglicky:

Depth-map super-resolution

Pokyny pro vypracování:

Prostudujte metody používané pro zvýšení rozlišení obrázků nebo hloubkových map založené na složení informace z několika po sobě následujících statických snímků. U vybraných metod dohledejte referenční implementace a metody porovnejte.
Dle analyzovaných metod navrhněte a implementujte metodu vhodnou speciálně pro zvýšení rozlišení hloubkové mapy pořízené ToF hloubkovou kamerou (Time of Flight).
Výsledky srovnejte s referenčními metodami jak na syntetických datech (referenční dataset), tak na datech pořízených vybranými hloubkovými kamerami.

Seznam doporučené literatury:

[1] Jiajun Lu, David Forsyth; Sparse Depth Super Resolution. CVPR 2015
[2] Jing Li, Zhichao Lu, Gang Zeng, Rui Gan, Hongbin Zha; Similarity-Aware Patchwork Assembly for Depth Image Super-resolution. CVPR 2014
[3] Jun Xie, Rogerio Schmidt Feris, and Ming-Ting Sun; Edge-Guided Single Depth Image Super Resolution. IEEE Transactions on Image Processing vol. 25, 428-438, January 2016

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. David Sedláček, Ph.D., katedra počítačové grafiky a interakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **26.09.2018** Termín odevzdání diplomové práce: **08.01.2019**

Platnost zadání diplomové práce: **19.02.2020**

Ing. David Sedláček, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. Datum převzetí zadání

Podpis studenta

Prohlášení

„Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.“

V Praze, 8.1.2019

Poděkování

Chtěl bych poděkovat své rodině a přátelům za podporu při studiu. Také chci poděkovat vedoucímu práce za neuvěřitelnou trpělivost a za zapůjčení 3D kamery.

Abstrakt

Diplomová práce se zabývá supersamplingem, metodou pro zvětření počtu vzorků, hloubkových map. Práce se zabývá existujícími implementacemi a algoritmy, které tyto implementace používají. Dále se práce zaměřuje na porovnání implementací. V druhé části je v práci představena vlastní metoda, která je založená na nápadech z již existujících metod a je zde popsána její implementace. V závěru je v této práci porovnání této implementace s ostatními metodami na testovacích sadách používaných k analýze metod supersamplingu.

Obsah

1 Úvod **3**

2 Měření vzdálenosti pomocí paprsku světla **3**

3 Senzory **4**

3.1 Princip Time of Flight (ToF) senzorů 7

3.2 Princip strukturovaného světla 8

4 Hloubkové mapy **9**

5 Supersampling **11**

5.1 Aliasing 11

5.2 Rekonstrukce z více snímků 12

5.3 Metody s učením 13

5.3.1 Neuronové sítě 13

5.3.2 Markovovy modely 14

5.4 S barevným obrazem 14

6 Studované algoritmy **15**

6.1 Medián 16

6.1.1 Bilaterální filtr 16

6.2 LidarBoost [16] 17

6.2.1 Implementace 18

6.3 Edge-Guided Single Depth Image Super Resolution [17] 18

6.4 Similarity-Aware Patchwork Assembly for Depth Image Super-resolution [18] . 19

6.5 Sparse Depth Super Resolution [19] 21

7 Vstupní data **21**

7.1 Ground truth 23

8 Analýza implementací **24**

9 Návrh implementace **27**

9.1 Detekce hran 27

9.1.1 Sobelův detektor 28

9.1.2 První derivace Gaussovy funkce 29

9.1.3 Detekce pomocí druhé derivace [24] 32

9.2 Rekonstrukce 33

10 Postup Implementace **33**

11 Výsledky **35**

11.1 Porovnání s mediánem 36

11.2 Vliv počtu snímků 37

11.3 Časová náročnost 38

11.4 Vizuální analýza výsledků 40

12 Závěr **47**

12.1 Pokračování práce 47

13 Návod - Aplikace pro prezentaci **49**

14 Rozhraní knihovny **52**

15 Reference **57**

16 Přílohy **60**

1 Úvod

Optické měření vzdálenosti má mnoho využití od skenování objektů, detekce překážek nebo v lékařství pro skenování povrchu pro výrobu protéz. Různé senzory mají různé přesnosti měření. Co je ale společné všem senzorům je malé rozlišení, pokud jde o měření hloubkových map. Barevné senzory v kamerách jsou v dnešní době schopné vytvářet fotografie s rozlišením v řádu megapixelů. Senzory, které vytváří hloubkové mapy, ale často nedosahují ani polovinu megapixelu. Tento problém znovu otevřel téma supersamplingu, který byl předtím rozšířeným tématem u barevných fotografií, kde vzniklo mnoho implementací a metod, jak problém řešit. Hloubkové mapy mají jiné využití a proto je třeba přezkoumat všechny metody, používané na fotografie, zda jsou použitelné zde, případně navrhnout jiné metody, vhodnější právě pro hloubkové mapy.

V této práci se zaměřím nejprve na zmapování dostupných metod pro generování hloubkových map, jejich výhody i nedostatky. Představím, co jsou hloubkové mapy a co je supersampling a k čemu se používá. Následně popíšu různé metody jak problém supersamplingu řešit a představím některé implementace těchto metod. Dále, pomocí výsledků z pozorování chování dříve popsaných metod, navrhnu vlastní řešení problému a popíšu implementaci tohoto řešení. Nakonec toto řešení porovnám s některými implementacemi, dostupnými na internetu.

2 Měření vzdálenosti pomocí paprsku světla

Měření vzdálenosti pomocí paprsku světla funguje na principu měření doby mezi vysláním světelného paprsku a detekcí návratu jeho odrazu. Rychlost světla je konstantní ve vakuu, jenže tato měření jsou často prováděná v atmosféře Země. Protože rychlost světla je na kratší vzdálenosti ve vzduchu konstantní, je možné vzdálenost spočítat jednoduše pomocí rychlosti světla (1) a indexu lomu dané látky, kterou prochází. Index lomu vzduchu při normálním atmosferickém tlaku je 1.00026. Na větší vzdálenosti je možné spočítat vzdálenost téměř přesně, pokud správně spočítáme poměry indexů lomu látek, kterými musel paprsek projít.

$$v_{air} = c/n_{air} = 2.998 \cdot 10^8 \ ms^{-1} \ / \ 1.00026 \tag{1}$$

Samotné spočítání vzdálenosti se provádí po změření doby letu paprsku. Změřenou dobu t je potřeba nejprve vydělit dvěmi, protože v této změřené době je obsažena cesta fotonů k místu

odrazu i zpět. Poté se jen vynásobí rychlostí světla v daném přenosovém médiu (nejčastěji vzduch) a jako výsledek dostaneme přesnou vzdálenost od kamery (2). Takto přímo je možné měření provádět, při příznivých podmínkách, i na vzdálenost 10 km [9] i více.

$$d = \frac{t}{2} \cdot v_{air} \tag{2}$$

3 Senzory

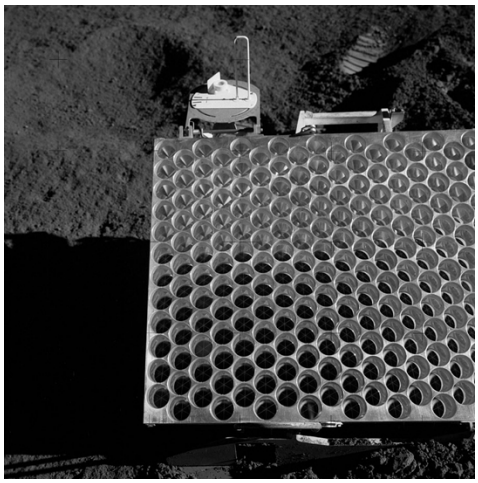
Technika optického měření vzdálenosti existuje už několik desítek let [1]. Optické měřiče vzdálenosti používají laserové paprsky, jejichž vlnová délka se liší podle použití. Vědci používají tato měření ke studiu atmosféry, povrchu planet nebo skenování 3D objektů. V posledních letech se začaly objevovat i běžně dostupné ruční optické metry, které používají levnější komponenty a nahrazují v některých odvětvích různá pevná měřidla, jako například svinovací metr.

Při studiu atmosféry se měří například výšky a hustoty mraků. S různými vlnovými délkami je potom možné měřit teplotu, tlak, vlhkost a jiné vlastnosti. Tím se zabývá spektrofotometrie, kde se určuje složení vzorku pomocí jeho optických vlastností.

Často se tato metoda používá ke skenování blízkých vesmírných těles. Například skenování povrchu měsíce je možné z povrchu Země. Ke změření přesné vzdálenosti Země a Měsíce je možné použít reflektory (Obr. 1), které na Měsíc umístily astronauti vesmírných misí Apollo [2]. Tyto reflektory ale už časem ztratily svoje původní vlastnosti a jsou nepřesné. Skenování Země samotné se provádí buď z letadel, nebo i ze satelitů. NASA změřila většinu povrchu Země [6] a data s rozdílnou přesností v řádu desítek až stovek metrů jsou veřejně přístupná na internetu.

Skenování 3D objektů a prostoru pomocí optických měření je v dnešní době velmi oblíbené v různých oblastech lidské činnosti od výzkumu až po zábavu. Používá se například k navigaci robotů a automobilů, k získání dat o okolí pro rozšířenou nebo virtuální realitu, ale i k uchování dat o památkách a archeologických nálezech. Začíná se objevovat i uplatnění v medicíně, například pro skenování povrchu pro různé náhrady a protézy. Pro všechny tyto aplikace se používají různé typy senzorů podle požadavků na přesnost, rychlost a výslednou pořizovací cenu.

K přesnému skenování statických objektů se používají optické měřiče vzdálenosti, které



Obr. 1: Část fotoreflektoru z mise Apollo 15, umístěno na měsíc a vyfoceno D. Scottem. [3]

používají rozmítaný laserový paprsek. Tyto systémy mají pevnou měřící jednotku a používají zrcátko k zaměření laserového paprsku na jednotlivé měřené body v měřeném rozsahu. Velmi často takovéto systémy umí měřit pouze v jedné rovině. Pro detekci překážek je to často dostačující. Skenování 3D objektu pomocí takovéhoto systému (2) se provádí postupně, kdy systém je připevněný na trojnožce a posun v druhé ose je zajištěn natočením celého systému, buď automaticky, nebo manuálně. Všechny tyto metody měří jednotlivé body zvlášť. Proto jsou tato měření sice přesná, ale omezená mechanickou rychlostí celé soustavy.

Další používaný princip je podobný senzorům ve fotoaparátech a kamerách, kde na jednom čipu je v mřížce vedle sebe uspořádáno více jednotlivých optických senzorů a tak jsme schopní generovat hloubkové mapy. Tyto senzory už nepoužívají laser, ale osvětlí celou scénu pomocí záření jedné vlnové délky. Senzory poté snímají odražené záření pomocí integrování detekované intenzity záření, které na ně dopadá. Více jednotlivých senzorů je často seřazeno, aby se oddělily jednotlivé časové úseky a tak bylo možné přesně určit zpoždění paprsku. Nejčastěji se takto sdružují 4 jednotlivé senzory na jeden obrazový bod. Takto dostupné senzory jsou drahé a jejich rozlišení je zatím, oproti optickým senzorům, nízké obvykle okolo 320×240 pixelů [8]. Jejich cyklické chování je ale vhodné pro zaznamenávání videí, či sekvencí obrazů rychle po sobě. Velmi často tyto kamery pracují s frekvencí 30 snímků za sekudnu. Tomuto principu se budu podrobně věnovat v kapitole 3.1, protože tento princip používá drtivá většina kamer a snímačů, se kterými budu pracovat a pro které je metoda supersamplingu primárně určená, kvůli jejich nízkému rozlišení.

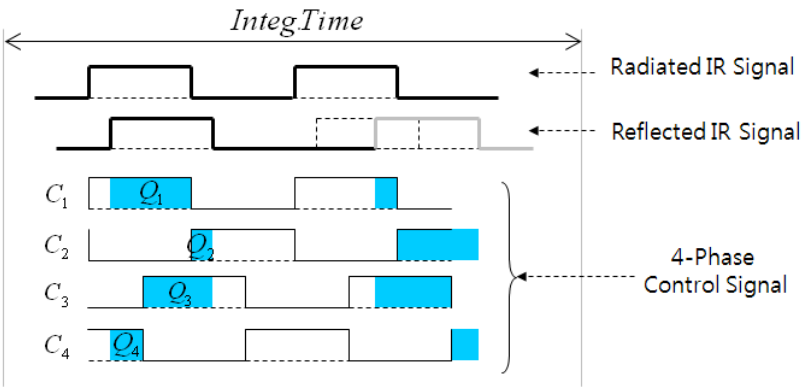
Obr. 2: Fotografie Lidar systému pro skenování 3D modelů památek, skalních formací, aj., model Leica HDS-3000 [4]. Systém má zrcátko, které rozmítá paprsek vertikálně a hlava systému se otáčí okolo horizontální osy. Systém potřebuje k řízení notebook se programem a s jeho pomocí tento systém automaticky skenuje okolí a skládá 3D modely.

Další používaný princip má uplatnění spíše v zábavním průmyslu, protože je nepřesný, ale velmi levný, protože vůbec nepoužívá optické měření vzdálenosti. Místo toho používá sérii kamer a projektor. Projektor promítá do prostoru neviditelný obrazec se speciálními vlastnostmi. Kamery potom snímají tento obrazec a program pomocí deformací tohoto obrazce je schopný vytvořit přibližný tvar monitorovaného prostoru. Toto s velkým úspěchem využil například známý systém Kinect v první verzi. Kinect druhé verze už používá princip popsaný výše a skutečně měří čas letu paprsků světla. Intel nabízí kombinovanou kameru [8] s rozlišením 1280×720, která ale používá tento princip. Tento princip vysvětlím v kapitole 3.2, ale přestože je možné aplikovat supersampling i na tyto hloubkové mapy, jejich kvalita je obvykle špatná a supersampling by zde mohl pomoct právě s kvalitou, protože rozlišení je dostatečně velké.

3.1 Princip Time of Flight (ToF) senzorů

Aby jsme získali možnost měřit se stejnou přesností na celé snímané ploše, musíme problém z kapitoly 2 přeformulovat. Místo detekování navráceného paprsku, budeme integrovat intenzitu přicházejícího světla a rozdělíme snímání a vysílání paprsku do dvou stejně velkých fází. Tímto limitujeme minimální a maximální vzdálenost od kamery, ale převedeme dobu, po kterou paprsek letí na sumu intenzity přicházejícího záření. Limity jsou přímo závislé na frekvenci s jakou přepínáme jednotlivé fáze, protože rychlost světla je v podstatě konstantní na krátké vzdálenosti, na které je tento princip používaný. Suma intenzity přijmutého záření je při použití tohoto principu úměrná času, po který paprsek letí. To nám dovolí synchronizovat měření v celém čipu a zároveň rozšířit takový čip do matice senzorů, které budou integrovat intenzitu záření. Také bude potřeba nahradit laserový paprsek, který je obvykle příliš směrový, zdrojem rozptýleného záření o specifické vlnové délce (například LED - Light Emiting Diode), tak, aby se eliminovalo rušení z viditelného spektra. Tento princip je samozřejmě zjednodušený, ale je základní pro funkci moderních ToF čipů, které měří dobu letu paprsků světla a tím vzdálenost bodu ve scéně od kamery.

Dnešní Time of Flight senzory používají rozdíl fázového posunu opakujícího se signálu k rozlišení vzdálenosti. Tyto senzory jsou mnohem složitější, než představené přeformulování problému. Nasnímaný průběh je porovnán se čtyřmi fázově rozdílnými signály, které jsou následně převedené na skutečnou hloubku. Toto eliminuje omezení na minimální vzdálenost od kamery. Omezení na maximální vzdálenost přesto zůstává.



Obr. 3: Obrázek z knihy [10], který zobrazuje vyzářený signál, odražený signál a čtyř-fázový řídící signál, díky kterým je možné zjistit rozdíl fází vyzářeného a odraženého signálu.

Paprsek je periodicky vysílán do scény a jeho odraz je registrován čtveřicí bodů z matice senzorů. Tyto body integrují signál podle kontrolního signálu z řídící jednotky. Řídící jednotka produkuje jeden stejný signál, který je ale pro každý ze čtveřice bodů fázově posunutý. Následně se spočítá pomocí (3) fázový posun vyslaného a odraženého signálu. Díky tomuto fázovému posunu můžeme měřit vzdálenost od kamery (4) až po určitou vzdálenost, danou frekvencí řídícího signálu. Tato maximální vzdálenost je definována jako $c/(2f)$. V podstatě tato vzdálenost odpovídá fázovému posunu většímu, než je 2π kdy začne být fázový posun větší, než je perioda signálu a my nedokážeme rozpoznat, zda se jedná o blízký předmět, nebo předmět, který je vzdálený víc, než je maximální vzdálenost.

Tento problém se dá částečně řešit pomocí předzpracování, kdy hledáme gradienty, které ve směru zvětšování vzdálenosti přeskakují o téměř celý rozsah výstupních dat. Tyto přechody není těžké detekovat a je nutné potom celou mapu přeškálovat, nebo změnit rozlišení pixelů. Ve své práci už ale pracuji s předzpracovanými daty, proto se tímto problémem nebudu dále zabývat.

$$t_d = atan\left(\frac{Q_3 - Q_4}{Q_1 - Q_2}\right) \tag{3}$$

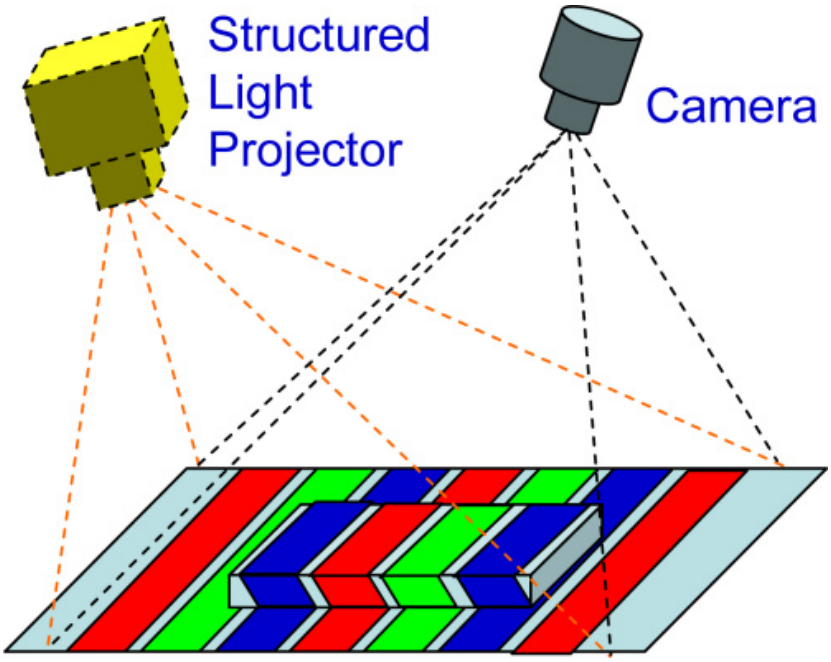
$$d = \frac{c}{2f}\frac{t_d}{2\pi} \tag{4}$$

3.2 Princip strukturovaného světla

Jiný princip měření používá senzor Kinect, který používá specifický princip strukturovaného světla. Tato metoda je založená na promítání předem vytvořeného světelného obrazce do scény. Podle deformací tohoto obrazce je potom možné vytvořit zpětně deformace ve scéně a také hloubku, podle vzdálenosti jednotlivých specifických bodů v obrazci. Tato metoda je obecně rychlá, ale její přesnost závisí na kvalitě promítaného obrazce a kvalitě kamer, které daný obraz snímají.

Obrazce lze zařadit do několika skupin podle vlastností, nebo počtu potřebných snímků. Používají se například sekvenční projekce, které promítají sekvence obrazců, které mají například různou hustotu, nebo jsou fázově posunuté. Pokud nám nevadí použití viditelného spektra k projekci, je možné kódovat prostor například barvou. Také se používají různé sekvence pruhů, matice teček, nebo mřížka z pruhů. Typy obrazců je také možné kombinovat,

například barevnými maticemi teček, nebo barevnou mřížkou.



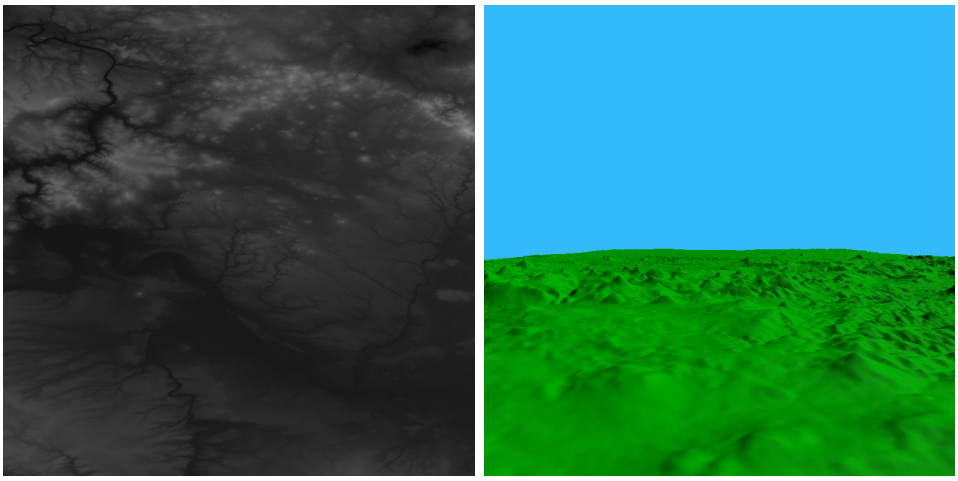
Obr. 4: Schéma systému kamery a projektoru strukturovaného světla [11]

4 Hloubkové mapy

Hloubkové mapy jsou uniformní 2D matice, které v jednotlivých buňkách obsahují vzdálenost od senzoru, neboli hloubku v obraze. Rozlišení hloubkových map může být různé. Pro některé senzory je také použité skládání obrazu, kdy hloubka má sice omezený počet bitů, ale v obraze se po gradientu opakuje, takže v různých bodech může stejné číslo znamenat násobek hloubky a skutečnou hloubku je pak potřeba rekonstruovat, jak bylo popsáno v předchozí kapitole.

Tím, že se jedná o uniformní matici, je pozice bodu ve 3D prostoru kódovaná jedním číslem a pozicí v matici. Tato reprezentace je méně přesná, protože není přesně schopná zaznamenat hrany objektů, zato je velmi dobře komprimovatelná. Na hloubkovou mapu se často dají uplatnit poznatky z geometrie a optiky, protože optická část těchto senzorů je stejná jako u kamer a fotoaparátů. Je možné tyto hloubkové mapy převést například na mračno bodů a snímáním objektu z více úhlů poté tyto mračna bodů spojit tak, aby vytvořila 3D model objektu.

Často je tato reprezentace použitá opačně, známá jako výšková mapa, kde jednotlivé buňky obsahují naopak výšku daného bodu v prostoru (Obr. 5). Vlastnosti výškových map jsou podobné. Výškové mapy se nejčastěji používají k reprezentaci terénu ať už reálného, nebo virtuálního.



(a) Obrázek samotné výškové mapy

(b) Ukázka vykreslení terénu z výškové mapy za pomocí metody sledování paprsku

Obr. 5: Ukázka výškové mapa a jejího vykreslení. Obrázky jsou z výsledků mojí semestrální práce z předmětu Datové struktury pro počítačovou grafiku

Jako každý vzorkovaný signál, jsou hloubkové mapy omezené stejnými pravidly jako fotografie. Jedno z nejdůležitějších omezení je Shannon/Nyquistův teorém [5]. Ten říká, že přesná rekonstrukce spojitého, frekvenčně omezeného signálu podle jeho vzorků, je možná pouze tehdy, pokud je vzorkovací frekvence vyšší, než dvojnásobek nejvyšší harmonické složky vzorkovaného signálu. Důsledky tohoto teorému se nejlépe ukazují na detailech, které jsou při příliš malém počtu vzorků nerozpoznatelné. Detaily ve fotografiích a hloubkových mapách totiž odpovídají vysokým frekvencím, které jsou nutné k přesnému zobrazení drobných ploch a přechodů mezi nimi. Supersampling je metoda, která se snaží zvýšit počet vzorků a tím umožnit rekonstrukci právě těchto detailů. Přesto je často počet vzorků nedostatečný i při opakovaném snímání prostoru a tak tato metoda není schopná věrně rekonstruovat detaily.

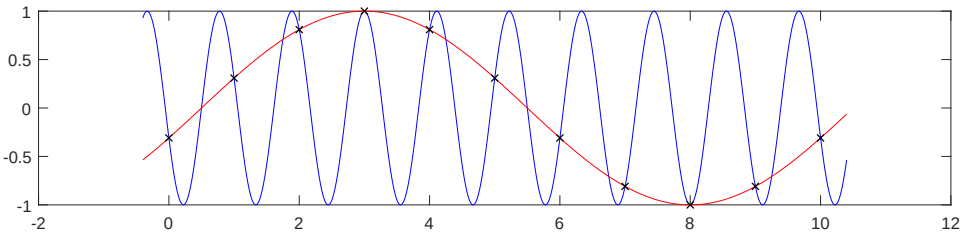
5 Supersampling

Supersampling samotný je téma, které je starší a bývalo doménou videa a fotografie. Tyto senzory už ale dávno překročily hranici, kdy by bylo potřeba podobné metody aplikovat, a proto vývoj v této oblasti ustal. V posledních letech ale bylo toto téma znovu otevřeno, protože jsou na trhu dostupné senzory, které snímají hloubku scény a tyto senzory mají stejný problém, který předtím měly i fotografické senzory - malé rozlišení.

Vzhledem k omezení, které vychází z povahy dat je důležité představit jeden jev, který úzce souvisí se supersamplingem a zvětšováním rozlišení jakýchkoliv diskrétních dat, která byla vytvořená navzorkováním.

5.1 Aliasing

Aliasing [12, 13] je jev, který nastává při převodu spojitého signálu na diskrétní, neboli při vzorkování. Pokud je vzorkovací frekvence nižší, než dvojnásobek nejvyšší frekvence, obsažené v původním signálu, dojde ke ztrátě informace o těchto vyšších frekvencích. V plošných datech se toto projevuje rozmazáním, ztrátou nebo deformací detailů, nebo dokonce se může objevit vzor, který se v původním signálu ani nevyskytoval (Obr. 6).



Obr. 6: Ukázka aliasingu při vzorkování funkce sinus. Při podvzorkování původní funkce (modrá) se při rekonstrukci změní úplně perioda funkce (červená). Vznikne tak frekvence, která v původním signálu nikdy nebyla.

Supersampling je obecně metoda k odstranění aliasingu, kdy se obraz vzorkuje více vzorky, než je potřebné. Ve zpracování obrazu, se ale často pod pojmem supersampling myslí zvětšení obrazu pomocí vícenásobného vzorkování. Toto vícenásobné vzorkování je samozřejmě realizované skrz násobné pořizování snímků scény.

Supersampling se v dnešní době běžně používá i při renderování počítačové grafiky. Pro přímé renderování se používá několik metod s různými efekty, od použití metody supersam-

plingu na celou scénu (SSAA - Super sampling anti-aliasing) až po supersampling pouze okolo hran trojúhelníků, ze kterých je tvořený model. Méně často se používá přímý supersampling, kdy se grafika renderuje ve vyšším rozlišení a následně v post-processing zpracování se scéna převzorkuje na požadované rozlišení.

Při použití paprsků k renderování grafiky se často používá velké množství paprsků, často rozptýlených po ploše obrazového bodu, aby se zpřesnil výpočet. To je nutné při použití materiálů, které mají odrazy nebo refrakci [13]. Tyto materiály totiž zvětšují plochu, která má být zobrazená v daném obrazovém bodě a při použití malého množství paprsků vede k velmi špatným výsledkům. Obzvlášť při výpočtu globálního osvětlení je nutné použít více paprsků, aby se odstranil náhodný šum, vzniklý střílením velkého množství paprsků náhodně do scény. Supersampling se tedy využije přirozeně tím, že se použije velké množství paprsků.

Metody na rekonstrukci hloubkových map se dají rozdělit zhruba do tří kategorií, které používají rozdílný princip a přístup k problému. První kategorie jsou rekonstrukce z více snímků, která je založená na supersamplingu, který se běžně používal u barevných fotek. Metoda je založená na zarovnání několika snímků a následné rekonstrukce chybějící informace skrz různé interpolace a filtry. Nejjednodušší takovou metodou je například průměrování, medián, nebo bilaterální filtr. Další kategorie jsou rekonstrukce, které používají nějaká data z učení, nebo z předešlých rekonstrukcí. Do této kategorie spadají především konvoluční neuronové sítě, nebo Markovovy modely. Třetí kategorie používá navíc barevný obraz k extrakci detailů.

Tyto metody lze samozřejmě kombinovat a dosáhnout tím lepších výsledků. Pro jednoduchost se ale většinou používá pouze jedna metoda podle vstupu, výsledné kvality a potřebné rychlosti.

5.2 Rekonstrukce z více snímků

Rekonstrukce z více snímků, nebo také multisampling [13], je především výhodná proto, že už ze samotného principu znevýhodňuje extrémy, které jsou často způsobené šumem. Tyto metody prosazují střední hodnotu v daném bodě a tím redukují šum v obraze. Skrz drobný pohyb kamery je pak možné nasnímat více pozic a tím dostat vzorky, které leží mezi vzorky původního obrazu. Podobného efektu se dá dosáhnout chytrou analýzou obrazu například pomocí detekce hran. Rekonstrukce potom může být velmi jednoduchá, složitější algoritmy

ale obvykle vedou k lepším výsledkům. Jasnou nevýhodou této metody je nutnost registrace obrazů, pokud se hýbe s kamerou a delší časový úsek, během kterého se sbírají jednotlivé snímky. Metoda je ale velmi přímočará a nepotřebuje učení ani další data, protože funguje pouze se sérií hloubkových map.

Samotný supersampling se provádí pomocí vzorkování z původního, nižšího rozlišení a různé aproximace.Ve všech metodách je velmi časté použití detekce hrany. Používá se detekovaná hrana z původního, menšího rozlišení, ale i hrana, která je detekováná v lineárně aproximovaném zvětšení. Častý je algoritmus bilaterálního filtru, který kombinuje lineární filtr na vyhlazení s nelineárním filtrem, který modifikuje vyhlazení podle velikosti rozdílu intenzit a tím zachovává v původním obraze ostré hrany.

5.3 Metody s učením

Tyto metody používají nějakou formu učení nebo databázi dat, podle kterých se provádí klasifikace a rekonstrukce dat. Tyto metody jsou náročnější na paměť právě kvůli potřebě uchovat naučená data. Tyto metody také nejde použít bez předem naučené databáze. Tato databáze je ale často přenosná, tudíž stačí algoritmus učení spustit jen jednou. Také je možné, pokud to algoritmus učení dovoluje, naučit pouze nutné minimum a následně speciální případy doučit v případě potřeby a tím minimalizovat paměťovou náročnost.

Metody s učením se rozdělují na dvě kategorie. První kategorie je rekonstrukce s použitím konvolučních neuronových sítí. Druhá kategorie používá ke klasifikaci plošných Markovovských modelů, které následně určí nejpravděpodobnější hranu ze seznamu, která na dané místo patří.

5.3.1 Neuronové sítě

Neuronové sítě [15] se snaží modelovat chování neuronů. Sítě neuronů jsou schopné klasifikovat, ale také dávát přímé výsledky a proto se používají jako složité nelineární filtry. Neuronové sítě mají naučená data uložená uvnitř jako váhy vstupů jednotlivých neuronů.

Nejčastěji se používá konvoluční typ neuronové sítě, který je v poslední době velmi oblíbený. Tyto sítě fungují právě jako nelineární filtry a jsou relativně malé, protože mají omezený vstup a výstup. Jejich výhoda je v přímé interakci, kde z jedné strany se přivedou data a z druhé přímo vypadávají výsledky, které je možné přímo použít. Nejčastěji se používá dvojice tvarů vstupní a výstupní hrany, kde se odpovídající zvětšená výstupní hrana pomocí učení uloží jako

váhy jednotlivých neuronů a potom je možné pomocí této klasifikace složit hrany objektů v obraze.

5.3.2 Markovovy modely

Markovovy modely [17] se používají ke klasifikaci a modelaci různých náhodných jevů. Zpracování je složitější, protože Markovovy modely pracují s pravděpodobnostmi a tak tyto modely obsahují často logaritmy, aby bylo možné, počítat s nimi v počítači, který má omezenou reprezentaci desetinných čísel, což často po několika krocích klasifikace vede bez logaritmování pravděpodobností k vynulování většiny čísel, kvůli tomu, že se pravděpodobnosti (které jsou v rozsahu $\langle 0; 1 \rangle$) mezi sebou násobí. Po spočítání pravděpodobností je potřeba ještě najít nejvyšší pravděpodobnost, což může při velkém množství klasifikačních dat trvat dlouhou dobu.

Na supersampling se používají Markovovy náhodné pole (Markov Random Field), které jsou zobecněním Markovových řetězců do obecného grafu, kde každý stav závisí na sousedních stavech.

Pomocí obou metod se vytváří čtvercové filtry, obvykle o velikostech 5 až 9 obrazových bodů na vstupu. V případě neuronových sítí bývá vstup 9 až 20 obrazových bodů. Tyto filtry jsou ale pevně nastavené na určité zvětšení, takže je možné obraz zvětšit pouze po násobcích předpřipravených filtrů. Další nevýhoda je generování artefaktů (Obr. 7), které se vyskytují v předem naučených vzorech, nebo kvůli špatné klasifikaci. Naučené vzory totiž nemohou, při rozumné velikosti učících dat, obsáhnout všechny možné tvary, které se ve vstupních datech a ve výsledcích mohou vyskytnout a tak se na daná místa dostanou nejpodobnější tvary. To má za následek vytváření tvarů, které v původním obraze ani neexistovaly. Výhodou je ale možnost použít jediný snímek k výpočtu supersamplingu. Teoreticky by tak bylo možné s touto metodou dosáhnout stejné snímkové rychlosti jaké dosahuje samotný senzor, pokud by klasifikace a vyhledávání bylo dostatečně rychlé. Technicky je ale klasifikace velmi pomalá, takže tyto algoritmy patří mezi nejpomalejší, přesto mají velmi kvalitní výsledky.

5.4 S barevným obrazem

Metody, které používají barevný obraz [19], využívají poznatků o globálním osvětlení, texturách a barvách objektů k tomu, aby zpřesnily rekonstrukci. Toto je možné detekcí

(a) Vstup v plném rozlišení z datové sady

(b) Rekonstrukce z více snímků.

(c) Klasifikace pomocí Markovských modelů

Obr. 7: Ukázka grafických artefaktů jednotlivých metod. Výřezy jsou z výsledků implementovaných algoritmů.

stejně barevných ploch, které naznačují, že daná plocha má stejnou hloubku, nebo alespoň konstantní gradient. Nevýhodou této metody je nutnost mít i barevné kanály ze senzoru. Tyto senzory proto musí být kombinované. Dostupné kombinované senzory mají dvě čočky, což znamená, že senzory jsou umístěné vedle sebe a proto nesnímají úplně stejný obraz. To ale ve výsledku nevadí, protože čočky jsou dostatečně blízko na to, aby chyba způsobená jinou pozicí byla zanedbatelná. Tyto metody jsou obvykle založené na hledání velkých změn v barvě, které mapují na ostré přechody v hloubce. Díky tomu, že barevné kamery mají dost často násobně větší rozlišení, je tak možné udělat celkem přesné a velké zvětšení z jednoho snímku. Nevýhodou je mnohem větší množství vstupních dat, právě kvůli barevným kanálům a jejich rozlišení.

6 Studované algoritmy

Hlavní účel této práce je zmapovat a prostudovat momentálně používané algoritmy a nejnovější metody, které jsou dostupné. Následně ze získaných poznatků navrhnout a naimplementovat novou metodu. Jako základní metodu pro porovnávání jsem zvolil medián z více snímků. Tato metoda je jednoduchá na implementaci a velmi rychlá. Nevýhodou tohoto zjednodušení je samozřejmě nízká kvalita výsledku. Navržený algoritmus by tedy rozhodně neměl být horší, než tato metoda, obzvlášť pokud se rozhodnu výstup z této metody dále zpracovávat.

6.1 Medián

Algoritmus najde pro každý výstupní bod obrazu odpovídající pixely ze vstupních dat. Tyto nalezené pixely seřadí a jako výstup vezme prostřední hodnotu. Podobně funguje i bilaterální filtr, který ale nalezené hodnoty vynásobí koeficientem podle toho, jak moc se hodnota liší od prostřední hodnoty. Pro více snímků se jako prostřední hodnota vezme medián v daném bodě ze všech snímků. Koeficient je menší, čím víc se hodnota liší. Mediánový filtr je především dobrý, stejně jako bilaterální filtr, pro odstranění šumu z obrazu, ale je možné tento typ filtru použít i k supersamplingu. Jeho další výhodou je velká rychlost. Díky své jednoduchosti je tento filtr velmi rychlý.

Tento filtr se dá také velmi dobře využít pro inicializaci dalších algoritmů, které jsou třeba iterativní, nebo používají nějakou aproximaci supersamplingu k vlastní inicializaci, případně se tento filtr použije jako jednoduchá aproximace, ze které se následně detekuje hrana.

Supersamplingu se dosahuje jednoduše pomocí hustšího vzorkování vstupních dat. Každý pixel vstupních dat se tedy použije vícekrát. Tento algoritmus dosahuje lepších výsledků, pokud se kamera lehce posunuje a zároveň je posunutí známé. Tím je zajištěno, že ve vstupních datech je mnohem víc unikátních vzorků.

6.1.1 Bilaterální filtr

Bilaterální filtr [14] je filtr na redukci šumu, který zachovává hrany. Tento filtr používá obvykle gaussovu funkci k vyhlazení i k modifikaci koeficientu vyhlazování.

$$(G * F)[s,t] = \frac{1}{W_p} \sum_x \sum_y F[s,t]\dot{G}[s-x,t-y]\dot{b}(F[x,y]-F[s,t]) \tag{5}$$

$$W_p = \sum_x \sum_y G[s-x,t-y]\dot{b}(F[x,y]-F[s,t]) \tag{6}$$

Bilaterální filtr (5) je složen ze dvou členů. Spojením G a b vznikne sice nelineární konvoluční jádro, ale toto jádro má tu vlastnost, že upřednostňuje vzorky, které mají menší rozdíl intenzit. Protože vzniklo nelinární konvoluční jádro, je třeba zvlášť spočítat normalizační koeficient W_p.

Tento filtr je také možné použít pro supersampling. Filtr tak funguje podobně jako mediánový filtr. Některé další algoritmy potom používají složitější nebo modifikovaný člen b

a tím dosahují různých efektů.

6.2 LidarBoost [16]

Algoritmus LidarBoost je založený na minimalizaci vzorce (7), který se dá rozložit na dvě části. Vzorec je složený z datového výrazu a regulačního výrazu (9). Datový výraz řídí rekonstrukci tak, aby odpovídala vstupním datům. K tomu je výraz sestavený ze čtyř matic.

$$\text{minimize} \quad E_{data}(X) + \alpha E_{reg}(X) \tag{7}$$

Matice W_k kóduje pozice dat, které jsou použité pro rekonstrukci daného bodu. Tato matice je blízká jednotkové matici, protože obsahuje pouze jedničky a nuly tak, že odpovídá posunutí obrazů vůči sobě. Další matice je T_k. Tato matice obsahuje nuly na pozicích, které jsou podle čtení ze senzoru nespolehlivé. Tím se vyřadí vzorky, které neměly dostatečně silný signál a jejich hodnota má velkou pravděpodobnost chyby. Matice D_k je matice se vstupními daty. Tyto vstupní data už jsou ale transformovaná. Poslední matice je matice X, která obsahuje výsledný obraz.

$$E_{data}(X) = \sum_{k=1}^{N} \|W_k. * T_k. * (D_k - X)\|^2 \tag{8}$$

Transformace vstupních dat pro matici D_k se provádí jednoduchým převzorkováním na větší velikost, kde se použije nejbližší soused. Podle [16] se neosvědčilo používat interpolace, protože to způsobuje rozmazání a nižší přesnost výsledné rekonstruované mapy.

Regulační výraz (9) se skládá ze sumy gradientů, které jsou dělené jejich vzdáleností. Autoři se tímto snažili docílit vyhlazení gradientů v obraze. Tyto gradienty jsou reprezentované rozdílem (10). Autoři chtěli docílit reprezentace gradientů, která by byla invariantní vůči rotaci a zároveň byla jednoduchá na spočítání. Tím se zajistí i redukce šumu.

$$E_{reg}(X) = \sum_{u,v} \left\| \begin{matrix} G_{u,v}(0,1) \\ G_{u,v}(1,0) \\ \dots \\ G_{u,v}(l,m) \end{matrix} \right\|^2 \tag{9}$$

$$G_{u,v}(l,m) = \frac{X(u,v) - X(u+l, v+m)}{\sqrt{l^2 + m^2}} \tag{10}$$

Matice W_k, T_k a D_k jsou různé pro každý vstupní snímek. Pro reálné použití jsou tyto matice hodně velké. Autoři proto výpočet rozdělili do malých kousků obrazu 20×20, které se překrývají o dva pixely. K testování použili sérii syntetických i reálných scén, které nasnímali vlastní hloubkovou kamerou. Dobu výpočtu uvádí mezi pěti minutami až dvěmi hodinami.

6.2.1 Implementace

Algoritmus LidarBoost jsem implementoval s některými změnami. Například jsem nepoužil kvadratický optimalizátor, který je pro tento vzorec potřeba, ale použil jsem iterační gradientní optimalizaci. To také znamená, že nemohu použít proměnnou α, která řídí projev regulačního výrazu, s takovými hodnotami, které jsou napsané v původní zprávě. To je nejspíš způsobené nevýhodou iteračního přístupu, kde se místo ke globálnímu minimu, dostaneme k minimu lokálnímu.

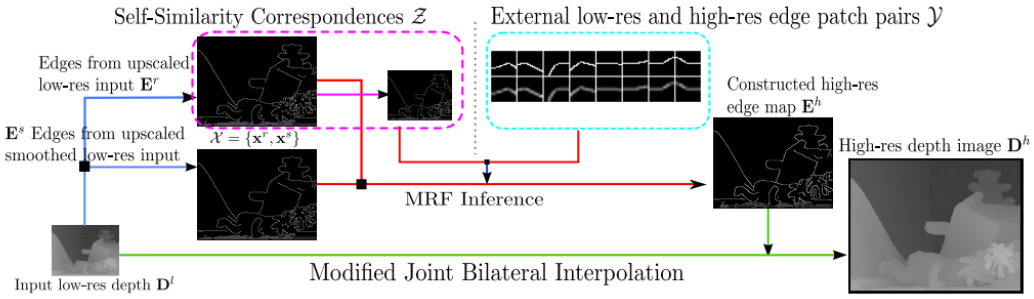
Při nastavení hodnoty α na vysokou hodnotu, začne převažovat regulační výraz nad datovým a obraz začne generovat šum okolo hran, kde se má nejvíce projevit vyhlazení, protože datový výraz na stejných místech způsobí aliasing.

Pro zrychlení iterativního přístupu jsem použil jako výchozí obraz výstup z mediánového filtru (Kapitola 6.1). Tím se výpočet značně urychlil.

6.3 Edge-Guided Single Depth Image Super Resolution [17]

Tento algoritmus je ze skupiny algoritmů, která používá naučené chování a díky tomu rekonstruuje snímky s lepší přesností. Algoritmus tedy potřebuje knihovnu s naučenými daty. Autoři ale také přemýšleli nad tím, že by knihovna s naučenými daty nebyla dostupná a studovali i možnost vytvořit si taková data dočasně pomocí hledání ve zmenšených datech.

Algoritmus nejprve pomocí bikubické interpolace zvětší vstupní data na velikost výstupních. Potom se aplikuje detektor hran. Detekované hrany se dále upravují klasifikátorem, který má k dispozici páry charakteristických hran ze zvětšených obrázků a z obrázků nasnímaných ve vyšším rozlišení. Autoři si od tohoto postupu slibují velké zpřesnění detekované hrany. Takto upravené hrany poté slouží k řízení modifikovaného bilaterálního filtru tak, aby ignoroval vzorky, které jsou na druhé straně hrany.



Obr. 8: Schéma algoritmu Edge-Guided Single Depth Image Super Resolution z [17]

Hrany se zpracovávají klasifikátorem, který používá Markovské náhodné pole. Pro klasifikování hran jsou k dispozici dva postupy. Jeden postup funguje s předem naučenými daty, která jsou uložená ve formě párů hran. Jedny hrany jsou supersamplované z nižšího rozlišení a druhé jsou z dat s vysokým rozlišením. Pokud nejsou dostupné tyto páry hran, je možné spustit algoritmus s použitím vstupu samotného. Pro použití obrazu samotného, nejprve se obraz zmenší v poměru, v jakém chceme výsledek zvětšit. Na takto změnšený obraz a vstup se aplikuje algoritmus učení, který vygeneruje páry hran, které jsou použité později stejným způsobem, jako již naučené. Tímto způsobem autoři zajistili, že není nutné ke spuštění algoritmu mít soubor s naučenými daty.

Pokud máme zvětšený obraz s hranami, provede se nad ním algoritmus bilaterálního filtru, který je řízený geodetickou vzdáleností dvou bodů, což je vzdálenost po křivce, ležící uvnitř objektu. Nejkratší geodetická vzdálenost tedy odpovídá nejkratší cestě mezi oběma body, když nesmíme překročit hranu. Touto metrikou se řídí bilaterální filtr místo rozdílu intenzit bodů ve druhém členu.

6.4 Similarity-Aware Patchwork Assembly for Depth Image Super-resolution [18]

Tento algoritmus používá přímou rekonstrukci pomocí databáze vzorků a následnou minimalizaci energie, kterou obraz vyhladí. Minimalizační výraz má čtyři části.

$$E_{total} = E_{data} + \lambda_l \cdot E_{label} + \lambda_d \cdot E_{depth} + \lambda_s \cdot E_{dimilar} \tag{11}$$

Datová část (12) se stará o to, aby byly vybrány vzorky tak, aby obraz odpovídal vstupu.

Tento výraz je podobný prvnímu výrazu z algoritmu LidarBoost (8). Člen D^H odpovídá obrazu ve vysokém rozlišení a člen D^L odpovídá obrazu v nízkém rozlišení. Funkce $u(x)$ mapuje bod pomocí metody nejbližšího souseda.

$$E_{data} = \sum_{x} e^{-\alpha\varphi(x)} \mid D^H(x) - D^L(u(x)) \mid \tag{12}$$

$$\varphi(x) = \sum_{v \in N_{u(x)}} \mid D^L(u(x) - D^H(x)) \mid \tag{13}$$

Druhý výraz (14) je pro vyhlazení přechodů mezi jednotlivými vzorky. Tento výraz je zvláštní tím, že obsahuje binární proměnnou δ, která tento výraz aktivuje pouze tehdy, když jsou dva zpracovávané pixely z jiných vzorků.

$$E_{label} = \sum_{x} \sum_{y \in N_x} \sigma(\varphi(x,y))\delta(L(x) \neq L(y)) \tag{14}$$

$$\varphi(x,y) = \frac{\varphi(x) - \varphi(y)}{2} \tag{15}$$

$$\sigma(w) = \frac{1}{1 - e^{-\alpha w}}; \delta(true) = 1; \delta(false) = 0 \tag{16}$$

Třetí výraz 17 upravuje hloubky vzorků, které leží vedle sebe. Tento výraz je důležitý proto, aby jednotlivé vzorky na sebe navazovaly, přestože mají v databázi rozdílné hloubky.

$$E_{depth} = \sum_{x} \sum_{y \in N_x} e^{-\alpha\varphi(x,y)} \mid D^H(x) - D^H(y) \mid \tag{17}$$

Čtvrtý výraz 18 je určený pro hledání podobností v obraze samotném a k oddělení jednotlivých mapování vzorků.

$$E_{similar} = \sum_{(x_i,x_j) \in S} [s(x_i,x_j) \cdot \sigma\left(\frac{\varphi(x_i) - \varphi(x_j)}{2}\right) \cdot \delta(k_i \neq k_j | x_i + d_i \neq x_j + d_j)] \tag{18}$$

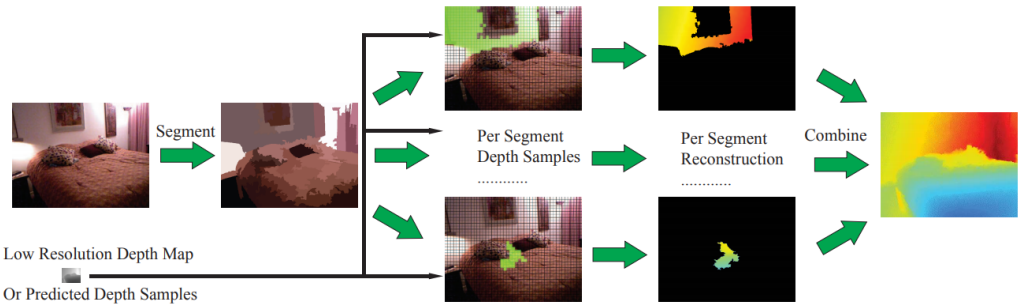
V dokumentu autorů mají napsané rozšíření některých členů a funkcí pro použití s daty barevné fotografie.

K nalezení řešení autoři použili kombinaci sumy rozdílů čtverců (SSD = Sum of Squared Differences) a metodu řezů v grafu. Pomocí SSD zredukovali počet vzorků z databáze na

nutný počet k sestavení obrazu a následnou minimalizaci energie provádí pomocí optimalizace založené na metodě maximálních toků. Tato metoda se tím řadí mezi ty nejsložitější, co existují.

6.5 Sparse Depth Super Resolution [19]

Přístup v tomto článku kombinuje všechny možné přístupy a tím dosahují až zvětšení 64×. Toho všeho je ale dosažené hlavně použitím barevného obrazu, který je ve vysokém rozlišení. Přesto mě zaujal tento přístup, právě kvůli kombinaci použitých metod.



Obr. 9: Schéma algoritmu Sparse Depth Super Resolution z [19]

Vstupní data se nejprve segmentují podle barevného obrazu. Segmenty jsou vytvářené pomocí klastrování podobných barev a výstup této metody je strom, který obsahuje jednotlivá dělení vstupních dat na segmenty.

Segmenty je potom možné paralelně zpracovat za pomocí vzorků hloubky, které jsou přístupné z libovolně zmenšené hloubkové mapy. Pomocí těchto vzorků se vytvoří vzorky v jednotlivých segmentech a ty se následně vyhladí. Touto metodou je možné získat ostré hrany a hladké plochy. Článek popisuje dva algoritmy na interpolaci mezi vzorky. Jednodušší, který je rychlejší a složitější, který si lépe poradí s detaily.

Pro účely mé práce ale tato metoda byla vyřazena kvůli požadavku na nepoužití barevného obrazu pro účely výpočtu.

7 Vstupní data

Na testy mi byla zapůjčena hloubková kamera Creative. Tato kamera snímá hloubkové mapy s rozlišením maximálně 320×240. Bohužel se mi ale nepodařilo tuto kameru zprovoznit

kvůli příliš starým ovladačům a příliš novému operačnímu systému, který už tuto kameru nepodporuje. I přes instalaci doporučených ovladačů, se kterými kamera fungovala se mi na mém počítači nepodařilo získat hloubkový obraz, přestože barevný senzor přístroje fungoval bez problémů.

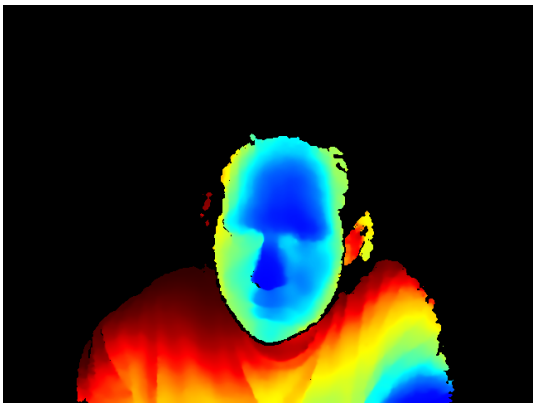
Později mi byla zapůjčena novější hloubková kamera Creative BlasterX SENZ3D, ze které se mi úspěšně podařilo získat data, která použiji později. Protože je kamera novější, má také vyšší rozlišení a to až 640 × 480. Na fotografii jsou vidět obě kamery systému. Levá je infračervená 3D kamera a pravá je kamera snímající barevné spektrum. Dále vpravo je pak vidět černý kruh s emitorem infračerveného záření. Tato kamera funguje přesně jak bylo popsáno v kapitole 3.1.

Obr. 10: Fotografie kamery Creative BlasterX SENZ3D

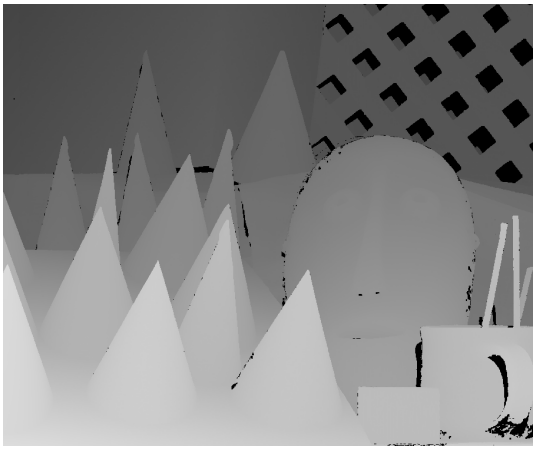
Také jsem použil testovací soubory z různých zdrojů. Tyto soubory už jsou zbavené šumu a mají jen předem připravená rozlišení a případná zmenšení. Jejich výhoda je, že se dají použít jako takzvaný ground truth. K testům jsem použil jen některé soubory z testovacích sad, protože tyto testovací sady jsou rozsáhlé a scény se často opakují, jen jsou snímané z jiných úhlů.

Na ukázce (Obr. 11) je vidět rozdíl mezi sejmutým obrazem a vyčištěnou mapou. Sejmutému obrazu chybí pozadí, přestože ukazatel v programu ukazoval, že by kamera měla dosáhnout až šestnáct metrů, ale na obraze není vídět ani opěradlo židle u které jsem seděl, když jsem obraz pořizoval. Navíc obraz je viditelně složený z náhodných útvarů, které na sebe navazují.

Vyčištěný obraz má naopak pěkně definované hrany, pozadí a tvary odpovídají. Tyto obrazy jsou totiž nejen nasnímané jiným snímačem, ale navíc jsou ještě ručně zpracované, aby byly velmi kvalitní.



(a) Sejmutá mapa z kamery Creative BlasterX SENZ3D



(b) Vyčištěný obraz ze sady Middleburry [20]

Obr. 11: Ukázka rozdílu kvality mezi vyčištěným obrazem, použitelným jako ground truth a obrazem, který je sejmutý hloubkovou kamerou.

7.1 Ground truth

Ground truth je termín, kterým se označují data, která jsou považovaná za pravdivá. Tato data jsou důležitá, protože mi dovolí porovnat jednotlivé algortmy a jejich přesnost. Tyto data jsou buď ručně předzpracovaná, nebo měřená laserovým měřením v největším možném rozlišení. Proto je možné tyto data použít i na testování přesnosti.

Největší z testovacích sad je Middlebury sada [20]. Tato sada je velmi často používaná k testování těchto algoritmů, přestože byla vytvořená za účelem rozpoznávání a registrace ve stereo obrazech, které vznikají ze dvou hloubkových kamer, které jsou vůči sobě posunuté o určitou vzdálenost, která je známá. Podařilo se mi najít jeden dataset, který nebyl úplně

vyčištěný od šumu, ale tento dataset postrádá data ve vysokém rozlišení. Z důvodů rychlosti jsem pro potřeby analýzy použil pouze data o velikosti 1/4 originálů v plném rozlišení. Algoritmus EdgeGuided pro plné rozlišení běžel přibližně 20 minut na jednu hloubkovou mapu. Výstupní data tedy mají přibližně velikost výstupů ze samotných senzorů.

Vstupní data pro algoritmy vytvářím z vybraných hloubkových map tak, že zvolím násobek zvětšení k, které budu provádět a vytvořím sérii vstupních dat, které jsou zmenšené přesně v daném násobku k. Každý takto vytvořený obraz ještě posunu náhodně o zlomek pixelu ve zmenšeném obraze. Tím získám vstupní data i jejich zarovnání, které by ideálně mělo vytvořit zpět použitý obraz s originálním rozlišením.

8 Analýza implementací

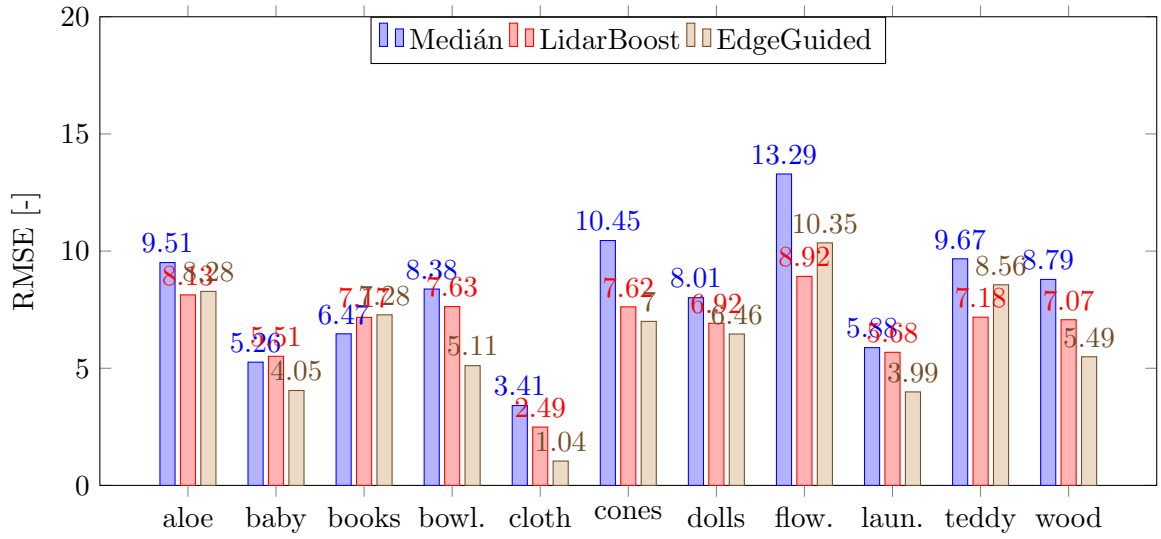
Pro porovnání výsledků jsem použil několik metod. Vizuální porovnání výsledků provádím pomocí rozdílových obrazů. Pro analýzu rekonstruovaných obrazů se často používá RMSE (root mean square error). Tento ukazatel se počítá jako průměr z odmocniny středních odchylek jednotlivých obrazových bodů. Dále jsem spočítal procenta špatných pixelů.

Z algoritmů, které jsem blíže probral v sekci 6 jsem pro potřeby analýzy jeden implementoval a u jednoho našel implementaci a použil ji. Tyto dva algoritmy jsou LidarBoost (Kapitola 6.2), který jsem implementoval s určitými omezeními a Edge-Guided Single Depth Image Super Resolution (Kapitola 6.3), který měl přístupnou implementaci přímo od autorů článku. U LidarBoost algoritmu jsem zvolil jinou optimalizační metodu, než použili původně autoři. Autoři původně použili Matlab, který jim dovolil maticový zápis problému a následně použili optimalizační algoritmy, které Matlab nabízí pro kvadratickou optimalizaci. Protože jsem psal program v C++, tak jsem použil jednodušší metodu optimalizace problému. Kvůli tomu mi vychází špatné výsledky při použití proměnných, které jsou zmíněmé v jejich práci [16].

Pro všechny testy jsem použil stejné argumenty vstupních parametrů. U metody LidarBoost jsou vstupní argumenty velikost regulačního výrazu, kterou jsem nastavil na 5 a parametr λ z rovnice (9) jsem nastavil na 1.25. Pro metody EdgeGuided jsem nastavil parametr *threshold*, který řídí především detekci hran, na hodnotu 0.1, okolo které se pohybovaly nastavené hodnoty v původní implementaci, které byly jiné pro některé testovací obrázky.

soubor	Median		LidarBoost		EdgeGuided	
	% špatných px	RMSE	% špatných px	RMSE	% špatných px	RMSE
aloe	19.2	9.51	30.4	8.13	15.6	8.28
baby	7.5	5.26	13.0	5.51	5.5	4.05
books	13.8	6.47	21.0	7.17	11.8	7.28
bowling	11.0	8.38	25.1	7.63	8.5	5.11
cloth	6.8	3.41	16.8	2.49	3.0	1.04
cones	10.9	10.45	21.6	7.62	9.9	7.00
dolls	20.9	8.01	31.7	6.92	18.9	6.46
flowerpots	13.3	13.29	23.2	8.92	10.1	10.35
laundry	15.9	5.88	25.6	5.68	16.6	3.99
teddy	11.7	9.67	16.8	7.18	8.8	8.56
wood	6.8	8.79	19.2	7.07	5.1	5.49

Tab. 1: Porovnání výsledků algoritmů na vybraných hloubkových mapách.

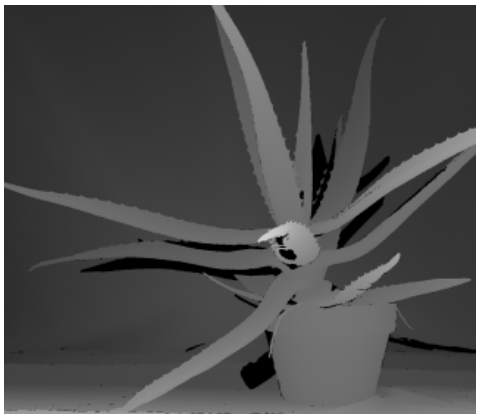


Obr. 12: Porovnání přesnosti použitých algoritmů

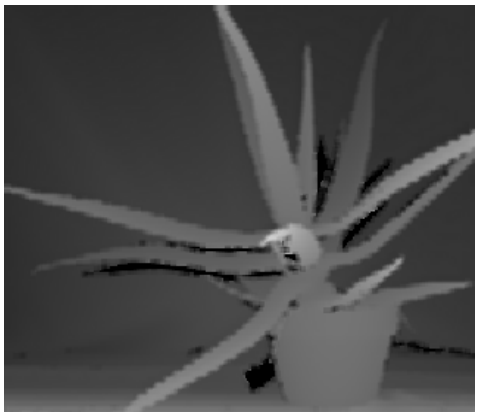
Z tabulky (Tab. 1) a grafu (Obr. 12) je vidět, že každý z algoritmů je lepší na jiné hloubkové mapy. Dokonce i algoritmus, který používá pouze jednoduchý medián byl nejlepší na dvou z jedenácti použitých map. Pozorováním výsledků měření v závislosti na členitosti map jsem vypozoroval, že implementace algoritmu LidarBoost je většinou lepší na mapách, které mají složitější hrany. Toto může být způsobené tím, že jsem smíchal dohromady tři

různé datové sady, ale k učení EdgeGuided algoritmu byla použitá pouze jedna datová sada. To by znamenalo, že v databázi chybí některé páry hran, které se objevují v datech. Na datech, která obsahují málo hran a velké plochy (například cloth) je algoritmus EdgeGuided lepší.

Časy algoritmů se dají měřit pouze velmi přibližně, protože jednotlivé implementace jsou v jiných programovacích jazycích. Algoritmus mediánu, protože je nejjednodušší, nezabral více, než pár milisekund. Složitější algoritmus LidarBoost zabral podle délky gradientní optimalizace desítky sekund. Nejpomalejší byl algoritmus EdgeGuided, kterému jedna hloubková mapa zabrala jednotky minut. To může být způsobené tím, že je algoritmus implementovaný jako Matlab skript, ale daleko pravděpodobnější je, že samotná klasifikace je velmi pomalá.

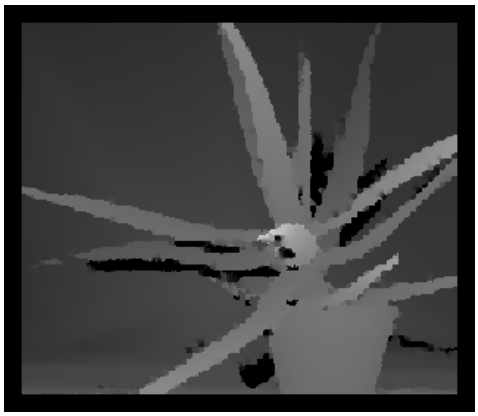


Obr. 13: Hloubková mapa aloe. Originál z datové sady.



Obr. 14: Hloubková mapa aloe. Výstup algoritmu LidarBoost.

Z obrázků (Obr. 13, 14 a 15) je vidět rozdíl v jednotlivých výstupech. Nevýhoda přístupu



Obr. 15: Hloubková mapa aloe. Výstup algoritmu EdgeGuided.

EdgeGuided je patrná na první pohled, protože implementace generuje výstup, kterému chybí rámeček o velikosti $2 \cdot window$, kde $window$ je proměnná, která určuje velikost segmentu mapy, která se klasifikuje zvlášť. Výsledek z algoritmu LidarBoost je viditelně příliš vyhlazený oproti výstupu z EdgeGuided implementace, ale nejsou v něm artefakty po špatné klasifikaci.

9 Návrh implementace

Analýza mi ukázala, že žádný z přístupů není výrazně lepší. Další poznatek je, že podstatně složitější přístupy nejsou mnohdy o moc lepší, než jednodušší přístupy a občas je nejjednodušší přístup lepší. Také jsem zjistil, že zkoušet klasifikaci trvá zbytečně dlouho a bude lepší se jí v budoucí implementaci vyhnout, obzvlášť pokud chci upřednostnit rychlost.

Rozhodl jsem se použít kombinaci nápadů z vyzkoušených algoritmů a zkombinovat je jiným způsobem. Jako první použiji základní rychlý supersampling pomocí metody mediánu 6.1. Poté použiji algoritmus pro detekci hran. Tento algoritmus je velmi často používaný v implementacích supersamplingu hloubkových map. Já tento algoritmus použiju k omezení interpolace pro vytvoření výsledného obrazu. Teoreticky by mi toto mělo dovolit udělat ostré hrany a hladké gradienty hloubkové mapy. Zároveň, protože jsem se vyhnul klasifikaci, by tento přístup měl být rychlý.

9.1 Detekce hran

Detekce hran je algoritmus jednoduché lineární filtrace. Detekce hran samotná je ale netriviální problém a plánuji použít více způsobů.

Protože hloubková mapa není binární objekt, tak stejně jako u fotografií a obrázků, je složité říct, kde je hrana. Pro program je složité rozeznat, jestli se jedná o hranu, nebo jen o strmý gradient. Hrana navíc může být široká víc než jeden bod, stejně jako může ležet mezi dvěma body a jediná identifikace ležící hrany je velká změna dvou bodů, které leží vedle sebe.

Detekce hran se provádí pomocí konvoluce s relativně malým jádrem a následným zpracováním výsledků. Většina detektorů hran potřebuje dvě konvoluce, protože mají nesymetrické jádro. Vyjímka z tohoto je laplaceův detektor a detektory, které používají druhou derivaci gaussovy funkce, které ale potřebují složitější zpracování výsledků.

Složitější metody detekce hran potom pracují s rozdíly ve fázové části obrazu k detekci různých prvků v obraze. Tyto metody jsou ale velmi složité.

Většina detektorů hran používá nějakou aproximaci první derivace gaussovy křivky. Tyto detektory potřebují dva průchody, protože při druhém průchodu je detektor otočený ve směru druhé osy, protože tyto detektory, založené na první derivaci, nejlépe detekují hrany pouze v jedné ose. Výsledkem je velikost gradientu v daném směru. Složením obou směrů je možné získat přesný směr hrany. Protože máme dvourozměrný prostor, je třeba druhý průchod s detektorem pootočeným o určitý úhel, nejlépe pravý. Toho se nejsnáze dosáhne tím, že se jednotlivé jádra zarovnají s osami snímku. Tím se dostatečně pokryjí hrany a následné zpracování je často pouze jen druhá odpocnina ze součtu druhých mocnin výsledků.

9.1.1 Sobelův detektor

Rozhodně budu implementovat tento známý a jednoduchý detektor hran. Jádro pro konvoluci je známé a danné (Tab.2). Existují variace tohoto jádra, ale základní myšlenka je stejná. Detektor je jednoduchou aproximací detektoru, který vychází z první derivace gaussovy funkce.

1	0	-1
2	0	-2
1	0	-1

(a) G_x

1	2	1
0	0	0
-1	-2	-1

(b) G_y

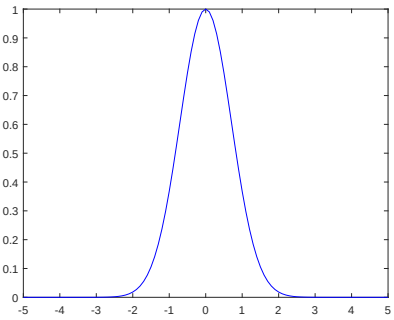
Tab. 2: Obě jádra sobelova detektoru hran

Tento detektor je velmi oblíbený pro svou jednoduchost. Díky své jednoduchosti je také

velmi rychlý. Protože má malé jádro, bude nutné použít nějaký algoritmus vyhlazení obrazu, protože předpokládám, že ve výsledku bude šum a takto jednoduchý detektor hran na takový šum obvykle reaguje jako na skutečnou hranu.

9.1.2 První derivace Gaussovy funkce

Gaussova funkce (Obr. 16) se běžně používá v matematice a statistice. Tato funkce (19) se poté dá jednoduše rozšířit do 2D prostoru (20).



Obr. 16: Gaussova funkce se $\sigma = 1$.

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{19}$$

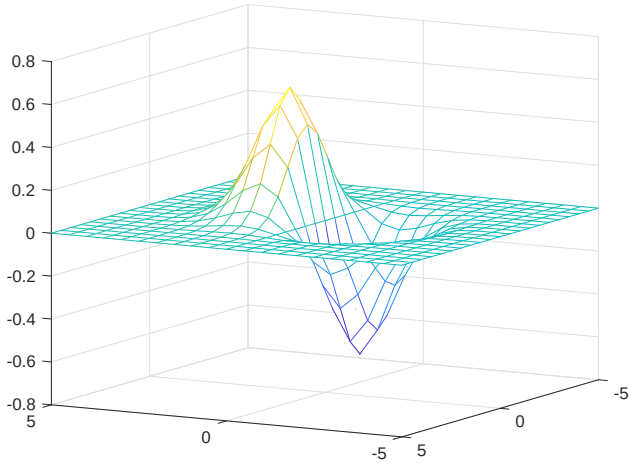
$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{20}$$

Protože můžeme jádro konvoluce předpočítat, není důležité znát přesně všechny konstanty, když zároveň víme, že součet všech prvků jádra musí být roven 1. Proto se můžeme v parciální derivaci soustředit pouze na proměnné x a y. Jádro můžeme, a okolo okrajů dokonce musíme, normalizovat, pokud chceme počítat konvoluci až k okrajům.

Obrázek (Obr. 17) nám ukazuje jaký tvar má jádro první derivace gaussovy funkce.

$$\frac{\partial G_\sigma(x,y)}{\partial x} \propto x e^{-\frac{x^2+y^2}{\sigma^2}} \tag{21}$$

$$\frac{\partial G_\sigma(x,y)}{\partial y} \propto y e^{-\frac{x^2+y^2}{\sigma^2}} \tag{22}$$



Obr. 17: První derivace gaussovy funkce ve 2D prostoru se $\sigma = 1$.

Tyto parciální derivace generují dvě jádra pro konvoluce. Tyto konvoluce vytvoří z původního obrazu F obrazy G_x a G_y. G_x a G_y obsahují velikost gradientu intenzity ve směrech os x a y, a dají se pomocí následujících funkcí převést na velikost hrany G a směr hrany, úhel, Θ.

$$G = \sqrt{G_x^2 + G_y^2} \tag{23}$$

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{24}$$

Tento postup se také nazývá Canny Edge detekce, podle svého autora [22]. Díky svému tvaru již není nutné používat vyhlazení, protože samotný detektor má tvar, který provádí vyhlazení během detekce.

Široké hrany

Tyto detektory hran dost často detekují hrany širší, než skutečně jsou. Hrany se mohou rozprostírat na více obrazových bodů, jenže také to mohou být plochy, které mají velký gradient. Tyto plochy můžou tyto detektory označit jako hranu, nebo široké hrany mohou označit celé. Proto se používají algoritmy, které se snaží tyto široké hrany ztenčit, ideálně na jediný obrazový bod. Existuje několik možností, z toho některé zkusím implementovat a vyzkoušet, jak moc jsou vhodné pro použití. Asi nejrozšířenější je algoritmus Non-maxima suppression

(potlačení ne-maximálních hodnot), který se snaží vynulovat hodnoty, které nejsou v rámci okolí maximální.

Non-maxima suppression

Tento algoritmus je asi nejběžněji používaný pro zlepšení detekce hran. Funguje na principu potlačení intenzity detekované hrany v bodech, které nejsou v daném směru maximální.

Nejprve se v algoritmu zjednoduší směr hrany na diskrétní hodnoty pro 4 směry - vertikální, horizontální, nebo diagonální. Toho se dosáhne zaokrouhlením směru na nejbližší násobek 45° při použití modula pro omezení rozsahu hodnot úhlů na $\langle 0; 180)$ stupňů, nebo také $\langle 0, \pi)$.

Následně se zkontroluje, zda body, které jsou kolmo ke směru hrany, jsou menší než právě zkoumaný bod. Pokud se zjistí, že některý z vedlejších bodů je větší, zkoumaný bod se vynuluje, protože v jeho blízkém okolí existuje bod, který reprezentuje silnější hranu.

Zhang-Suen algoritmus [23]

Tento algoritmus je určený pro binární obrazy, například jako součást algoritmu na optické rozpoznávání textu (OCR). Při použití prahování je ale možné tento algoritmus použít i v tomto případě.

Body v obraze jsou rozdělené na černé a bílé. Bílá značí nulové hodnoty a černá značí nenulové, v originále body s hodnotou 1.

Algoritmus počítá pro každý bod dvě metriky z okolních osmi bodů, značených obvykle podle následující tabulky 3. Počítá počet černých bodů v okolí jako $B(P1)$ a poté počítá počet změn z bílé na černou ve směru hodinových ručiček jako $A(P1)$. Následně jsou tyto metriky porovnané se dvěma sadama podmínek a nevyhovující body jsou označené jako bílé. Algoritmus se opakuje tak dlouho, dokud v iteraci existují body, které nevyhověly podmínkám a tak byly označené jako bílé.

$$A(P1) = \sum (0 \to 1) pro P2, P3, ..., P9, P2 \tag{25}$$

$$B(P1) = \sum (P2, P3, ..., P9) \tag{26}$$

P9	P2	P3
P8	P1	P4
P7	P6	P5

Tab. 3: Zkoumané body v algoritmu. P1 je zkoumaný bod, P2-P9 jsou okolní body.

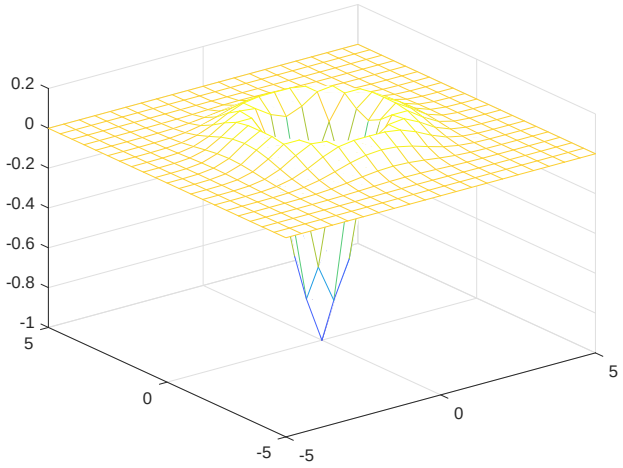
Oba sety podmínek jsou téměř stejné, až na poslední dvě podmínky. První podmínka je, že zkoumaný bod musí být černý. Další podmínka je, že počet okolních černých pixelů můsí být v rozsahu $\langle 2; 6 \rangle$. Dále musí být počet přechodů z bílé na černou rovný 1. Další dvě podmínky jsou různé v každém setu. Každá podmínka zkoumá jinou trojici bodů a zda je alespoň jeden z nich bílý. První set zkoumá trojice bodů P2, P4, P6 a P4, P6, P8. Druhý set zkoumá trojice bodů P2, P4, P8 a P2, P6, P8.

Tento algoritmus má především výhodu v tom, že je možné ho jednoduše paralellizovat. Problém ale je, že je velmi agresivní a eliminuje některé hrany, které nejsou spojené na koncích s nějakou další hranou, jak je možné vidět v příloze v Obr. 29.

9.1.3 Detekce pomocí druhé derivace [24]

Tyto metody používají nejčastěji laplacián (27) a hledání průchodu nulou. Tyto metody jsou rychlejší, protože používají pouze jednu konvoluci. Zato jsou citlivé na šum a je třeba pro detekci průchodu nulou použít dvanáct různých masek, aby se ošetřily všechny možné případy, které mohou nastat. Osm masek pro všech 8 směrů, pokud je detekovaná přímo nula uprostřed a potom další čtyři pro případy, kdy průchod nulou leží mezi obrazovými body, což se projeví záporným a kladným číslem, které leží vedle sebe.

$$\bigtriangledown G_\sigma(x, y) = \frac{\partial_2 G_\sigma(x, y)}{\partial_2 x} + \frac{\partial_2 G_\sigma(x, y)}{\partial_2 y} \tag{27}$$



Obr. 18: Vizualizace laplaciánu (27) ve 2D prostoru se $\sigma = 1$.

9.2 Rekonstrukce

Rekonstrukci provedu nějakou metodou, která uvažuje předem detekovanou hranu z předchozího stupně. Mám několik možností, které mohu použít. Například se mohu inspirovat metodou, která byla použita v implementaci, probrané v kapitole 6.3. V této implementaci autoři použili modifikaci bilaterálního filtru, kde jako koeficient velikosti rozdílu intenzit byla použita vzdálenost bodu od středu jádra s uvažováním již detekované hrany.

Také mě napadla jednoduchá aproximace této metody, která se dá použít na původní jednoduchou aproximaci výsledku pomocí mediánu, kdy pro každý bod hrany zkontroluji, zda v blízkém okolí je správná intenzita/hloubka a pokud ne, použiji extrapolaci k doplnění správné intenzity až k hraně.

10 Postup Implementace

Implementaci jsem se rozhodl provést v programovacím jazyce standardu C++11. Jako vývojové prostředí jsem použil Visual Studio 2015. Samotný algoritmus jsem se rozhodl implementovat jako knihovnu. Aby bylo možné tento algoritmus testovat, spouštět a předvádět, v tom samém vývojovém prostředí a za pomoci stejného standardu jazyka jsem implementoval aplikaci, která toto dovoluje. K implementaci aplikace jsem použil knihovnu OpenGL (konkrétně generátor hlavičkového souboru GLAD, který zajistí načtení funkcí kni-

hovny OpenGL) k zobrazení výsledků. Dále jsem použil knihovnu GLM (OpenGL Mathematics) a STB knihovnu k načtení a ukládání obrázků. Protože neexistuje standardní postup pro práci s okny v jazyce C++, předváděcí aplikaci jsem implementoval pouze pro operační systém Windows.

Jako základ pro předváděcí aplikaci jsem použil projekt, který jsem používal již dříve na jiné projekty. Tento projekt uměl vytvořit okno, které mělo inicializovanou knihovnu OpenGL. Dále měla tato aplikace třídy na vykreslování textu a obdélníků s texturou. Také jsem k této aplikaci přidal svůj vlastní systém na zpracování asynchronních úkolů. Ten jsem použil k asynchronnímu spouštění generování nových obrazů a spouštění algoritmu knihovny.

Jako první při implementaci samotné knihovny jsem navrhnul funkce, které se budou volat. Knihovnu jsem navrhl tak, aby používala pouze jednu instanci. Funkce jsou pro jednoduchost pouze v jazyce C a předávané jsou pouze základní typy proměnných. Tím se vyhnu problémům se zarovnáním dat ve strukturách a třídách. Použití knihovny se také tímto rozhodnutím stane jednodušší a zároveň bude možné tuto knihovnu používat i s jazyky, které nepodporují třídy.

Definoval jsem také konstanty, které dovolují knihovnu použít pro různé formáty vstupních dat. Předpokládám, že hlavní typ bude hloubka s rozlišením 8 bitů, ale je možné použít i 16 a 32 bitů rozlišení v hloubce. Také jsem definoval konstanty na inicializaci pro různé způsoby výpočtů, především pro použití více vláken, a nebo grafické karty. Také jsem definoval konstanty na řízení jednotlivých fází výpočtu. Je tak možné zablokovat finální vyhlazení, nebo vrátit pouze detekovanou hranu. Tyto proměnné jsou ale spíše pro účely testování, protože vyřazují jednotlivé fáze algoritmu.

Během implementace knihovny jsem narazil na problém s poslední fází rekonstrukce. Protože jsem nepoužil modifikovaný algoritmus bilaterálního filtru, zkusil jsem několik aproximací a nakonec jsem úspěšně naimplementoval již zmíněnou aproximaci. První verze měla fungovat pomocí vzorkování, kdy jsem do předpřipraveného pole dopřednou projekcí transformoval body z původních snímků a následně dvěmi průchody jsem našel nejbližší body v ose x a y, a zároveň jsem pomocí interpolací doplnil chybějící body. Tato verze fungovala dobře, dokud jsem nezačal implementovat respektování hrany a obraz mi tvořil prázdné pruhy okolo hran a při extrapolacích i body, které byly naprosto nesmyslné. Tento návrh měl také zjevnou chybu, která by se projevila při vzorkování mimo mřížku (při použití pohyblivé kamery). Proto jsem

tento způsob implementace zavrhl a začal znovu.

Jako další způsob jsem použil vyhledávání tří nejbližších bodů s respektováním nalezené hrany. Algoritmus je na papíře jednoduchý, ale v praxi jsem nebyl schopný ani v řádu minut získat jediný snímek, což je vzhledem k momentální rychlosti algoritmu velké zpomalení.

Nakonec jsem použil velmi jednoduchou aproximaci, která generuje přijatelnou chybu, vzhledem k jednoduchosti algoritmu. Zároveň to ale znamená, že tato aproximace je velmi rychlá.

Protože hrana detekovaná ve výsledku jednoduchého mediánového supersamplingu byla příliš vzdálená od hrany v původním obraze a obsahovala příliš velký aliasing, předřadil jsem ještě před detekci hrany pomocí Sobelova jádra jednoduché vyhlazení pomocí konvoluce s malým jádrem gaussovy funkce. Toto je běžný postup k vyhlazení obrazu, který se používá ve zpracování obrazu. Při použití první derivace Gaussovy funkce toto není potřeba.

Také jsem zařadil do algoritmu prahování hrany. Pro algoritmus Zhang-Suen je toto nutné, aby bylo možné použít tento algoritmus, který je určený pro binární vstup. Při použití tohoto algoritmu se nejprve provede prahování a poté se provede algoritmus na ztenčení hrany. Při použití algoritmu Non-maxima suppression jsem také použil prahování, ale až v pozdější fázi, kdy pokud je hrana opravdu maximální v daném bodě a bude zachována, tak hranu zahodím, pokud je její intenzita menší než daný práh.

11 Výsledky

Pro test přesnosti jsem pouze lehce rozšířil původní sadu pro analýzu. Přidal jsem několik testovacích snímků z vyčištěných datasetů a také jsem zkusil jeden obrázek ze zapůjčené kamery. Také jsem testoval zpracování pro různé parametry tak, abych našel nejlepší variantu. Zde už nebudu porovnávat můj algoritmus s existujícími implementacemi, ale pouze s jednoduchým zpracováním pomocí mediánu. Pro vybrané snímky poté ukáži vliv počtu snímků na výsledek.

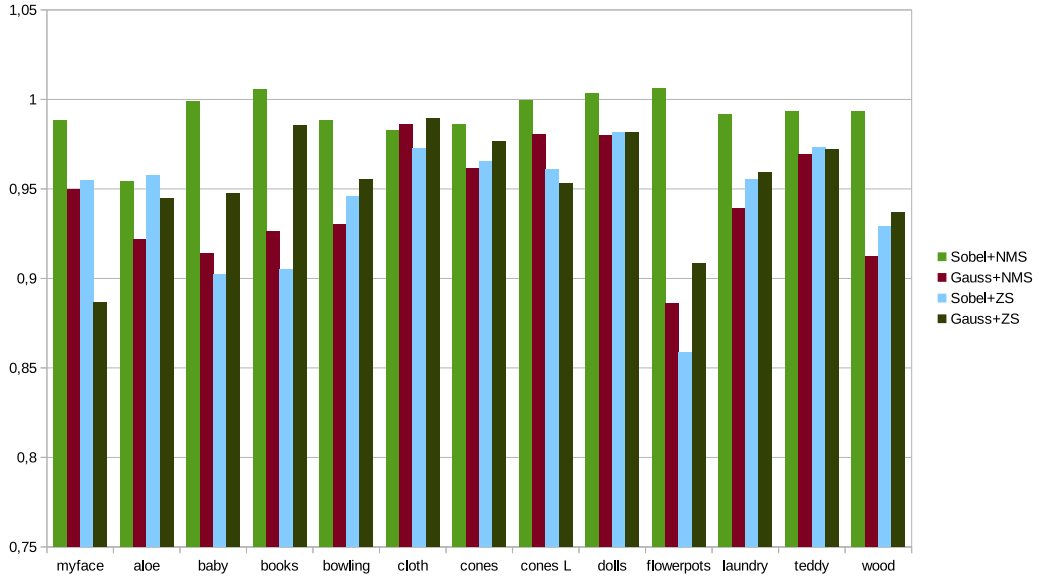
Pro všechny metody analýzy výsledků jsem zvolil zvětšení $4\times$, protože je to nejčastěji používané zvětšení. To znamená, že výsledek má $4\times$ více pixelů, než původní snímek. Toho se lehce dosáhne tím, že se výška i šířka vstupu zdvojnásobí. Knihovna je ale schopná zpracovat všechny možná zvětšení, včetně těch deformujících, jako například zvětšení pouze v ose x.

11.1 Porovnání s mediánem

V této části analýzy výsledků jsem se zaměřil na nalezení správných parametrů pro dané postupy. Pro ohodnocení výsledků jsem použil metriku RMSE, kterou jsem popsal již v kapitole 8. Tento ukazatel mi ukáže průměrnou střední odchylku od originálu v celém obraze. Tabulka 19 ukazuje poměr mezi výslednou metodou a mediánem. Pokud je hodnota rovna jedné, metoda funguje stejně jako medián. Menší hodnota znamená lepší výsledek.

Pro každý snímek jsem předgeneroval 50 zmenšených snímků, kterým jsem přidal náhodný šum s rozptylem $\sigma = 0.05$ s normálovým rozložením. Tyto sady snímků jsem nahrál do knihovny a spustil výpočet. Pro různé parametry jsem potom vybral ty nejlepší výsledky.

Tabulka 19 ukazuje, že všechny metody, až na vyjímečné případy měly po nalezení správných parametrů lepší výsledky, než medián. Z grafu je patrné, že zlepšení vůči mediánu je možné až o téměř 15%. Například použití detekce hrany pomocí Sobelova filtru a následné ztenčení hrany algoritmem Zhang-Suen vede k průměrně o 5% lepším výsledkům. Podobné zlepšení ukazuje i detekce hrany pomocí Canny Edge detektoru, kdy jde o spojení konvoluce s jádrem, které obsahuje první derivaci gaussovy funkce a ztenčení hrany pomocí Non-maxima suppression algoritmu.



Obr. 19: Porovnání postupů vůči mediánu. Na ose y je poměr, o kolik byl daný postup lepší s ideálními parametry.

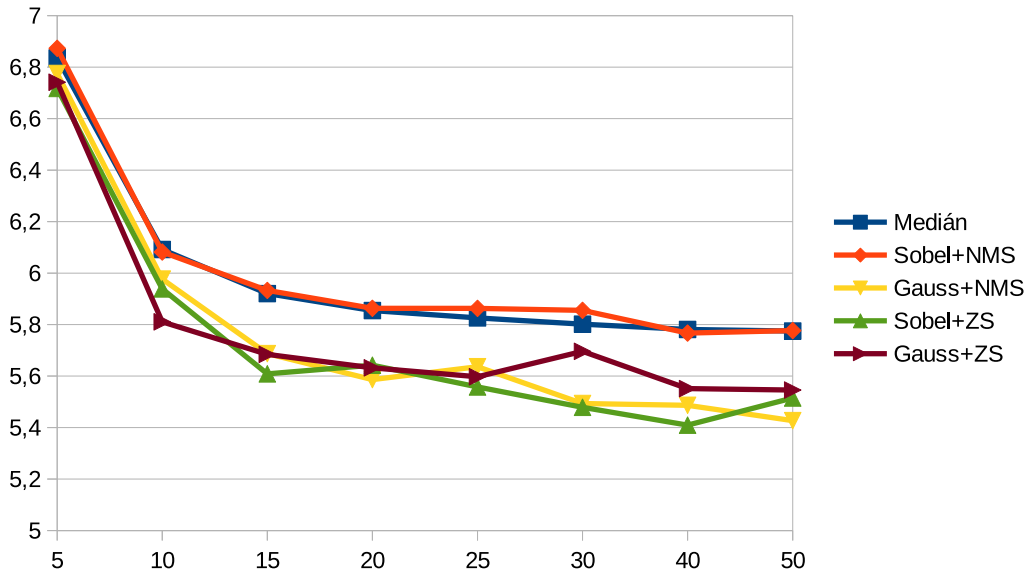
Z tohoto porovnání vyplývá, že přesnost a výsledné zlepšení je vysoce závislé na kvalitě

detekované hrany. Tato kvalita ale ne vždy musí znamenat stejnou kvalitu jakou myslíme u detekce hrany v barevných snímcích, jak ukazuje úspěšnost algoritmu Zhang-Suen, který některé hrany skrývá. Budoucí práce by se nejspíš zaměřila na zlepšení kvality detekce hrany, protože tato část je v tomto algoritmu klíčová. Existuje totiž mnoho dalších metod, jak detekovat hranu ve snímku, které nebyly implementovány.

11.2 Vliv počtu snímků

Počet snímků v zásobníku knihovny, které jsou použité k rekonstrukci má samozřejmě velký vliv na kvalitu výsledku, což také ukazuje graf (Obr. 20). Z grafu je patrné jak množství snímků ovlivňuje výslednou hodnotu. Také je vidět, že navržená metoda využívá výhody detekce hrany a kopíruje křivku mediánu. Ke konci osy je patrné, že náhodný generátor již vygeneroval dostatečné množství správných bodů napříč snímky a není už tolik výhodné používat více dat, protože již nepřináší další informace.

Samozřejmě zlepšení mezi 20 a 50 snímky je z grafu stále patrné, přesto největší zlepšení je právě mezi prvními 15 snímky. Tato hranice se samozřejmě posune se změnou rozptylu náhodného generátoru. Pokud bude náhodný generátor generovat více špatných bodů, znamená to, že bude třeba více snímků, aby převážily správné body. Při použití samozřejmě je třeba najít nejen vhodnou metodu, ale i vhodný počet snímků, protože při použití příliš velkého množství snímků může trvat naplnění zásobníku neúměrně dlouho, vzhledem k výslednému efektu.



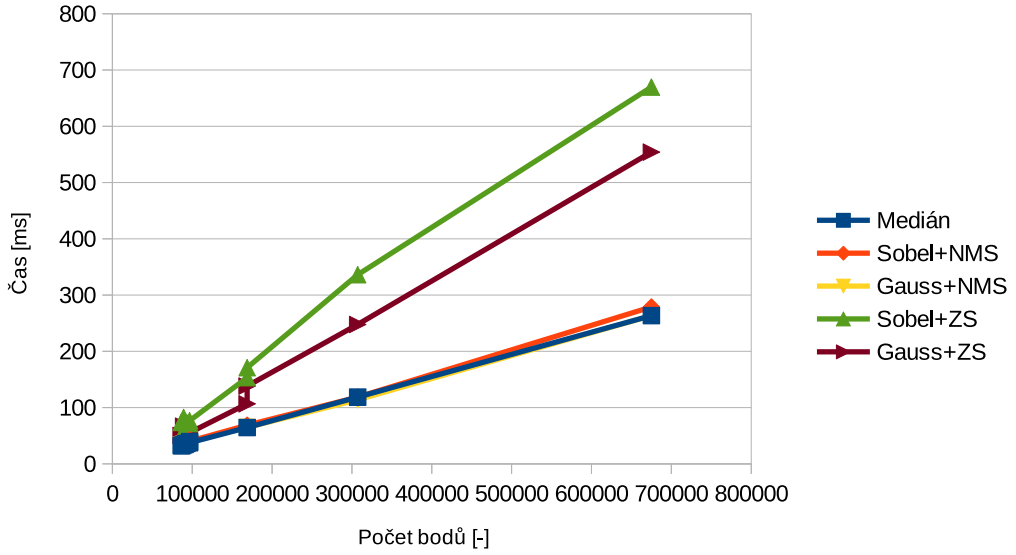
Obr. 20: Průběh RMSE pro zvyšující se počet snímků v zásobníku knihovny. Použitý snímek laundry.

11.3 Časová náročnost

Algoritmus je velmi rychlý. Postupy, které používají algoritmus Zhang-Suen pro ztenčení detekované hrany navíc mají výhodu v tom, že mohou běžet téměř celou dobu paralelně, jen je třeba synchronizovat data mezi jednotlivými fázemi algoritmů. Přesto nejspíš nebude potřeba použití metod pro paralelní výpočty, protože knihovna je velmi rychlá.

K měření času jsem použil svůj osobní počítač s procesorem Intel i7-7700HQ s pracovní frekvencí 2.80GHz. Pro měření času jsem použil standardní knihovnu *chrono* z C++.

Z grafu (Obr. 21) je vidět přibližně lineární růst, což může být způsobené i nedostatečnou variací v datech. Většina snímků má totiž nízké rozlišení, podobné tomu, co 3D kamery v dnešní době nabízí. Takto nízká rozlišení zvládne knihovna při 50 snímcích zpracovat do 200 ms. V grafu je vidět také, že algoritmus Zhang-Suen je o hodně pomalejší, než Non-maxima suppression. Přesto pro tyto velikosti algoritmus běží svižně a zvládl by při dobré správě bez problémů i 60 snímků za sekundu. Musela by se však lehce změnit správa paměti uvnitř knihovny.



Obr. 21: Porovnání časů v závislosti na počet bodů snímku.

Po teoretické stránce je algoritmus složen z několika fází, které mají různé časové složitosti.

První fáze je supersampling pomocí mediánu. Tato fáze pro každý bod výsledného snímku projde všechny snímky v zásobníku a seřadí body na příslušném místě. To se dá zapsat jako časová složitost $O(h \times w \times n^2 \times log(n))$, kde h je šířka a w je výška výsledného snímku a n je počet snímků v zásobníku.

Druhá fáze je nepovinná, ale protože jsem použil vyhlazení pomocí gaussovy funkce, která se dá provést pomocí konvoluce se separabilním jádrem, složitost je $O(h \times w \times \sqrt{3 \times sigma})$. Tato fáze je tedy méně náročnější, než první.

Dále je detekce hrany, která má kvůli použití dvojité konvoluce složitost $O(2 \times h \times w \times e^2)$, kde e znamená velikost jádra pro konvoluci.

Poté následují algoritmy pro ztenčení hrany. Non-maxima suppression algoritmus má i přes složité podmínky jednoduchou složitost $O(h \times w)$, protože má pouze jediný průchod. U algoritmu Zhang-Suen je teoretická složitost těžká na určení. Ze své povahy je tento algoritmus závislý na vstupních datech. S určitostí tedy můžu určit jen to, že nebude mít vyšší, než kvadratickou složitost vůči počtu vstupních bodů, což znamená $O((h \times w)^2)$. Toto by byla totiž složitost, když by algoritmus eliminoval pouze jeden bod za každý průchod.

Jako poslední je kontrola bodů okolo hran. Tento algoritmus je závislý na počtu bodů, které jsou označené jako hrana. Nejhorší složitost, ale tento algoritmus bude mít $O(h \times w)$, protože projde celý obraz jen jednou.

Když složím všechny tyto složitosti, získám, že algoritmus je shora omezený kvadratickou složitostí.

$$O(h{\times}w{\times}n^2{\times}log(n))+O(h{\times}w{\times}\sqrt{3 \times sigma})+O(2{\times}h{\times}w{\times}e^2)+O(h{\times}w)+O(h{\times}w) = O(h{\times}w) \tag{28}$$

$$O(h{\times}w{\times}n^2{\times}log(n))+O(h{\times}w{\times}\sqrt{3 \times sigma})+O(2{\times}h{\times}w{\times}e^2)+O((h{\times}w)^2)+O(h{\times}w) = O((h{\times}w)^2) \tag{29}$$

Z grafu (Obr. 21) je ale vidět, že reálná složitost bude blíže k lineární, protože konstanty, které jsou v těchto složitostech jsou nastavené na velmi malé hodnoty oproti velikosti h a w, a protože algoritmus Zhang-Suen může eliminovat mnoho bodů najednou. Ani počet snímků n nemusí být příliš velký, jak jsem ukázal v předchozí kapitole, ale samozřejmě pro opravdu velký počet snímků se tento člen projeví v (28).

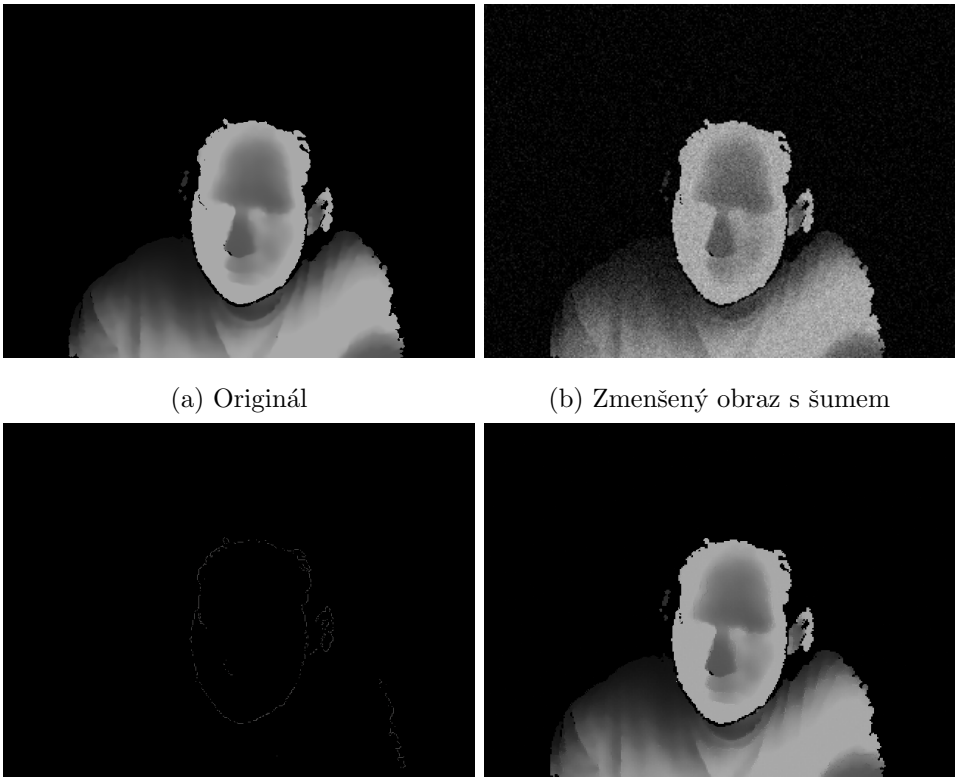
11.4 Vizuální analýza výsledků

V této kapitole otestuji vizuální výsledky, včetně zpracování série snímků ze zapůjčené kamery. Tyto výsledky ale nemohu ohodnotit exaktně, protože samozřejmě nemám přesnou reprezentaci, která se obvykle snímá přesným laserovým systémem a následně se ještě ručně čistí. Na těchto výsledcích proto budu hledat různé chyby, které najdu a pokusím se je vysvětlit. Většina problémů ale je způsobená především aliasingem a potom nesprávnými metodami rekonstrukce.

Snímek myface je snímek, který jsem pořídil, když se mi podařilo zprovoznit zapůjčenou kameru Creative. Moje aplikace může použít i barevné snímky, ale protože je nastavená na hloubku s rozlišením osmi bitů, aplikace barevnou fotku převede na černobílou. Na tomto obraze už neodpovídá barva hloubce, ale to nevadí pro potřeby analýzy. Toho je možné si všimnout podle porovnání barev nosu a ramen. Nos je šedý, podobně jako ramena, ale jejich vzdálenost od kamery musí být rozdílná, vzhledem k tomu, že tváře jsou podstatně světlejší.

Jak je vidět na snímcích (Obr. 22), přestože jsem přidal šum a obraz zmenšil jednoduchou metodou nejbližšího souseda, podařilo se zrekonstruovat obraz až na hrany, které se objevily v rozdílovém obraze. V rozdílovém obraze navíc nejsou vidět všechny hrany, ale pouze ty s největším rozdílem mezi hranou a černou, která značí, že senzor nebyl schopný detekovat vzdálenost. Tyto chyby právě vznikají kvůli aliasingu a jejich eliminování je záležitostí dobré metody pro detekci hran.

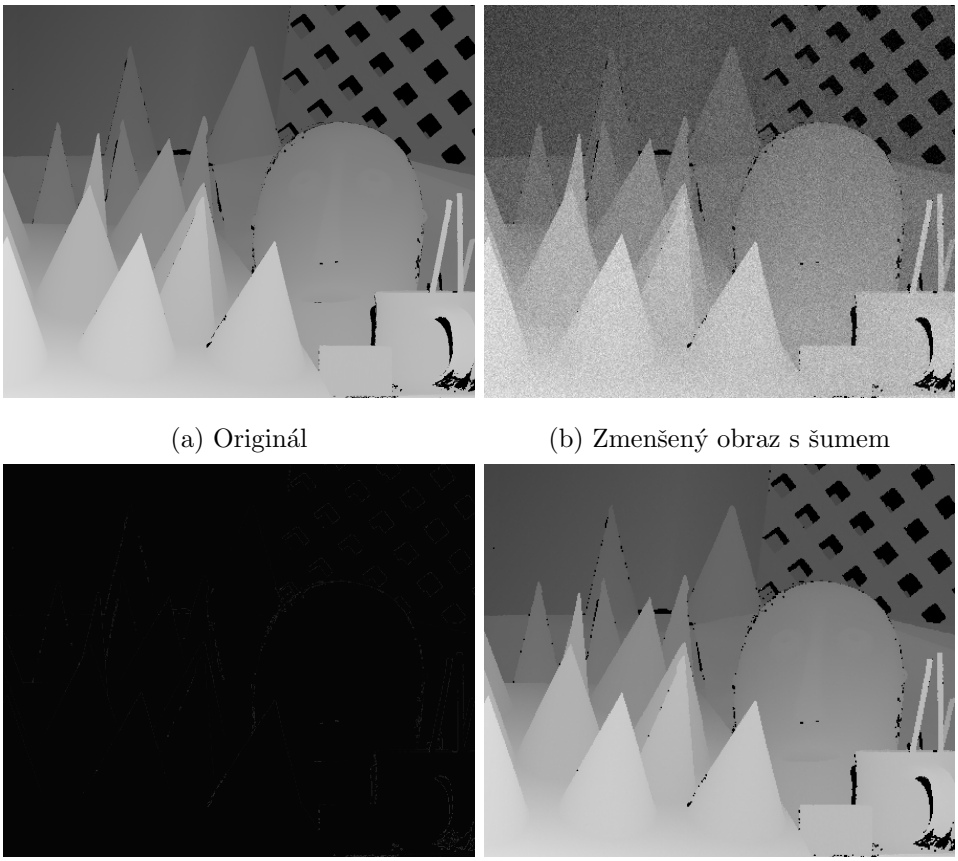
Na snímcích ze setu Middlebury (Obr. 23) je vidět podobný efekt. V tomto obraze ale už hloubka odpovídá barvě. Na výsledku je navíc vidět aliasing na tyčinkách vpravo. Na rozdílovém obraze jsou vidět chyby okolo hran, které jsou způsobené nejspíš chybou v detekci hrany.



(a) Originál (b) Zmenšený obraz s šumem

(c) Rozdílový obraz (d) Výstup

Obr. 22: Ukázka výstupu z předváděcí aplikace s obrazem myface. Originál ukazuje jak vypadá načtený obraz z původního souboru. Zmenšený obraz s šumem je jeden z obrazů, které jsou přímo nahrané do knihovny. Výstup je výstup z knihovny, která má načtené padesát různých zmenšených obrazů. Na rozdílovém obraze je vidět chyba okolo hrany, která není příliš velká.



(a) Originál (b) Zmenšený obraz s šumem

(c) Rozdílový obraz (d) Výstup

Obr. 23: Ukázka výstupu z předváděcí aplikace s obrazem cones. Originál ukazuje jak vypadá načtený obraz z původního souboru. Zmenšený obraz s šumem je jeden z obrazů, které jsou přímo nahrané do knihovny. Výstup je výstup z knihovny, která má načteno padesát různých zmenšených obrazů. Na rozdílovém obraze je vidět chyba okolo hrany, která není příliš velká.

Jako poslední test jsem dal dohromady testovací scénu z různých předmětů (Obr. 24) a po nastavení kamery jsem sejmul 20 snímků pomocí senzoru hloubky. Výsledky jsou velmi podobné, což je nejspíš způsobené dost jasnými hranami v obraze a také tím, že algoritmy fungují velmi podobným způsobem. Přesto se mi podařilo najít rozdíly a změny k lepšímu v některých místech obrazu. Nejlépe vypadá spojení Sobelova jádra pro detekci hran a Zhang-Suen algoritmus pro ztenčení hran. Tato metoda má výhodu v tom, že hrany jsou hladší, aliasing není tolik viditelný. Naopak detekce hran pomocí Gaussovy první derivace na těchto datech nefunguje příliš dobře, přesto obraz není horší, než jednoduchý filtr pomocí mediánového algoritmu.

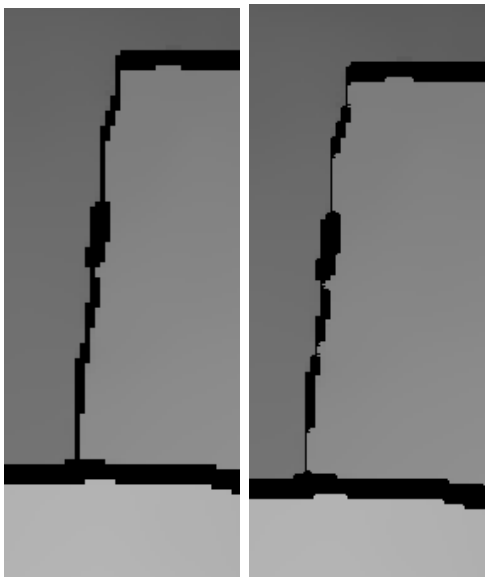
Obr. 24: Fotografie mnou vytvořené scény.

Původně jsem nečekal, že vůbec bude možné nasnímat figurku draka, kvůli černé barvě, ale nakonec je v hloubkové mapě vidět. Díky tomu se mi povedlo vytvořit sérii map s poměrně složitým objektem uvnitř. Podle očekávání ale ovladač v pozadí vytvořil černý stín, kvůli slabému odrazu infračerveného záření, které kamera používá k měření. Ostatní objekty se na snímcích dle očekávání projevily, přesto malá plocha lžičky v hrnku vytvořila spíše černou oblast, než očekávaný obdélník. Kamera také nedokázala sejmout stěnu v horních rozích, což je na snímcích patrné postupným zčernáním obrazu směrem k rohům.

Obr. 25: První snímek ze série dvaceti snímků heap.

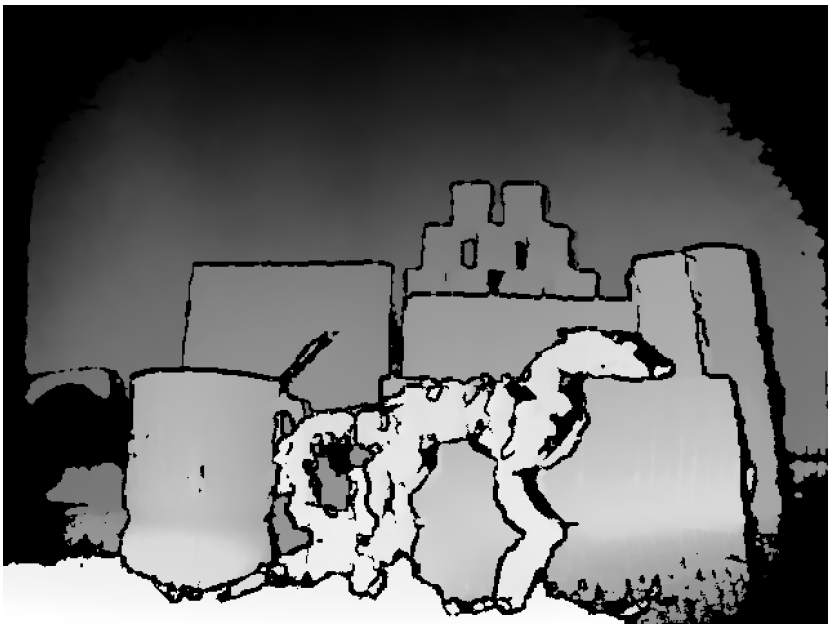
Kamera tvoří především velké prostory okolo hran, kde není vůbec detekovaná hloubka v obraze. Kamera také nemá očekávaný šum intenzity, což je nejspíš způsobené dalším zpracováním, které provádí aplikace pro snímání obrazu, zato tato kamera má prostorový šum. Ve spodní čtvrtině obrazu je také možné si všimnout, jak se paprsky odrážely od podlahy a je patrné, že část spodní čtvrtiny obrazu je zrcadlový odraz scény.

Šum je dobře potlačený už samotnou metodou mediánového supersamplingu. Přesto moje metoda zde našla drobná vylepšení, která jsou více patrná ve vyšších zvětšeních, než je, mnou běžně používané, čtyřnásobné zvětšení. například při zvětšení 9× (Obr. 26) je patrná tenčí černá hrana, která je ve výsledku spíše na škodu.



(a) Medián (b) Sobel + Zhang-Suen

Obr. 26: Výřez z obrázku zvětšeného 3× v každém rozměru. Je vidět jak výstup z knihovny má tenčí černou hranu, kterou mediánový supersampling rozšířil.



(a) Výstup pomocí metody mediánového supersamplingu

(b) Výstup pomocí kombinace Sobelovy detekce hrany a Zhang-Suen algoritmu pro ztenčení hrany

Obr. 27: Ukázka výstupu z knihovny pro sérii dvaceti obrázků heap, nasnímaných kamerou Creative BlasterX SENZ3D. Největší zlepšení je vidět na krku a těle figurky draka, kde se podařilo rekonstruovat větší množství bodů, které jsou na výstupu z mediánového supersamplingu jednoduše černé. Také má výstup mnohem menší aliasing, než samotný mediánový supersampling.

12 Závěr

Jak je z výsledků vidět, mnou navržený postup funguje a implementace je v průměru o 5% lepší, než jednoduchá metoda mediánu a zároveň není o moc pomalejší, jako existující implementace, které ale používají jiné postupy. Postup, který jsem navrhl má velkou výhodu v rychlosti, ale v kvalitě rekonstrukce zcela určitě ještě nedosáhl svého limitu. Existují složitější postupy na extrakci hrany a určitě by se dala vylepšit moje jednoduchá aproximace zpřesnění hran.

Mnou navržený algoritmus je momentálně asi nejvhodnější použít pro rychlé zpracování signálu, který je poměrně kvalitní, nasnímaný třeba v laboratoři, případně pro nekritické aplikace, kde není nutné znát přesnou hranu. Ideální bude moje knihovna na časově kritické aplikace, kde je nutná rychlost, protože zpracování je podobně rychlé jako zpracování pomocí algoritmu mediánu, takže zde se nabízí lepší kvalita obrazu za cenu téměř stejně rychlého zpracování.

12.1 Pokračování práce

Další pokračování práce by mělo směřovat k implementaci jiných detekcí hran. Například existující složité metody, které jsou založené na fitování křivek generují hrany s přesností vyšší, než je přesnost, kterou mi mohou nabídnout obrazové body, které jsou momentálně použité.

Další místo, kde je velká možnost ke zlepšení je vylepšit, nebo nahradit moji jednoduchou aproximaci kontroly pixelů okolo hran. Zde by bylo potřeba vymyslet rychlé hledání okolních bodů, možná zkusit náhodný sampling, nebo testovat všechny pixely okolo hrany místo testování pixelů pouze ve směru hrany.

Nakonec je zde možnost, která nebyla v této práci z časových důvodů zkoumána, a to je implementace algoritmu pro OpenMP a OpenCL, aby bylo možné výpočty urychlit pomocí výpočtů na více jádrech nebo GPGPU.

Pokračování dále by pak zahrnovalo nejspíše nějaké strojové učení, protože běžným filtrováním by se dalo těžko rozlišit mezi prostorem, který je nesejmutý a hranou, které jsou v obrazech z kamery patrné. Ve všech obrazech jsou totiž některé hrany ohraničené prostorem, který kamera nebyla schopná analyzovat a vrátit v tomto prostoru hloubku v obraze. Tyto prostory by nejspíše bylo výhodné nějakou chytrou metodou detekovat a eliminovat z obrazu

a tím získat úplnou mapu. Přesto, jak je patrné z mé série nasnímaných obrazů je nutná pečlivá analýza, protože se v obraze mohou vyskytovat různé objekty, které by tato metoda mohla skrýt, jako například drát vedoucí od ovladače.

13 Návod - Aplikace pro prezentaci

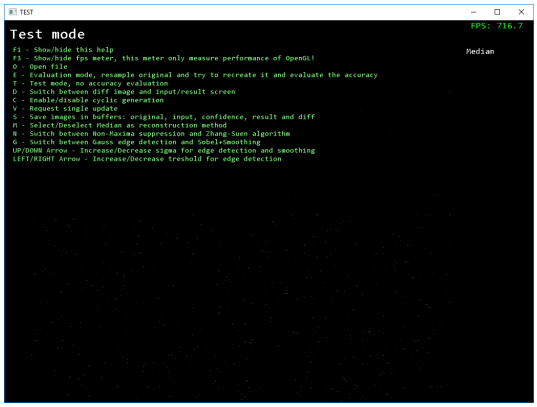
Aplikace pro prezentaci má svoji konzoli, která slouží spíše k výpisům, co aplikace zrovna dělá a samotné okno aplikace. Okno aplikace je hlavní pro ovládání, ale konzoli není možné zavřít, protože to způsobí ukončení aplikace.

Aplikace se ovládá pomocí několika vybraných kláves na klávesnici. V tabulce 4 jsou vypsané funkční klávesy a jejich funkce, kterou v aplikaci mají.

Na obrázku (Obr. 28) je vidět, jak aplikace vypadá v různých stavech. Na prvním obrázku je vidět vypsané ovládání uvnitř programu. To je viditelné automaticky při spuštění. S velikostí okna je možné manipulovat běžnými způsoby. Počet snímků za sekundu (FPS) v okně neodpovídá reálné době výpočtu, ale váže se k vykreslování aplikace knihovnou OpenGL. Protože v okně aplikace není mnoho trojúhelníků a není zapnutá synchronizace, počty snímků jsou vysoké.

Esc	Způsobí zavření aplikace.
F1	Zobrazí a schová popisek s klávesami a jejich funkcí uvnitř programu.
F3	Zobrazí a schová měřič snímků za sekundu vpravo nahoře. Toto číslo odpovídá pouze rychlosti vykreslování. Knihovna běží v jiném vlákně.
O	Otevře dialog k otevření nového obrazu z disku počítače pro vnitřní generátor, který potom generuje vstupní obrazy pro knihovnu.
E	Zapne evaluační mód, kdy aplikace resampluje originální obraz na nižší rozlišení a následně porovná výstup s originálem a vytvoří rozdílový obraz a spočítá statistiku.
T	Zapne testovací mód, kdy aplikace použije rozlišení originálního obrazu a pomocí přidání šumu vygeneruje vstupní obraz, následně jen zobrazí výstup. V tomto módu se nepočítá statistika, protože není výstup s čím porovnat.
D	Přepíná zobrazení kombinovaného výstupu a diferenciálního obrazu v evaluačním módu.
C	Zapíná a vypíná cyklické spouštění generování dalších snímků a počítání výsledného obrazu.
V	Vynucení vygenerování dalšího vstupního snímku a spočítání výsledného obrazu.
S	Uloží všechny obrazy v aplikaci. V testovacím módu se jedná o originální obraz, momentální vstupní snímek, snímek s jistotou a výstupní snímek z knihovny. V evaluačním módu se navíc uloží i rozdílový obraz. Tyto obrazy se uloží do cesty, ze které byla aplikace spuštěná a jejich jména jsou definovaná v kódu, takže se nedají měnit.
M	Zapne, nebo vypne omezení na rekonstrukci pouze pomocí mediánu.
N	Pokud je vypnuté omezení na rekonstrukci pomocí mediánu, tato klávesa přepíná mezi algoritmy na ztenčení hrany.
G	Pokud je vypnuté omezení na rekonstrukci pomocí mediánu, tato klávesa přepíná mezi detekcí hrany pomocí první derivace gaussovy funkce, nebo detekci hrany pomocí Sobelova jádra s vyhlazením.
Šipky nahoru/dolů	Mění sigma pro vyhlazení před použitím Sobelova jádra pro detekci hrany, nebo přímo sigma první derivace gaussovy funkce pro detekci hrany.
Šipky vlevo/vpravo	Mění sigma nastavení prahu, který je aplikován po detekci hrany.

Tab. 4: Klávesy a jejich funkce



(a) Po spuštění

(b) Testovací mód s otevřeným snímkem laundry

(c) Evaluační mód s rozdílovým obrazem.

(d) Evaluační mód bez rozdílového obrazu

Obr. 28: Hlavní okno programu v různých stavech.

14 Rozhraní knihovny

Rozhraní knihovny je napsané pro jazyk C, aby bylo jednodušší knihovnu použít i bez toho, aby jazyk podporoval třídy. Knihovna uvnitř třídy sice používá, ale z vnějšku má pouze instanci, uchovanou v globální paměti knihovny. K této třídě se nedá běžným způsobem dostat z aplikace, která používá knihovnu.

Rozhraní je definované v souboru dmssMain.h. Jako první jsou definice výčtových typů, které definují některé vstupy funkcí.

```
typedef enum e_DMSS_ComputingType {CT_DMSS_SIMPLE = 0,
                                   CT_DMSS_OPENMP = 1,
                                   CT_DMSS_OPENCL = 2} DMSS_ComputingType;
typedef enum e_DMSS_Quality { Q_NA = 0,
                                   Q_8b = 1,
                                   Q_16b = 2,
                                   Q_32b = 4 } DMSS_Quality;
typedef enum e_DMSS_EdgeThinningAlgorithm {
                                   ETA_NonMaxima = 0,
                                   ETA_ZhangSuen = 1
                                   } DMSS_EdgeThinningAlgorithm;
```

ComputingType je momentálně nevyužitý. Plánované využití bylo pro inicializaci pamětí a třídy, která používá k výpočtu jiné metody. V této implementaci funguje pouze jednoduchá verze výpočtu `CT_DMSS_SIMPLE`. Třídy pro použití knihoven OpenMP (výpočet s více vlákny) nebo OpenCL (výpočet s pomocí GPGPU) nejsou definovány.

Quality je použitý pro inicializaci paměti a pro definování kvality vstupu a výstupu. Nejčastěji používané je rozlišení 8 bitů, ale knihovna je připravená i na vyšší rozlišení.

EdgeThinningAlgorithm je použitý pro nastavení algoritmu, který bude použitý po algoritmu detekce hrany, pro její ztenčení. Oba algoritmy byly popsány v kapitole 9.

Dále v souboru následují preprocesorové definice, rozdělené pro operační systém Windows a ostatní a také definice konvence volání `CALL_CONV`. Pro operační systém Windows je třeba definovat i definici `DMSSLIB_API`, která zabaluje pomocné definice knihovny právě pro operační

systém Windows. Pro ostatní operační systémy je tato definice prázdná.

Po těchto nutných definicích již následují definice jednotlivých funkcí.

Následující definice funkcí jsou pro ovládání samotné knihovny. Funkce init provede inicializaci s příslušným typem výpočtů. Momentálně jediný správný typ je `CT_DMSS_SIMPLE`. Pokud inicializace proběhne v pořádku, funkce vrátí true na znamení úspěchu. Funkce uninit uvolní veškerou paměť, kterou knihovna zabírá. Funkce switchComputingType je momentálně nepoužitá, protože ostatní typy nejsou použité, ale její předpokládané použití je, že provede funkci uninit a následně funkci init s příslušným předaným typem. Pokročilá verze této funkce by mohla umět i převod paměti knihovny, takže by se neztratilo nastavení, ale toto je složitější, pokud by se toto provádělo i pro výpočty na GPGPU.

```
bool dmss_init(const DMSS_ComputingType t);
void dmss_uninit();
bool dmss_switchComputingType(const DMSS_ComputingType t);
```

Další funkce jsou pro vyčtení a nastavení vnitřní paměti a formátu přijímaných a vytvořených dat. Funkce getBufferImageMax vrátí maximální počet uložených snímků a funkce getBufferImageCount vrátí momentální počet uložených snímků. Počet momentálně uložených snímků nikdy nepřekročí maximální počet, protože při přidání snímku nad maximum se jednoduše přepíše poslední snímek v paměti.

```
unsigned int dmss_getBufferImageMax();
unsigned int dmss_getBufferImageCount();
```

Funkce na nastavení kvality udávají velikost vstupního a výstupního typu proměnné a také rozlišení v hloubce vstupního a výstupního obrazu. Pro tyto funkce se používá hodnota z výčtového typu, nebo je možné použít velikost v bajtech. Povolené hodnoty jsou ale pouze 1, 2 a 4 bajty, které korespondují s 8, 16 a 32 bity rozlišení hloubky.

```
void dmss_setInputQuality(const DMSS_Quality q);
void dmss_setOutputQuality(const DMSS_Quality q);
```

Následující funkce dovolují vyčíst nastavení vstupního zásobníku a výstupní paměti. Funkce jsou definované tak, že parametry i funkce jsou jasné z názvů. Pokud knihovna není inicializovaná, tyto funkce vrací nuly pro všechny argumenty i vrácené proměnné.

```
void dmss_getInputImageSize(unsigned int & height,
                            unsigned int & width);
void dmss_getOutputImageSize(unsigned int & height,
                             unsigned int & width);
DMSS_Quality dmss_getInputQuality();
DMSS_Quality dmss_getOutputQuality();
```

Tyto funkce nastavují velikosti pamětí uvnitř knihovny. Tyto funkce pouze nastaví proměnné velikosti a zároveň uvnitř knihovny nastaví proměnnou, která řídí realokaci paměti. Dokud se nezavolá první přidání snímku, nebo rekonstrukce obrazu, příslušné paměti nejsou inicializovány. Funkce setBufferSize nastaví velikost vstupních snímků a jejich maximální počet v zásobníku knihovny. Funkce setOuputSize nastaví výstupní paměť na požadovanou velikost. Alternativně je možné použít setOutputScale, která vynásobí velikost vstupních snímků argumentem a nastaví tuto velikost jako výstupní. Tato funkce předpokládá, že vstupní velikost je již nastavená.

```
void dmss_setBufferSize(unsigned int height, unsigned int width,
                        unsigned int count);
void dmss_setOutputSize(unsigned int height, unsigned int width);
void dmss_setOutputScale(float scale);
```

Funkce na přidání snímku mají nedefinovaný typ void, aby bylo možné použít stejnou funkci s různými rozlišeními hloubky vstupních obrazů. Vstupní pole confidence označuje jak jsou jednotlivé prvky správné. Toto pole je nepovinné a pokud je ukazatel nastavený na nulu, toto pole není použité. Jinak toto pole využívá plný rozsah, kde 255 znamená plná jistota, že je buňka správně a 0 znamená, že buňka je zcela jistě špatně. Velikost tohoto pole je stejně jako u vstupního image dané šířkou a výškou obrazu.

Druhá funkce navíc uvažuje posunutý obraz, který je z kamery, u které známe posunutí

snímků vůči sobě. Posunutí je absolutní vůči nějakému jednomu snímku ve velikosti obrazového bodu.

```
void dmss_addImage(void * image, unsigned char * confidence);
void dmss_addDisplacedImage(void * image,
                            unsigned char * confidence,
                            float displacementX,
                            float displacementY);
```

Funkce reconstructImage spouští samotný algoritmus knihovny. Tato funkce má několik parametrů, které řídí rekonstrukci. Pokud je argument median nastavený na true, všechny ostatní argumenty se ignorují a knihovna vrátí pouze výstup z algoritmu pro počítání supersamplingu pomocí mediánu. První argument, smoothingSigma, je přímo rozptyl vyhlazování, které se provede před detekcí hrany, aby se odstranil aliasing, který může být na výstupu mediánového algoritmu. Doporučená hodnota se pohybuje okolo 1. Argument edgeSigma je rozptyl pro generování jádra pro detekci hrany pomocí první derivace gaussovy funkce. Pokud je tento argument nenulový, je lepší ponechat první argument nulový a naopak. Obvyklá hodnota pro tento argument je 2. Pokud je tento argument nula, použije se jádro Sobelova detektoru hrany. Argument edgeTreshold působí jako práh pro detekci hrany. Slabší detekce jsou rovnou potlačené. Obvyklá hodnota tohoto argumentu je 20. Poslední argument slouží ke zvolení algoritmu na ztenčování hrany po detekci.

Funkce clearBuffer, věrně ke svému názvu, smaže obsah zásobníku knihovny a nastaví počet uložených snímků znovu na 0, jako při inicializaci knihovny. Funkce getOutput vrátí ukazatel na paměť knihovny. Tento ukazatel se nesmí mazat, správu tohoto ukazatele si řídí knihovna sama.

```
void dmss_reconstructImage(float smoothingSigma,
                           float edgeSigma,
                           float edgeTreshold,
                           bool median,
                           DMSS_EdgeThinningAlgorithm alg);
```

```
void dmss_clearBuffer();
void * dmss_getOutput();
```

15 Reference

[1] James Ring, "The Laser in Astronomy." pp. 672–73, New Scientist June 20, 1963

[2] The Lunar and Planetary Institute - Apollo 15 mission https://www.lpi.usra.edu/lunar/missions/apollo/apollo_15/experiments/lrr/

[3] Stránky o umístění fotoreflektoru, jehož umístění bylo součástí mise Apollo 15 https://www.nasa.gov/mission_pages/LRO/multimedia/lroimages/lroc-20100413-apollo15-LRRR.html

[4] David Monniaux, Lidar systém - fotografie, model: Leica HDS-3000, https://commons.wikimedia.org/wiki/File:Lidar_P1270901.jpg

[5] Shannon, C.E. (1949) Communication in the Presence of Noise. Proceedings of the IRE, 37, 10-21. http://dx.doi.org/10.1109/JRPROC.1949.232969

[6] Nasa Shuttle Radar Topography Mission (2000) https://www2.jpl.nasa.gov/srtm/ https://dds.cr.usgs.gov/srtm/

[7] Intel RealSense Depth Camera D415 https://ark.intel.com/products/128256/Intel-RealSense-Depth-Camera-D415

[8] Infineon Real3D product details https://www.infineon.com/cms/en/product/sensor/3d-image-sensor-real3/

[9] Pawlikowska, A. M., Halimi, A., Lamb, R. A., & Buller, G. S.: Single-photon three-dimensional imaging at up to 10 kilometers range. Optics Express, 25(10), 11919-11931. DOI: 10.1364/OE.25.011919 https://researchportal.hw.ac.uk/en/publications/single-photon-three-dimensional-imaging-at-up-to-10-kilometers-ra

[10] Miles Hansard, Seungkyu Lee, Ouk Choi, Radu Horaud. Time of Flight Cameras: Principles, Methods, and Applications. Springer, pp.95, 2012, SpringerBriefs in Computer Science, ISBN 978-1-4471-4658-2 https://hal.inria.fr/hal-00725654/PDF/TOF.pdf

[11] Jason Geng, "Structured-light 3D surface imaging: a tutorial," Adv. Opt. Photon. 3, 128-160 (2011) https://www.osapublishing.org/aop/abstract.cfm?uri=aop-3-2-128

[12] Gonzalez R. C., Woods R. E.: Digital Image Processing (3rd Edition), Prentice Hall, 2008

[13] Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. Physically Based Rendering: From Theory to Implementation (3rd ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. https://www.pbrt.org/

[14] Paris S., Kornprobst P., Tumblin J., Durand F.: Bilateral Filtering: Theory and Applications, Now Publishers, 2009

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12), F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.), Vol. 1. Curran Associates Inc., USA, 1097-1105. https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[16] Sebastian Schuon, Christian Theobalt, James Davis, Sebastian Thrun; LidarBoost: Depth superresolution for ToF 3D shape scanning. CVPR 2009 https://ieeexplore.ieee.org/document/5206804/

[17] Jun Xie, Rogerio Schmidt Feris, and Ming-Ting Sun; Edge-Guided Single Depth Image Super Resolution. IEEE Transactions on Image Processing vol. 25, 428-438, January 2016 http://www.clairexie.org/depth_superresolution/index.html http://www.clairexie.org/resources/TIP16.pdf https://github.com/ClaireXie/edgeGuidedSDSP

[18] Jing Li, Zhichao Lu, Gang Zeng, Rui Gan, Hongbin Zha; Similarity-Aware Patchwork Assembly for Depth Image Super-resolution. CVPR 2014 https://ieeexplore.ieee.org/document/6909827/ https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Li_Similarity-Aware_Patchwork_Assembly_2014_CVPR_paper.pdf

[19] Jiajun Lu, David Forsyth; Sparse Depth Super Resolution. CVPR 2015 https://ieeexplore.ieee.org/document/7298837/ https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Lu_Sparse_Depth_Super_2015_CVPR_paper.pdf

[20] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International Journal of Computer Vision, 47(1/2/3):7-42, April-June 2002.

D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), volume 1, pages 195-202, Madison, WI, June 2003.

D. Scharstein and C. Pal. Learning conditional random fields for stereo. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), Minneapolis, MN, June 2007.

H. Hirschmüller and D. Scharstein. Evaluation of cost functions for stereo matching. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), Minneapolis, MN, June 2007.

`http://vision.middlebury.edu/stereo/data/`

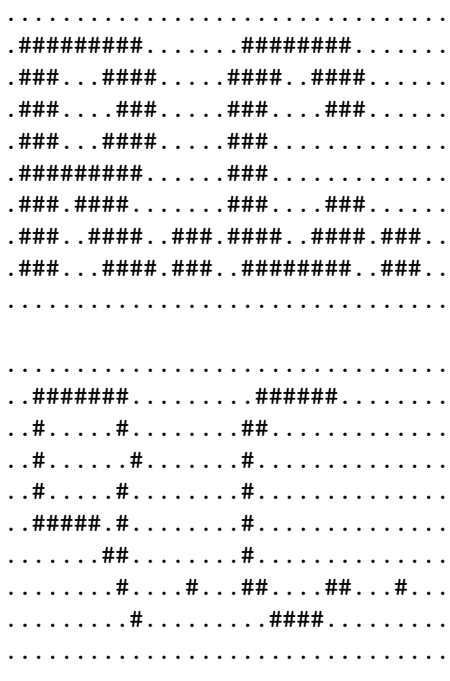
[21] Valeria Garro, Carlo Dal Mutto, Pietro Zanuttigh, Guido M. Cortelazzo; Edge-preserving interpolation of depth data exploiting color information datasets. `http://lttm.dei.unipd.it/downloads/superres/`

[22] John Canny; A Computational Approach to Edge Detection. IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986 `https://ieeexplore.ieee.org/document/4767851`

[23] T. Y. Zhang, C. Y. Suen, A fast parallel algorithm for thinning digital patterns. Communications of the ACM, 27(3), 236–239, 1984 `https://doi.org/10.1145/357994.358023`

[24] J. S. Chen, A. Huertas, and G. Medioni. Fast convolution with Laplacian-of-Gaussian masks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 9(4):584–590, 1987

16 Přílohy



Obr. 29: Ukázka Zhang-Suen algoritmu na obrázku reprezentovaném znaky. Nuly jsou reprezentované tečkami, křížky reprezentují jedničky. Na příkladu je vidět, jak algoritmus agresivně zničí celou jednu hranu písmena R a částečně poničí vrchní oblouk C. Převzato z `https://rosettacode.org/wiki/Zhang-Suen_thinning_algorithm`, kde se nachází vysvětlení a implementace v různých jazycích.