Transformace objektů z 2D do 3D pomocí metod A.I.

Bc. Tomáš Polívka

Diplomová práce
2024

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Tomáš Polívka**
Osobní číslo:	**A22277**
Studijní program:	**N0613A140022 Informační technologie**
Specializace:	**Softwarové inženýrství**
Forma studia:	**Kombinovaná**
Téma práce:	**Transformace objektů z 2D do 3D pomocí metod A.I.**
Téma práce anglicky:	**Transformation of Objects from 2D to 3D Using A.I. Methods**

Zásady pro vypracování

1. Vypracujte literární rešerši se zaměřením na současné technologické možnosti transformace objektů z 2D do 3D.
2. Proveďte podrobnější popis možností a funkcionality současných technologií zpracování a převod fotografie do modelu ve formátu 3D mesh.
3. Zvolte vhodný dostupný dataset, nebo z více zdrojů vytvořte svůj vlastní dataset.
4. Zvolte vhodný jeden či více AI modelů, jejich nastavení a programovací prostředí.
5. Implementujte AI model či modely pro transformaci objektů.
6. Popište funkcionalitu navrženého přístupu, možnosti dalšího rozvoje či nastavení a v případě více modelů také proveďte porovnání.

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. GÉRON, Aurélien. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems*. Third edition. Beijing: O'Reilly, [2023]. ISBN 978-1-098-12597-4.
2. CHOLLET, François. *Deep learning v jazyku Python: Knihovny Keras, TensorFlow*. Grada, 2019. ISBN 978-80-247-3100-1.
3. SOLEM, Jan Erik. *Programming computer vision with Python*. Beijing: O'Reilly, 2012. ISBN 9781449316549.
4. ROSEBROCK, Adrian. *Deep learning for computer vision with Python*. [Místo vydání není známé]: PyImageSearch, 2017. ISBN 9781986538138.
5. HARTLEY, Richard a ZISSERMAN, Andrew. *Multiple view geometry in computer vision*. Second edition. Cambridge: Cambridge University Press, 2003. ISBN 0521540518.
6. JIN, Yiwei; JIANG, Diqiong; CAI, Ming. 3d reconstruction using deep learning: a survey. *Communications in Information and Systems*, 2020, 20.4: 389-413.
7. YUNIARTI, Anny; SUCIATI, Nanik. A review of deep learning techniques for 3D reconstruction of 2D images. In: *2019 12th International Conference on Information & Communication Technology and System (ICTS)*. IEEE, 2019. p. 327-331.
8. POUYANFAR, Samira, et al. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 2018, 51.5: 1-36.

Vedoucí diplomové práce: **Ing. Peter Janků, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **5. listopadu 2023**
Termín odevzdání diplomové práce: **13. května 2024**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 5. ledna 2024

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomové práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 10.5.2024

Tomáš Polívka, v. r
podpis studenta

ABSTRAKT

Tato práce se zabývá technologickými možnostmi převedení 2D informace do 3D podoby, se zaměřením na využití umělé inteligence. Práce nejprve popisuje a porovnává aktuální možnosti této transformace a dále se zabývá praktickou implementací programu, který bude tuto transformaci provádět s využitím 3D rekonstrukce pomocí metody stereo vize a CNN (convolutional neural network).

Klíčová slova: 3D rekonstrukce, stereo vize, umělá inteligence, konvoluční neuronová síť, odhad hloubky, bodový oblak, trojrozměrná síť

ABSTRACT

This diploma thesis deals with the technological possibilities of converting 2D information into 3D form, focusing on the use of artificial intelligence. The paper first describes and compares the current options for this transformation and then addresses the practical implementation of a program that will perform this transformation using 3D reconstruction with stereo vision and CNN (convolutional neural network).

Keywords: 3D reconstruction, stereo vision, artificial intelligence, convolutional neural network, depth estimation, point cloud, 3D mesh

Poděkování

Děkuji svému vedoucímu této diplomové práce panu Ing. Peterovi Janků, Ph.D. za jeho čas, přínosné rady a zpětnou vazbu.

OBSAH

ÚVOD

V této diplomové práci se zaměřuji na možnosti transformace informace z 2D do 3D. Téma mapování 2D do 3D je v dnešní době velmi relevantní, jelikož techniky spojené s tímto procesem mají široké uplatnění v moderních a trendových technologiích. Nejjednodušším příkladem a také příkladem, na který se v této části soustředím, je převod obrázku na 3D model, v tomto případě ve formátu 3D mesh.

Mezi příklady využití těchto technologií patří příprava dat pro 3D tisk, asistence v řízení vozidel, aplikace rozšířené reality a nebo třeba využití v medicíně, kde jsou detailní 3D modely nezbytné.

Tato práce je rozdělena do dvou částí. První část se věnuje literární rešerši, která se zaměřuje na současné technologické možnosti transformace objektů z 2D do 3D. Tento průzkum zahrnuje přehled existujících metod a přístupů z různých technologických oblastí. Mezi ně patří počítačové vidění, strojového učení a umělé inteligence ale i laserové skenování. Dále tato práce podrobněji popisuje technologie zpracování a převodu fotografií do modelu ve formátu 3D mesh, včetně algoritmů pro hloubkovou mapu, strukturální reprezentaci objektů a rekonstrukci povrchu.

Druhá část je implementačního rázu. Zde popisuji implementaci praktického příkladu této transformace pomocí stereovize a vybraných metod umělé inteligence. Tento příklad je realizován v jazyce Python s využitím knihoven jako je OpenCV a PyTorch. Výstupem bude fungující program, který bude mít na vstupu dva obrázky a na výstupu 3D model ve formátu 3D mesh. Tento projekt je v diplomové práci popsán, ohodnocen a jsou navrženy další kroky, které by mohly zvýšit kvalitu a efektivitu tohoto programu.

Hlavním cílem této práce je přispět k rozšíření poznatků o technikách mapování 2D do 3D a poskytnout ucelený pohled na současné metody a možnosti v této oblasti. Kromě toho se práce snaží o praktickou aplikaci těchto poznatků prostřednictvím implementace konkrétního příkladu, v němž bude AI hrát klíčovou roli. Tím by mohlo dojít k dalšímu pokroku v oblasti vývoje aplikací a systémů využívajících 2D a 3D technologie. Důkladná analýza a zhodnocení implementovaného příkladu by měla nabídnout přínosné poznatky pro další výzkum a vývoj v oblasti zpracování obrazu, strojového učení a umělé inteligence.

I. TEORETICKÁ ČÁST

1 POPIS ZÁKLADNÍCH POJMŮ

V této úvodní části popisuji základní pojmy spojené s tématem celé práce. Níže zmíněné pojmy jsou dále použité v několika dalších kapitolách.

2D vs 3D

2D (dvojrozměrný) a 3D (trojrozměrný) se vztahují k počtu dimenzí, ve kterých objekt nebo informace existují. 2D informace se skládají z šířky a výšky a jsou obvykle reprezentovány v obrazech nebo grafice, jako jsou fotografie a náčrty. Naproti tomu 3D informace přidávají hloubku k šířce a výšce, což umožňuje reprezentaci ve třech prostorových dimenzích. Tento přechod z 2D do 3D umožňuje zachytit reálné rozměry objektů, což je zásadní pro aplikace, jako je 3D tisk, architektonická vizualizace a virtuální realita. [1]

Mapa disparity

Disparitní mapa je klíčovou mezireprezentací ve stereovizi, která měří rozdíly mezi odpovídajícími pixely ve dvou stereosnímcích. Každý pixel na disparitní mapě představuje vzdálenost (disparitu) mezi odpovídajícími body na dvou obrazech pořízených z mírně odlišných úhlů. Tato mapa hraje hlavní roli v odhadu hloubky tím, že ukazuje, jak moc se každý bod ve scéně posunul mezi dvěma snímky, což umožňuje rekonstrukci 3D modelů. [2]

Bodové mračno (Point Cloud)

Bodové mračno je sada datových bodů reprezentujících trojrozměrný prostor, obvykle shromážděných z 3D skenovacích technologií nebo rekonstruovaných ze stereo obrazů. Každý bod v mračnu má 3D souřadnice (X, Y, Z) a může také zahrnovat další informace, jako je barva nebo intenzita. Bodová mračna jsou základní pro tvorbu 3D modelů a lze je zpracovat do jiných formátů, jako jsou 3D mesh modely. [2]

CNN (Konvoluční neuronová síť)

Konvoluční neuronová síť je druh modelu hlubokého učení optimalizovaný pro zpracování vizuálních informací, což ji činí vysoce efektivní v úlohách rozpoznávání obrazu a videa. CNN se skládá z vrstev, které aplikují konvoluční operace k extrakci rysů z obrazů, snižují jejich rozměrovost pomocí pooling vrstev a nakonec zpracovávají výsledky přes plně propojené vrstvy pro generování požadovaného výstupu, jako jsou hloubkové mapy nebo klasifikace. CNN jsou zásadní pro automatizaci generování disparitních map ze stereo obrazů, což významně zlepšuje efektivitu 3D rekonstrukce. [3]

Dataset

Dataset v kontextu umělé inteligence a hlubokého učení je soubor dat, který slouží k trénování a testování algoritmů strojového učení. Tento soubor může obsahovat různé typy dat, jako jsou textové dokumenty, obrázky, zvukové záznamy nebo časové řady. Jeho hlavním účelem je poskytnout dostatečné množství informací algoritmům, aby se mohly naučit identifikovat vzory a vytvářet předpovědi na základě nových dat. Dataset může být rozdělen na trénovací, validační a testovací části, aby bylo možné vyhodnotit výkon modelu na nezávislých datech. Důkladná příprava a správa datasetu jsou klíčové pro úspěšné trénování a nasazení modelů umělé inteligence. [3]

Depth Estimation

Odhad hloubky je proces určování vzdálenosti mezi objekty ve scéně a pozorovatelem nebo kamerou na základě vizuálních klíčů. Tento proces je zásadní pro převod 2D obrazů na 3D modely, zejména u metod, jako je stereovize, kde je hloubka odhadována z disparit mezi dvojicí obrázků. [2]

Feature Matching

Feature matching je proces identifikace odpovídajících bodů mezi dvěma obrazy nebo sadami dat. Ve stereovizi hraje klíčovou roli při generování disparitních map, což umožňuje odhad hloubky srovnáním relativních pozic spárovaných bodů. Různé algoritmy a modely, jako je SIFT, SURF nebo CNN, mohou být použity pro tento účel. [2]

2 VÝZNAM TRANSFORMACE 2D OBJEKTŮ DO 3D

Techniky transformace 2D objektů do 3D mají velkou hodnotu v různých odvětvích. V této sekci se budu věnovat příkladům využití transformace informace z 2D do 3D.

2.1 Aplikace napříč odvětvími

Medicína a zdravotnictví

3D rekonstrukce hraje klíčovou roli v lékařském zobrazování, kde pomáhá při diagnostice, plánování léčby a chirurgických zákrocích. Převedením tradičních 2D snímků na podrobné 3D modely získávají lékaři komplexní pohled na vnitřní struktury, což usnadňuje přesné zákroky a snižuje rizika. [4]

Architektura a stavebnictví

Architekti a inženýři využívají 3D rekonstrukce k vytváření poutavých vizualizací budov a infrastrukturních projektů. Tato technologie umožňuje zúčastněným stranám detailně prozkoumat návrhy ještě před zahájením výstavby, což zefektivňuje proces plánování a minimalizuje chyby.

Archeologie a kulturní dědictví

Archeologové tuto techniku využívají k digitálnímu uchování a analýze historických artefaktů, památek a míst, což nabízí neocenitelné poznatky o dávných civilizacích a zároveň zachovává kulturní dědictví pro budoucí generace.

Zábava a hry

Zábavní průmysl využívá 3D rekonstrukce k poskytování poutavých vizuálních zážitků ve filmech, videohrách a simulacích virtuální reality. Vytvářením realistických prostředí a postav tvůrci vtahují diváky do bohatých příběhových světů a posouvají hranice představivosti.

Geoprostorové mapování a územní plánování

Urbanisté využívají 3D rekonstrukce k vytváření přesných map a modelů měst, což usnadňuje informované rozhodování o rozvoji infrastruktury, dopravních sítí a ochraně životního prostředí.

Marketing

3D rekonstrukce z 2D modelu je používaná i v on-line marketingu, možnost pohledu na 3D model produktu v e-shopech zvyšuje zájem zákazníků.

Vzdělávání a výzkum

3D rekonstrukce slouží jako vzdělávací nástroj, který umožňuje studentům a výzkumníkům zkoumat složité koncepty v oborech, jako je biologie, fyzika a strojírenství. Vizualizací abstraktních teorií a jevů v hmatatelné podobě získávají žáci hlubší porozumění probírané látce, což podporuje zvědavost a schopnost kritického myšlení.

Automobily a robotika

Pokroky v oblasti 3D rekonstrukce mění automobilový průmysl a robotiku. Autonomní vozidla spoléhají na 3D rekonstrukci při navigaci a detekci překážek, což zvyšuje bezpečnost a efektivitu na silnicích. Stejně tak robotika využívá 3D rekonstrukci pro prostorové povědomí a manipulaci s objekty, což robotům umožňuje provádět složité úkoly se zvýšenou přesností. [4]

Drony a letecké mapování

Drony vybavené technologií 3D rekonstrukce přinášejí revoluci do leteckého mapování a geodézie. Tato bezpilotní letadla pořizují snímky s vysokým rozlišením a vytvářejí podrobné 3D modely krajiny, infrastruktury a přírodního prostředí. To umožňuje přesnější a efektivnější sběr dat pro různé aplikace, včetně plánování měst, zemědělství a monitorování životního prostředí. [4]

Rychlostní radary a řízení dopravy

Rychlostní radary a systémy řízení dopravy využívají 3D rekonstrukci pro přesné sledování a vymáhání dodržování rychlostních limitů. Díky rekonstrukci 3D geometrie vozidel a vozovek mohou tyto systémy přesně měřit rychlost vozidel a odhalovat dopravní přestupky, což přispívá ke zvýšení bezpečnosti silničního provozu a řízení dopravních toků.

3 TECHNOLOGICKÉ MOŽNOSTI TRANSFORMACE 2D OBJEKTŮ DO 3D

Všechny níže zmíněné technologické možnosti transformace informace z dvourozměrného do třírozměrného prostoru řeší stejný základní problém. K jeho řešení ale přistupují jinými způsoby. Tyto přístupy mají odlišné vlastnosti.

Tímto základním problémem je určení hloubky neboli "depth estimation". Tato vlastnost obrazu určuje vzdálenost jednotlivých bodů od pozorovatele. Za pozorovate v tomto kontextu považuji jakékoliv zařízení, které zpracovává obraz za účelem 3D rekonstrukce.

Pokud máme způsob jakým zjišťovat hloubku jednotlivých bodů, tak je poměrně snadné z těchto dat vytvořit bodový mrak (point cloud). A s ním dál pracovat a transformovat ho (v příkladu mé praktické části do formátu 3D mesh).

3.1 Fotogrammetrie

Základní popis a princip

Fotogrammetrie je na tomto seznamu technologií uvedena jako první, protože se jedná o obecně nejrozšířenější a nejvíce užívanou metodu a to hlavně díky její obecnosti a jednoduchosti. Zároveň se také jedná o techniku, která dosahuje poměrně přesných a spolehlivých výsledků.

Principem fotogrammetrie je získání prostorových informací z obrazů zachycených fotoaparátem nebo jiným obrazovým zařízením. Tato technika využívá znalostí o geometrii a optice, aby přemapovala 2D obrazy na 3D modely. Jedním z klíčových prvků fotogrammetrie je schopnost měřit vzdálenosti, velikosti a tvary objektů z jejich obrazů. [5]

Při transformaci 2D do 3D pomocí fotogrammetrie se provádí několik kroků. Nejprve se zachytí série fotografií objektu z různých úhlů a pozic. Poté se tyto fotografie zpracují pomocí algoritmu, který identifikuje charakteristické body na každé fotografii a sleduje jejich polohu v různých snímcích. [2] [1]

Po identifikaci charakteristických bodů a jejich polohy se provede tzv. triangulace, což je proces, při kterém se na základě pozic těchto bodů ve více snímcích určuje jejich prostorová poloha. Tímto způsobem je možné rekonstruovat 3D model objektu. [2]

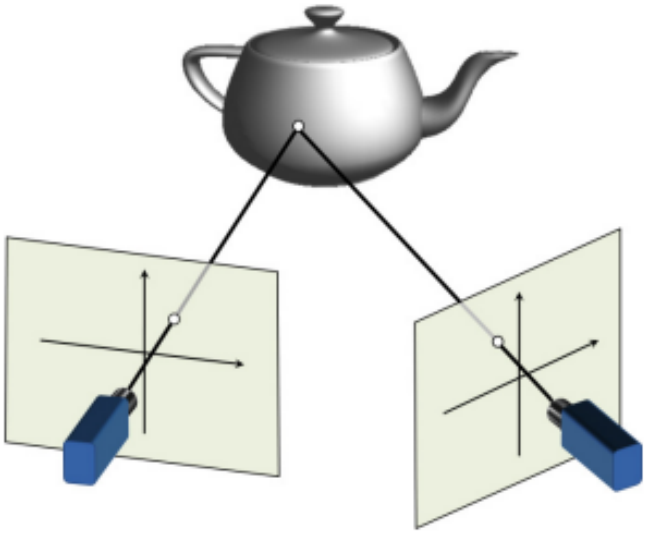
Fotogrammetrie je velmi obecná metoda a má mnoho variant. Základní rozdělení těchto metod spočívá v přístupu k zjišťování informací. Jedná se o pasivní triangulaci a aktivní triangulaci. [5]

Pasivní triangulace je metoda, která nevyžaduje speciální osvětlení nebo projekční zařízení. Tato technika je základem pro metody jako stereovize. Fotogrammetrie

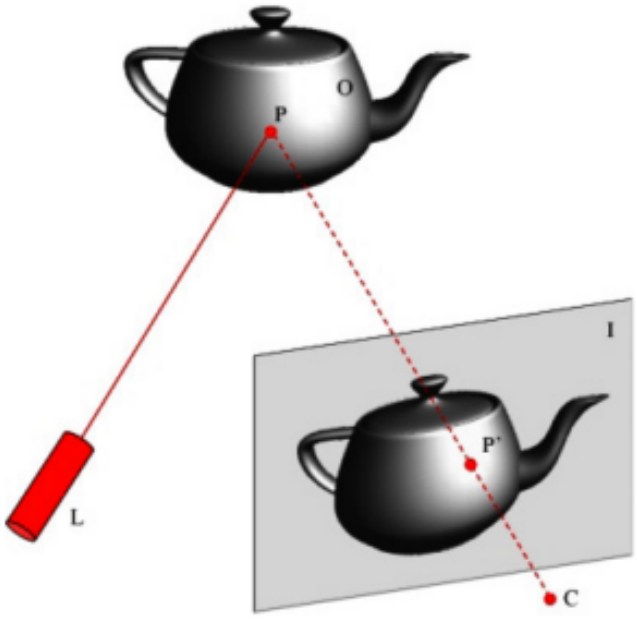
využívá více kamer nebo pouze jednu kameru v různých polohách k zachycení scény z různých úhlů. Získané obrazy se pak analyzují na shodu bodů, jejichž 3D poloha je určena průsečíkem spojnic těchto bodů. Klíčem k úspěchu pasivní triangulace je schopnost správně určit korespondence mezi bodovými páry ve více obrazech. [5]

Aktivní triangulace vylepšuje proces získávání dat pomocí kontrolního osvětlení nebo projekčních zařízení. Tato metoda často používá jednu kameru společně s laserovým projekčním zařízením, které vrhá specifický vzor světla (např. bod, čáru, nebo více složitých geometrických vzorů) na objekt. Poloha a orientace jak projekčního zařízení, tak kamery jsou přesně známé, což umožňuje přesné vypočítání 3D pozic bodů, na které laser ukazuje, založené na průsečících paprsků světla a zorného pole kamery. [5]

Pasivní triangulace je obecně jednodušší na implementaci a méně nákladná, protože nevyžaduje speciální osvětlení. Je však více náchylná k chybám v důsledku nedostatečného osvětlení nebo složitého pozadí, které může ztížit korespondenční analýzu. Aktivní triangulace poskytuje vyšší přesnost a je méně závislá na vnějších světelných podmínkách, ale vyžaduje dražší a složitější vybavení. [5]



Obrázek 3.1 Pasivní triangulace [5]



Obrázek 3.2 Aktivní triangulace [5]

Pro úspěšnou 3D rekonstrukci s pomocí fotogrammetrických metod je nutná znalost parametrů pozorovacího zařízení. Tyto parametry popisují vlastnosti kamer a jejich pozice vůči sobě. Tyto metody využívají znalosti těchto parametrů, jako jeden ze základních vstupů pro výpočet hloubky. Do této skupiny fotogrammetrických metod patří i stereovize, která toto provádí ze dvou obrazů. Tuto metodu budu popisovat v pozdějších sekcích této práce. [5] [2]

Základní Parametry

Dle Hartleyho a Zissermana [2] níže popisuji základní parametry kamer pro 3D rekonstrukci pomocí fotogrammetrických metod.

Ohnisková délka

Ohnisková vzdálenost, často označovaná jako focal length, je vzdálenost mezi středem čočky kamery a snímačem (neboli filmem nebo digitálním snímačem), když je kamera zaostřená na nekonečno. V digitální fotografii a videu určuje ohnisková vzdálenost zorný úhel a zvětšení obrazu. Je to základní charakteristika čočky, která má přímý dopad na rekonstrukci scény z obrazových dat.

Baseline

Baseline v kontextu stereo vidění a fotogrammetrie označuje vzdálenost mezi dvěma kamerami (nebo dvěma různými pozicemi téže kamery v různých časech), které jsou

použity k získání dvou pohledů na scénu. Tato vzdálenost je typicky měřena v horizontálním směru mezi optickými středy obou kamer.

Překrytí snímků

Pro efektivní rekonstrukci je nutné, aby se snímky překrývaly alespoň z 60-70 %. To zajišťuje dostatek společných referenčních bodů pro správné sloučení snímků.

Další důležité Parametry

Kvalita obrazu a rozlišení

Vyšší rozlišení a kvalita obrazu jsou výhodné pro identifikaci a spárování jednotlivých bodů mezi snímky.

Osazení kamery

Kalibrace kamery je nezbytná pro odstranění zkreslení způsobeného optickými charakteristikami a pro určení přesné polohy a orientace kamery.

Osvětlení

Jednotné a dobře rozložené osvětlení je klíčové pro minimalizaci stínů a odlesků, což pomáhá zlepšit interpretaci a zpracování obrazu.

Výhody a Nevýhody Fotogrammetrie

Následuje seznam výhod a nevýhod fotogrammetrie vůči ostatním metodám dle Simona Wyatt-Spratta [6], Richarda Szeliského [4] a Wilfrieda Lindera. [7]

Výhody fotogrammetrie

1. **Nízké náklady:** Fotogrammetrie vyžaduje pouze standardní fotografické vybavení, jako je fotoaparát nebo kamera. To je často levnější alternativa ve srovnání s drahými specializovanými senzory jako je LIDAR, který vyžaduje sofistikovanější technologii a investice.

2. **Vysoká dostupnost:** Fotoaparáty a kamery jsou široce dostupné a snadno použitelné. Tato přístupnost umožňuje širokému spektru uživatelů v různých oblastech aplikovat fotogrammetrii.

3. **Snadná implementace:** Proces fotogrammetrie je relativně snadný na implementaci, což je podpořeno dostupností specializovaného softwaru, který usnadňuje práci s fotografiemi a 3D modelováním.

4. **Vysoké rozlišení:** Díky moderním vysokorychlostním fotoaparátům je možné dosáhnout vysokého rozlišení výsledného 3D modelu, což je ideální pro detailní vizualizace a analýzy.

Nevýhody fotogrammetrie

1. **Citlivost na osvětlení:** Fotogrammetrie může být výrazně citlivá na změny osvětlení, což může negativně ovlivnit kvalitu a přesnost 3D modelů.

2. **Závislost na texturách:** Metoda je závislá na dostatečné texturaci povrchů objektů. Objekty s hladkými, monotónními nebo chudě texturovanými povrchy mohou být pro rekonstrukci problematické.

3. **Přesnost:** Přesnost fotogrammetrických rekonstrukcí může být ovlivněna různými faktory, včetně odrazivosti povrchů, kvality fotoaparátu a přesné kalibrace zařízení.

4. **Omezení v prostoru:** Fotogrammetrie může čelit omezením v prostředích, kde je obtížné získat dostatečný počet vhodných fotografií, například v uzavřených nebo těžko přístupných oblastech.

Použití fotogrammetrie

Fotogrammetrie je široce využívána v mnoha odvětvích, včetně geodézie, kartografie, archeologie, inženýrství, herního průmyslu a virtuální reality, díky své schopnosti rychle a efektivně vytvářet přesné 3D modely z běžných fotografií. [4]

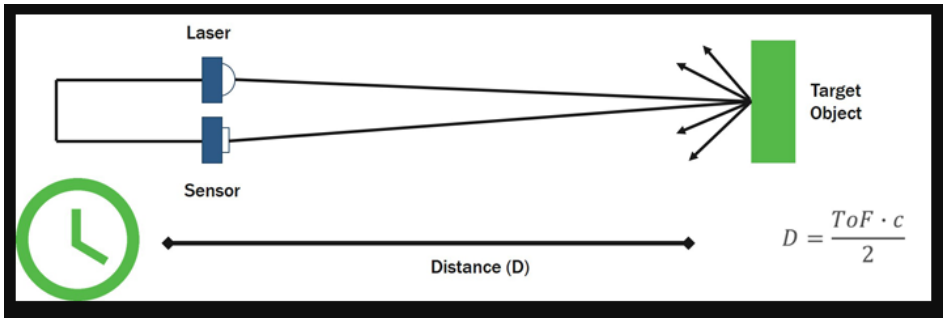
3.2 LIDAR

Základní popis a princip

LIDAR, což je zkratka pro „Light Detection and Ranging“, je pokročilá technologie měření vzdáleností pomocí laseru, široce používaná pro 3D rekonstrukci objektů a terénu. LIDAR je průmyslově velmi rozšířená technologie díky své vysoké přesnosti a schopnosti rychle generovat detailní 3D mapy. [8]

Hlavním principem LIDARu je emise laserových paprsků a následné zachycení odraženého světla senzorem. Časový interval mezi vysláním a přijetím světla se používá k výpočtu vzdálenosti. LIDAR tedy poskytuje rychlé a přesné měření vzdáleností, což umožňuje vytvářet komplexní trojrozměrné modely skenované oblasti. [8]

Využití LIDARu k transformaci 2D dat do 3D prostoru je založeno na jeho schopnosti sbírat prostorové informace z okolí. Při snímání prostředí LIDARem jsou získávána data ve formě bodových mračen, která obsahují informace o vzdálenosti a pozici každého naměřeného bodu. Tyto body jsou poté využity k vytvoření detailní 3D reprezentace objektů a prostředí. [8] [9]



Obrázek 3.3 Základní princip LIDARu [9]

Na předchozím obrázku je znázorněn princip měření vzdálenosti pomocí LIDARu (Light Detection and Ranging). Uprostřed je umístěn senzor s laserem, který vysílá světelné paprsky směrem k cílovému objektu, zde zobrazenému jako zelený obdélník na pravé straně. Světlo je od objektu odraženo zpět k senzoru. [9]

Vzdálenost (D) mezi senzorem a cílovým objektem je určena využitím Time of Flight (ToF) - doby letu, což je čas, který trvá laserovému paprsku cestovat od senzoru k objektu a zpět. Vzdálenost se vypočítá vynásobením ToF rychlostí světla (c) a následným

dělením dvěma, protože paprsek cestuje tam a zase zpět. [9]

Na levé straně je symbol hodin, což naznačuje měření času, které je zásadní pro určení vzdálenosti metodou ToF. [9]

Výhody a Nevýhody LIDARu

V následující sekci je rozpis výhod a nevýhod LIDARu vůči ostatním metodám dle Simona Wyatt-Spratta [6] a Santiaga Royo a Marie Ballesta-Garciové [10].

Výhody LIDARu

1. **Rychlost a efektivita:** LIDARy jsou schopny rychle zachytit velké množství dat a generovat 3D modely ve velmi krátkém čase. To je zvláště užitečné při mapování rozsáhlých oblastí nebo při sledování pohybu objektů v reálném čase.

2. **Vysoká přesnost:** LIDAR poskytuje velmi přesné měření vzdáleností v rychlém časovém sledu, což je ideální pro komplexní a rozsáhlé 3D mapování.

3. **Nezávislost na světelných podmínkách:** LIDARy používají vlastní zdroj světla (laser), což znamená, že jsou méně citlivé na změny osvětlení v prostředí. Na rozdíl od fotogrammetrie, která závisí na dostatku světla a vhodných texturách pro extrakci rysů, LIDARy mohou fungovat i v temných nebo špatně osvětlených prostředích.

4. **Schopnost pronikat určitými materiály:** LIDARy jsou schopny průchodu většinou materiálů, což znamená, že mohou snadno zachycovat informace o povrchu i skrytých strukturách objektů, což může být obtížné pro metody založené na vizuálních datech, jako je fotogrammetrie. Příkladem může být mapování terénu pod vegetací.

5. **Snadná automatizace:** LIDARy mohou být snadno integrovány do autonomních systémů a robotů, což umožňuje jejich použití pro průmyslové aplikace, jako je inspekce a navigace.

Nevýhody LIDARu

1. **Vysoké náklady:** Přístroje LIDAR jsou obvykle dražší než standardní fotogrammetrické systémy kvůli sofistikované technologii a potřebě speciálního zařízení.

2. **Omezený rozsah:** LIDARy mají omezený dosah, zejména v prostředích s vysokou mírou průchodu, jako jsou mlha, déšť nebo sníh. Tyto podmínky mohou omezit schopnost LIDARů zachytit data přesně nebo úplně.

3. **Potřeba vyrovnaného povrchu:** Pro přesné měření vzdálenosti je ideální, aby byl povrch, ze kterého je paprsek odražen, rovný a rovnoměrný. Nerovnosti povrchu, jako jsou sklo, voda nebo prach, mohou ovlivnit přesnost měření LIDARu.

4. **Omezené rozlišení textury:** LIDARy jsou schopny získat velmi přesné informace o geometrii objektů, ale mají omezené schopnosti zachycení textury povrchu. To může být nevýhoda pro aplikace, které vyžadují detailní vizuální informace o povrchu, jako je například archeologie.

5. **Zranitelnost vůči interferenci:** LIDARy mohou být náchylné k interferenci způsobené jinými LIDARy nebo laserovými zdroji v okolí, což může mít za následek zkreslení nebo ztrátu dat.

6. **Komplexní zpracování dat:** Velké objemy dat získané LIDARem vyžadují pokročilé softwarové nástroje pro zpracování, což může být časově náročné a vyžaduje specializované znalosti.

Aplikace LIDARu

Tato inovativní metoda využívá laserového paprsku k měření vzdáleností a vytváření detailních 3D map prostředí. I když LIDAR původně vznikl pro geodetické účely, jeho aplikace se od té doby rozšířily do mnoha oblastí, od autonomních vozidel po archeologii. Zde je pohled na některé z nejzajímavějších možností použití LIDARu. [10] [9]

LIDARy hrají klíčovou roli v rozvoji autonomních vozidel. Tyto senzory umožňují vozidlům „vidět“ okolní prostředí s neuvěřitelnou přesností a detaily. Tímto způsobem LIDAR umožňuje vozidlům efektivně navigovat v různých podmínkách, včetně změn povětrnostních podmínek a nočních jízd. [10]

Možnosti použití LIDARu pro přesné měření vzdáleností je velmi rozšířené a obecné. Tato technologie se využívá v geodetických průzkumech, stavebnictví, topografii a dokonce i při archeologických vykopávkách. LIDAR umožňuje vytvářet detailní mapy terénu s minimální chybou a značně zrychluje procesy, které by jinak byly pracné a nákladné. [10] [9]

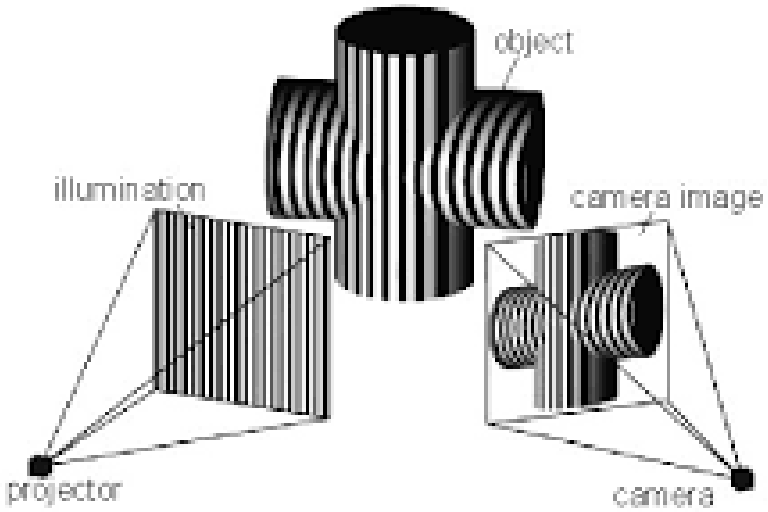
LIDARy jsou důležitým nástrojem pro monitorování životního prostředí. Tyto senzory mohou například detekovat znečištění ovzduší, sledovat hladiny vody nebo dokonce monitorovat lesní porosty. Tímto způsobem LIDAR umožňuje lepší porozumění a ochranu naší planety. [10]

V oblasti virtuální reality a animace se LIDAR využívá k vytváření realističtějších a detailnějších virtuálních světů. Tato technologie umožňuje skenovat reálné objekty a prostředí a převést je do digitálního prostoru s přesností a detaily. [9]

3.3 Metoda strukturovaného světla

Základní popis a princip

Metoda strukturovaného světla je moderní technikou v oblasti 3D skenování a modelování, která umožňuje detailní rekonstrukci tvarů objektů. Principem metody je promítání specificky strukturovaného světla, jako jsou mřížky nebo různé vzory, na objekty, jejichž třírozměrný tvar se má zjistit a definovat. Deformace světelných vzorů, které jsou promítány na objekty a odraženy zpět do senzorů, jsou analyzovány a používány pro výpočet tvarů a rozměrů objektů ve 3D prostoru. [11]



Obrázek 3.4 Schematický diagram metody strukturovaného světla ukazující promítání vzoru na objekt a jeho následnou deformaci [11]

3D Rekonstrukce Metodou Strukturovaného Světla

Metoda strukturovaného světla využívá projekci známého vzoru (například pruhy, mřížky nebo body) na scénu. Sledováním deformací tohoto vzoru na nehomogenních površích lze určit tvar objektů ve scéně. [12]

Základní Principy

Když světlo dopadá na objekt, jeho vzor se deformuje v závislosti na 3D tvaru objektu. Tato deformace je zaznamenána kamerou z jiného úhlu než je úhel projekce. Vzdálenost mezi projektorem a kamerou je pevně daná a známá. [12]

Výpočet 3D Souřadnic

Pro metodu strukturovaného světla se obvykle používá triangulační metoda k určení 3D souřadnic bodů na objektu z deformovaného vzoru, který je projektován na objekt.

Zde je nástin způsobu výpočtu hloubky bodu p od kamery:

Nechť Z_p je hloubka bodu p, f je ohnisková vzdálenost kamery, B je vzdálenost mezi projektorem a kamerou (baseline), a d je rozdíl ve vertikální nebo horizontální pozici bodů p a q v obraze. [12]

$$Z_p = \frac{f \cdot B}{d} \tag{3.1}$$

Předchozí rovnice popisuje výpočet hloubky bodu p od kamery Z_p.

Hloubková Mapa

Pro každý pixel v obraze, kde je vidět projekční vzor, se spočte hloubka pomocí výše uvedené rovnice. Výsledkem je hloubková mapa, která reprezentuje 3D tvar objektu.

Přesnost této metody a schopnost získávat detailní informace o povrchu objektů ji činí vhodnou pro použití v průmyslové metrologii, počítačovém vidění a digitální archeologii. Specifické aplikace zahrnují kontrolu kvality ve výrobě, design interiérů, rekonstrukci archeologických nálezů, a vytváření digitálních dvojčat pro virtuální realitu. [12] [11]

Výhody a nevýhody

V této části popisuji výhody a nevýhody metody strukturovaného světla pro 3D rekonstrukci dle společnosti Bitfab [11] a Davida L. Andrewse [12].

Výhody metody strukturovaného světla

1. Vysoká přesnost: Strukturované světlo je schopné získávat velmi přesné informace o tvaru a povrchu objektů.
2. Rychlost skenování: Tato metoda umožňuje rychlé skenování, což je neocenitelné pro průmyslové aplikace a kontrolu kvality.
3. Nízké náklady: Strukturované světlo může být cenově výhodnější ve srovnání s jinými metodami, jako je například LIDAR.
4. Snadná implementace: Systémy strukturovaného světla lze obvykle implementovat s menším množstvím specializovaného vybavení.

Nevýhody metody strukturovaného světla

1. Omezená účinnost v proměnlivých světelných podmínkách: Kvalita získaných dat může být negativně ovlivněna světelnými podmínkami.

2. Potíže s některými povrchy: Určité materiály, jako jsou lesklé nebo průhledné povrchy, mohou narušovat přesnost měření.

3. Omezený dosah: Metoda může mít omezení vzdálenosti, s jakou může efektivně pracovat, což omezuje její použití na menší objekty.

4. Citlivost na překážky: Přítomnost překážek v prostředí může komplikovat proces skenování a rekonstrukce.

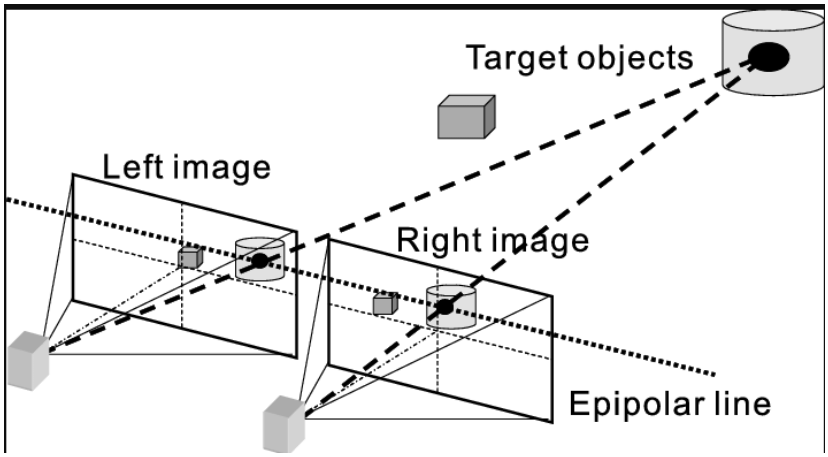
Použití

Strukturované světlo nachází uplatnění v řadě průmyslových a výzkumných oblastí. Je používáno pro rychlou a přesnou kontrolu kvality ve výrobních procesech, v digitálním modelování pro animaci a herní průmysl, ve zdravotnictví pro plánování chirurgických zákroků, a v archeologii pro digitální rekonstrukci a analýzu archeologických nálezů. [11]

3.4 Stereovize

Základní popis a princip

Stereovize je technika zpracování obrazu, která napodobuje lidský binokulární zrak, využívajíc dvou kamer umístěných v určité vzdálenosti od sebe pro vytváření 3D modelů. Na rozdíl od ostatních možných metod fotogrammetrie, která může využít jedinou kameru z různých úhlů, stereovize vytváří prostorové informace synchronizovaným zachycením obrazů v reálném čase. Tyto obrazy jsou poté analyzovány na shodu bodů, které se využívají pro výpočet hloubkové mapy scény. Jedná se ale o podmnožinu fotogrammetrie. Tedy přesněji o jednu z pasivních metod. [5] [2]



Obrázek 3.5 Diagram principu stereovize ukazující dvě kamery zachycující obraz ze dvou různých úhlů pohledu. [5]

Zatímco fotogrammetrie může být vhodná pro aplikace s pevnými objekty a vyžaduje komplexní zpracování po akvizici dat, stereovize se často používá pro dynamické scény a nabízí rychlé zpracování v reálném čase, což je nezbytné pro robotické navigace a interaktivní aplikace, jako je rozšířená realita. [5] [2]

Výpočet hloubky pomocí triangulace

Tato část popisuje základní principy a matematické postupy pro výpočet hloubky jednoho bodu pomocí fotogrammetrie.

Matematický zápis triangulace

Triangulace je proces, kde se z dvou různých snímků a znalosti poloh a orientací kamer určuje 3D pozice bodů. Předpokládáme, že máme dvě kamery s ohniskovými délkami f_1 a f_2, které jsou od sebe vzdáleny o baseline B. Pokud máme měření úhlu θ_1 a θ_2 pro daný bod z každé kamery, hloubku D bodu můžeme vypočítat jako:

$$D = \frac{B \cdot f_1 \cdot f_2}{f_1 \cdot \tan(\theta_2) + f_2 \cdot \tan(\theta_1)} \tag{3.2}$$

Tento výpočet předpokládá, že kamery jsou perfektně kalibrovány a že neexistuje žádné optické zkreslení.

Matematický model ukazuje ideální případ. Ve skutečných aplikacích je třeba zohlednit další faktory, jako je geometrické a radiometrické zkreslení, které může významně ovlivnit přesnost měření. [5]

Základní problém stereovize

V této části práce je popsán základní problém, který je nutno vyřešit pro úspěšné provedení 3D rekonstrukce pomocí stereovize. Tímto problémem je nalezení bodů na obou snímcích, které zobrazují stejný objekt. Tento proces se nazývá "feature matching". Výsledkem tohoto procesu je poté disperita, která popisuje změnu pozice stejného bodu na obou obrázcích (jmenovatel v předchozí rovnici pro výpočet hloubky bodu). Pro výpočet disperity v mém projektu používám konvuluční neuronovou síť. [5] [2]

Feature Matching

Feature matching je klíčový krok v procesu stereovize, kde cílem je identifikovat odpovídající si body (features) na dvou různých snímcích téže scény. Tyto body jsou základem pro výpočet disparity a následnou 3D rekonstrukci scény. [5] [2]

Detekce features

Prvním krokem ve feature matchingu je detekce features, které mohou být robustní proti změnám v osvětlení, rotaci a škálování. Běžně používané detektory zahrnují SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features), a ORB (Oriented FAST and Rotated BRIEF), které poskytují soubor klíčových bodů spolu s jejich deskriptory. [5] [2]

Matching deskriptorů

Po detekci features následuje korespondence pomocí matchingu deskriptorů. Deskriptory, které popisují vzhled jednotlivých features, umožňují porovnat, zda dva různé body na různých snímcích odpovídají stejnému bodu ve scéně. Matching může být prováděn různými algoritmy, jako jsou Brute-Force matcher nebo FLANN (Fast Library for Approximate Nearest Neighbors) matcher. [2]

Filtrace nesprávných spojení

Přestože proces matchingu může nalézt mnoho korespondencí, některé z nich mohou být nesprávné (tzv. outliers). Pro vyřazení těchto nesprávných spojení se často používají metody jako RANSAC (Random Sample Consensus), která pomáhá odstranit spojení, která neodpovídají odhadovanému modelu geometrie mezi snímky. [2]

Tímto způsobem feature matching poskytuje základní data pro výpočet disparity a následnou rekonstrukci 3D modelu scény. Bez efektivního a přesného feature matchingu by bylo těžké dosáhnout kvalitních výsledků v 3D rekonstrukci pomocí stereovize. [5]

Kroky k úspěšné 3D rekonstrukci pomocí stereovize

V této sekci popisuji jednotlivé kroky k úspěšné 3D rekonstrukci pomocí stereovize dle T. Moonse, L. Van Goola a M. Vergauwena [5] a Richarda Hartleyho a Andrewa Zissermana [2].

1. **Kalibrace kamer:** Určení vnitřních parametrů kamer, jako jsou ohnisková vzdálenost a optické středisko, a vnějších parametrů, jako je vzájemná poloha a orientace kamer.

2. **Synchronizace kamer:** Zajištění, že obě kamery zachytávají obrazy ve stejném okamžiku, což je kritické pro dynamické scény.

3. **Snímání obrazu:** Získání páru obrazů scény z obou kamer, které jsou následně použity pro rekonstrukci 3D informace.

4. **Detekce korespondujících bodů:** Použití algoritmů pro nalezení odpovídajících bodů (feature matching) mezi dvěma snímky.

5. **Výpočet disparity:** Stanovení rozdílů v pozici stejných bodů na obou snímcích, což umožňuje odhad hloubky.

6. **Rekonstrukce hloubkové mapy:** Vytvoření 3D mapy pomocí disparity a kalibračních dat k získání hloubkových informací pro každý bod scény.

7. **Optimalizace a čištění dat:** Použití filtrů a optimalizačních technik k odstranění šumu a vylepšení kvality rekonstrukce.

Výhody a nevýhody

V této sekci popisuji výhody a nevýhody stereovize pro 3D rekonstrukci vůči ostatním metodám dle Simona Wyatt-Spratta [6] a Santiaga Royo a Marie Ballesta-Garciové [10].

Výhody stereovize

1. Nízké náklady na zařízení ve srovnání s LIDARem, což činí stereovizi dostupnou pro širokou škálu aplikací.

2. Relativní jednoduchost implementace díky široce dostupným kamerám a pokročilým algoritmům zpracování obrazu.

3. Vysoké rozlišení umožňující detailní zobrazení textur a povrchových detailů.

4. Schopnost zachycovat dynamické scény v reálném čase, což je klíčové pro aplikace vyžadující rychlou odezvu.

Nevýhody stereovize

1. Omezený dosah a přesnost ve srovnání s LIDARem, což omezuje její použití v určitých průmyslových aplikacích.

2. Citlivost na okolní světelné podmínky, které mohou ovlivnit kvalitu rekonstrukce.

3. Potřeba přesné kalibrace a synchronizace kamer pro efektivní fungování.

4. Komplexní zpracování obrazu může být náročné na výpočetní zdroje, zejména při vysokém rozlišení.

3.5 Monokulární hloubkové odhady

Základní popis a princip

Monokulární hloubkové odhady jsou metoda zpracování obrazu, která umožňuje 3D rekonstrukci scény z jediného 2D obrazu. Na rozdíl od stereovize, která vyžaduje dva obrazy pro určení hloubky, monokulární odhady využívají sofistikované algoritmy k odhadu hloubky na základě vizuálních klíčů obsažených v jediném obrazu. Tyto klíče zahrnují perspektivní zmenšování, texturové gradienty, stínování a další informace o kontextu scény. [13]

Monokulární hloubkové odhady poskytují cenné informace pro aplikace, kde není možné použít stereo kamery nebo kde prostorové omezení brání použití více kamer. Tato metoda je zvláště užitečná v robotice, autonomních vozidlech a mobilních aplikacích, kde je potřeba rychlé a efektivní zpracování obrazu. [13] [14]

Kroky k úspěšné 3D rekonstrukci pomocí monokulárních hloubkových odhadů

Pro úspěšnou 3D rekonstrukci pomocí monokulárních hloubkových odhadů je dle [14], [16] a [17] třeba postupovat podle následujících kroků.

1. **Akvizice obrazu:** Získání kvalitního 2D snímku scény, který bude použit pro hloubkový odhad.
2. **Předzpracování obrazu:** Aplikace filtrů a korekcí pro zlepšení kvality obrazu a zvýraznění důležitých vizuálních klíčů.
3. **Analýza obrazu:** Využití algoritmů pro identifikaci a interpretaci vizuálních klíčů, které naznačují hloubku (například rozmazání, perspektivní změny).
4. **Odhad hloubky:** Použití algoritmů strojového učení nebo tradičních přístupů pro výpočet hloubkových hodnot z analýzy obrazu.
5. **Rekonstrukce 3D modelu:** Transformace hloubkových dat do 3D modelu scény.
6. **Optimalizace a validace:** Aplikace post-procesních technik pro zlepšení přesnosti a realističnosti 3D modelu a jeho validace proti známým měřítkům nebo modelům.

Výhody a nevýhody

V této sekci se práce věnuje výhodám a nevýhodám metody monokulárních hloubkových odhadů dle [13].

Výhody monokulárních hloubkových odhadů

1. **Nižší hardwarové nároky:** Nevyžaduje drahé nebo komplexní vybavení, stačí jedna kamera.
2. **Flexibilita v aplikacích:** Možnost použití v mobilních zařízeních a aplikacích s omezeným prostorem.
3. **Široké využití:** Uplatnění v oblastech, kde není možné nasadit více kamer.

Nevýhody monokulárních hloubkových odhadů

1. **Nižší přesnost ve srovnání se stereovize a LIDARem.**
2. **Závislost na kvalitě a typu vstupního obrazu.**
3. **Komplexnost algoritmů:** Výzvy spojené s přesným odhadem hloubky z jediného obrazu.

Použití

Monokulární hloubkové odhady se využívají v robotice pro navigaci a objektovou detekci a v autonomních vozidlech pro detekci a reakci na prostředí. [13]

4 TRANSFORMACE INFORMACE Z 2D DO FORMÁTU 3D MESH

Tato kapitola se zabývá převodem 2D informace do 3D reprezentace ve formátu 3D mesh. Nejprve definuji problém, popíši vstupní data a následně se budu věnovat různým algoritmům, které tuto transformaci řeší.

4.1 Popis problému

V předchozí kapitole jsem se věnoval popisu různých metod, které slouží k odhadnutí hloubky z 2D informace. Výstupem těchto metod je tedy mračno bodů, u kterých známe i vzdálenost od pozorovacího zařízení. Tento formát se nazývá "point-cloud". [15]

Nejčastější číselná reprezentace bodového mračna (point cloudu) využívá formát zvaný "XYZ", kde každý bod je reprezentován třemi číselnými hodnotami odpovídajícími jeho souřadnicím ve třech dimenzích: X, Y a Z. Tento formát může být rozšířen o další informace, jako jsou barva (RGB), intenzita, a normály pro každý bod. [2]

Výběr datové reprezentace může záviset na aplikaci a potřebných operacích, ale XYZ formát je velmi běžný díky své jednoduchosti a přímé podpoře v mnoha softwarových nástrojích a knihovnách pro zpracování 3D dat. [2]

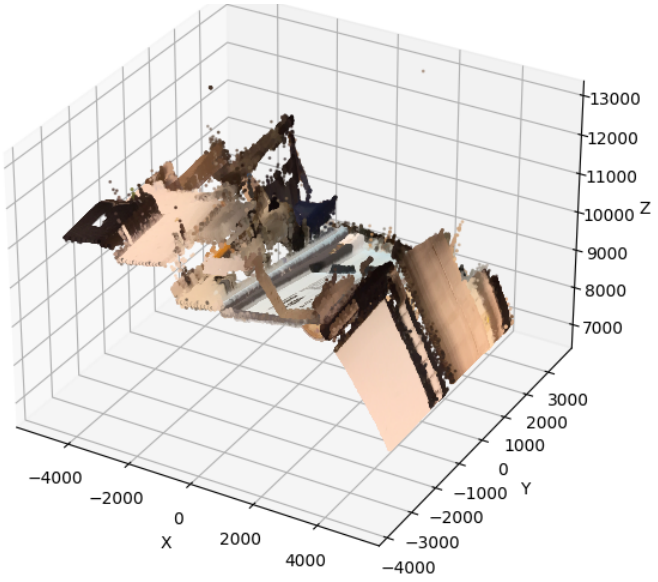
Ukázka datové reprezentace náhodně generovaného bodového mračna v pythonu

Následující kód spolu s komentáři ukazuje generování náhodného bodového mračna s pomocí knihovny numpy.

```
import numpy as np
# Vytvoření náhodného point cloudu se 100 body
# Každý bod má souřadnice X, Y, Z
# 100 bodů, každý se 3 souřadnicemi
point_cloud_xyz = np.random.rand(100, 3)
# Vytvoření point cloudu s barvami
# Každý bod má souřadnice X, Y, Z a barvy R, G, B
# 100 bodů, každý se 3 souřadnicemi a 3 barvovými kanály
point_cloud_xyzrgb = np.random.rand(100, 6)
```

Z předešlé definice je jasné, že bodové mračno nemá povrchovou strukturu. Jedná se tedy pouze o skupinu bodů v prostoru. Pro 3D rekonstrukci včetně povrchové struktury je nutné toto mračno převést do formátu 3D mesh. [18]

Pro reprezentaci 3D mesh můžete použít různé datové struktury, které závisí na konkrétních potřebách a nástrojích, které máte k dispozici. Mesh se obecně skládá ze tří základních prvků: vrcholů, hran a stěn (nebo polygonů). [1]

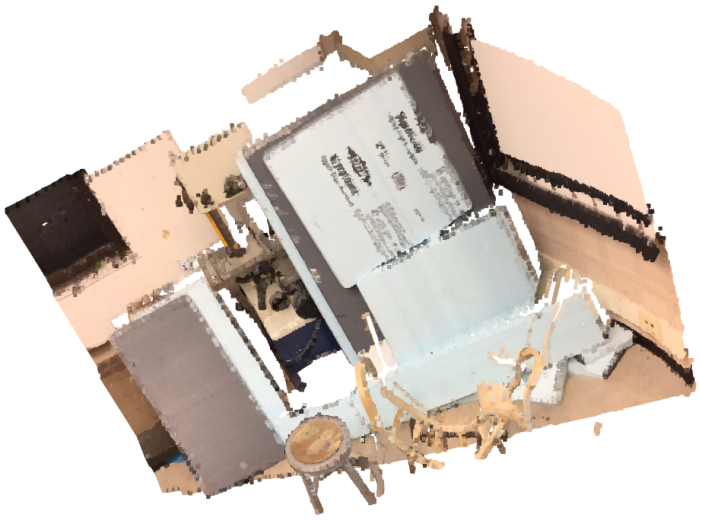


Obrázek 4.1 Příklad vizualizace bodového mračna (vlastní zdroj)

Ukázka datové reprezentace 3D mesh v pythonu

Následující ukázka kódu nastinuje datovou strukturu 3D modelu ve formátu 3D mesh.

```
import numpy as np
# Seznam vrcholů (x, y, z souřadnice) ve formátu numpy array
# Každý vrchol je definován jako bod ve 3D prostoru
vertices = np.array([
    [1, 1, 1],     # Vrchol 0 na pozici (1, 1, 1)
    [1, -1, 1],    # Vrchol 1 na pozici (1, -1, 1)
    [1, 1, -1],    # Vrchol 2 na pozici (1, 1, -1)
    [1, -1, -1],   # Vrchol 3 na pozici (1, -1, -1)
    [-1, 1, 1],    # Vrchol 4 na pozici (-1, 1, 1)
    [-1, -1, 1],   # Vrchol 5 na pozici (-1, -1, 1)
    [-1, 1, -1],   # Vrchol 6 na pozici (-1, 1, -1)
    [-1, -1, -1]   # Vrchol 7 na pozici (-1, -1, -1)
])
# Seznam trojúhelníků definovaných indexy vrcholů
# Každá trojice indexů odpovídá jednomu trojúhelníku v mesh
faces = np.array([
    [0, 1, 2],   # Trojúhelník tvořený vrcholy 0, 1 a 2
    [1, 3, 2],   # Trojúhelník tvořený vrcholy 1, 3 a 2
    [4, 5, 6],   # Trojúhelník tvořený vrcholy 4, 5 a 6
    [5, 7, 6]    # Trojúhelník tvořený vrcholy 5, 7 a 6
])
# Tato struktura umožňuje efektivní ukládání a manipulaci s meshem
# Indexy v 'faces' odkazují přímo na pozice ve 'vertices',
# což umožňuje efektivní změny a snadnou správu mesh dat.
```



Obrázek 4.2 Příklad vizualizace 3D mesh (vlastní tvorba)

4.2 Porovnání 3D Mesh a Point Cloud

Definice a vlastnosti

3D Mesh je typ 3D modelu, který se skládá z polygonů, obvykle trojúhelníků, které jsou propojeny svými vrcholy a hranami. Tyto polygony tvoří spojitý povrch, který reprezentuje tvar objektu ve 3D prostoru. Mesh je široce používán v grafice a simulacích pro detailní a vysoce definovanou strukturu povrchu, což je užitečné pro vizualizace, animace a fyzikální simulace. [1]

Point Cloud je soubor bodů ve 3D prostoru. Každý bod má svou pozici ve třech rozměrech. Některá bodová mračna mohou také obsahovat další data, jako jsou barvy nebo intenzity každého bodu. Tato mračna jsou často výsledkem skenování reálného světa pomocí technologií, které byly zmíněny v minulé kapitole.

Meshe reprezentují spojité povrchy, zatímco point clouds jsou diskrétní soubory bodů. [1]

4.3 Algorytmy pro převod bodového mračna do formátu 3D mesh

V této části práce popisuji různé metody a algoritmy pro konverzi bodového mračna (point cloud) do formátu 3D mesh. U všech algoritmů jsou zmíněny výhody i nevýhody. Příklady pseudokódu jsou zahrnuty tam, kde je to vhodné.

Každý z těchto algoritmů má specifické využití a výběr vhodného algoritmu závisí na konkrétních charakteristikách dat a požadovaných vlastnostech výsledného modelu.

4.3.1 Poisson Surface Reconstruction

Popis

Tato metoda se zakládá na statistickém přístupu k interpolaci povrchu, který modeluje implicitní povrch pomocí 3D Poissonovy rovnice.

Poissonova metoda pro rekonstrukci povrchu je založena na myšlence, že povrch lze modelovat jako gradientové pole skalární funkce, kde normály k povrchu odpovídají gradientům této funkce. Proces začíná výpočtem normál z bodového mračna, které jsou následně použity k definici divergentního vektorového pole. Toto pole je pak interpretováno jako pravá strana Poissonovy rovnice. Řešením této rovnice získáme skalární funkci, jejíž izopovrchy jsou extrahovány jako výsledný mesh. Tento přístup je schopen interpolovat mezi vzorky dat a může vyplnit mezery v datech, což vede k vysoce detailním a hladkým povrchům. [18] [19]

Izopovrch: Izopovrch je povrch ve 3D prostoru, který odpovídá konstantní hodnotě skalární funkce. Používá se k vizualizaci úrovní různých fyzikálních veličin ve 3D, což je zásadní pro rekonstrukci hladkých a detailních povrchů z bodového mračna. [18] [19]

Skalární pole: Skalární pole je matematická funkce, která každému bodu v prostoru přiřazuje skalární hodnotu. Ve vizualizaci a modelování se používá pro reprezentaci rozložení nějakého jevu, jako je teplota nebo hustota, což je klíčové pro správné řešení Poissonovy rovnice při rekonstrukci povrchů. [18] [19]

Divergence vektorového pole: Divergence je míra toku (rozšiřování nebo kontrakce) vektorového pole z bodu. V kontextu Poissonovy rekonstrukce se používá k definici změn v orientaci normál, což je fundamentální pro správné modelování povrchu. [18]

Výhody

- Efektivní pro velké datové sady
- Dobře interpoluje hladké povrchy a složité topologie.

Nevýhody

- Vyžaduje dobře definované normály bodů.
- Může být citlivý na šum a odlehlé body.

Pseudokód

Následující pseudokód s komentáři prezentuje transformaci bodového mračna do 3D modelu ve formátu 3D mesh pomocí metody poisson surface reconstruction.

```
function PoissonReconstruction(pointCloud):
    ComputePointCloudNormals(pointCloud)
    # Vypočítá normály pro každý bod v bodovém mračnu.
    # Normály jsou potřebné pro definici orientace povrchu.
    InitializeSpatialTree(pointCloud)
    # Inicializuje prostorový strom pro efektivní vyhledávání
    # a organizaci bodů.
    # Tento krok zlepšuje výkon při řešení Poissonovy rovnice.
    SolvePoissonEquation()
    # Aplikuje Poissonovu rovnici na vypočítané normály a řeší ji
    # pro získání implicitní funkce povrchu.
    ExtractIsoSurface()
    # Extrakce izopovrchu z výsledného skalárního pole.
    # Izopovrch reprezentuje hledaný 3D mesh.
    return mesh
    # Vrací vytvořený 3D mesh z bodového mračna.
```

4.3.2 Marching Cubes

Popis

Marching Cubes je populární algoritmus pro vytváření 3D meshů z volumetrických dat.

Marching Cubes je algoritmus určený pro extrakci povrchu z 3D skalárních polí (často používaných v lékařských obrazových datech a simulacích). Algoritmus projde 3D mřížku rozdělenou na krychle (voxely). Pro každý voxel se určí, zda jeho rohy leží uvnitř nebo vně povrchu, což se určuje na základě srovnání hodnoty skalárního pole s prahem (threshold). Podle tohoto kritéria je z tabulky vyhledána odpovídající topologie trojúhelníků, která nejlépe vyjadřuje průnik povrchu s daným voxlem. Tento proces je opakován pro každý voxel, čímž se postupně sestavuje celý mesh. [20]

Voxel: Voxel, nebo volumetrický pixel, je základní jednotka pro reprezentaci 3D dat v pravidelné mřížce. V algoritmu Marching Cubes se pro každý voxel rozhoduje, jak bude reprezentován povrch přecházející skrz něj, což je klíčové pro extrakci přesného 3D modelu z volumetrických dat. [20]

Výhody

- Jednoduchý a poměrně přímočarý
- Snadno implementovatelný a velmi populární ve vizualizačních aplikacích

Nevýhody

- Může generovat velké množství dat
- Vyžaduje uniformní mřížkovou reprezentaci

Pseudokód

Následující pseudokód s komentáři popisuje transformaci bodového mračna do formátu 3D mesh pomocí algoritmu Marching Cubes.

```
function MarchingCubes(volume):
    for each cube in volume:
    # Iteruje přes každou kostku (voxel) ve volumetrických datech.
        if cube intersects surface:
        # Zjišťuje, zda kostka protíná teoretický povrch objektu.
            triangles += GenerateTriangles(cube)
            # Pro každou kostku, která protíná povrch,
            # generuje odpovídající trojúhelníky
            # podle tabulky konfigurací hraničních vrcholů.
    return triangles
    # Vrátí seznam trojúhelníků, který reprezentuje mesh objektu.
```

4.3.3 Delaunay Triangulation

Popis

Delaunayova triangulace v 3D je proces, při kterém je množina bodů ve 3D prostoru rozdělena na tetrahedry tak, že žádný bod z množiny není uvnitř obkružující koule žádného z tetrahedrů. Tento algoritmus optimalizuje úhel tetrahedrů, což vede k robustnější a rovnoměrnější triangulaci. Triangulace se často používá pro vytvoření meshů z neuspořádaných bodů, jelikož efektivně identifikuje sousední body a spojuje je do geometricky koherentních struktur. Extrahovaný mesh je poté tvořen vnějšími stěnami tetrahedrů, což reprezentuje povrch objektu. [21]

Tetrahedron: Tetrahedron je čtyřstěnný polyhedron, který je základním stavebním blokem pro 3D Delaunayovu triangulaci. Rozdělení prostoru na tetrahedry umožňuje efektivní a matematicky robustní reprezentaci 3D objemů, což je nezbytné pro vytváření meshů z neuspořádaných bodových mračen. [21]

Výhody

- Matematicky robustní.
- Dobře funguje pro data s dobrou distribucí bodů ve 3D prostoru.

Nevýhody

- Není vhodný pro velké datové sady.

- Může generovat nežádoucí spoje mezi odlehlými body.

Pseudokód

Následující pseudokód s komentáři popisuje transformaci bodového mračna do formátu 3D mesh pomocí algoritmu Delaunay Triangulation.

```
function DelaunayTriangulation(points):
    tetrahedra = ComputeDelaunayTetrahedra(points)
    # Vypočítá Delaunayovy tetrahedry z bodového mračna.
    # Tetrahedry jsou 3D obdoby trojúhelníků,
    # které optimalizují minimální úhel.
    mesh = ExtractSurface(tetrahedra)
    # Extrakce vnějšího povrchu tetrahedrů.
    # Tento krok vybírá pouze ty stěny tetrahedrů,
    # které tvoří vnější obal.
    return mesh
    # Vrátí vytvořený mesh, který je sestaven
    # z vnějších stěn tetrahedrů.
```

4.3.4 Ball Pivoting Algorithm (BPA)

Popis

BPA modeluje mesh pomocí kuličky o určitém poloměru.

BPA je relativně jednoduchý, ale efektivní algoritmus pro rekonstrukci povrchu z bodového mračna, který používá kuličku o definovaném poloměru jako základní nástroj pro vytváření trojúhelníků. Algoritmus začíná s jedním trojúhelníkem a postupně "otáčí"kuličku kolem hrany trojúhelníku dokud nenarazí na další bod, který je dostatečně blízko k uzavření dalšího trojúhelníku. Tento proces pokračuje, dokud nelze vytvořit další trojúhelníky. Velikost kuličky ovlivňuje schopnost algoritmu vyplnit mezery a zachytit detaily povrchu. Algoritmus je vhodný pro mračna s relativně rovnoměrnou distribucí bodů a může selhat v oblastech s nízkou hustotou bodů. [22]

Výhody

- Relativně rychlý.
- Generuje hladké povrchy, pokud jsou body rovnoměrně rozložené.

Nevýhody

- Výsledek závisí na velikosti kuličky.
- Může selhat v oblastech s malou hustotou bodů.

Pseudokód

Následující pseudokód s komentáři popisuje transformaci bodového mračna do formátu 3D mesh pomocí metody Ball Pivoting Algorithm.

```
function BallPivoting(pointCloud, radius):
    InitializeMesh()
    # Inicializace prázdného mesh modelu,
    # který bude postupně naplněn trojúhelníky.
    while not complete:
    # Opakuje cyklus, dokud nejsou zpracovány všechny
    # relevantní body bodového mračna.
        pivot = FindPivotPoint(pointCloud, mesh, radius)
        # Najde pivot bod,
        # kolem kterého se bude "otáčet" kulička o definovaném
        # poloměru a hledá body, které mohou tvořit trojúhelník
        # s již existujícími body meshu.
        if pivot:
        # Pokud je nalezen vhodný pivot bod.
            mesh += CreateTriangle(pivot)
            # Vytvoří nový trojúhelník
            # s využitím pivot bodu a přidá ho do meshu.
    return mesh
    # Vrátí finální mesh model sestavený z trojúhelníků.
```

4.4 Doplnění barev do 3D meshe z obrazových dat

Tato část se zaměřuje na metody doplnění barev do 3D meshů využívajících obrazová data. Postupy, které jsou zde diskutovány, umožňují transformovat 2D obrazové informace na 3D struktury, což je klíčové pro vytvoření realisticky vyhlížejících 3D modelů v aplikacích jako je digitální umění, virtuální realita a simulace fyzikálních procesů. [1]

4.4.1 3D Mesh a jeho struktura

3D Mesh je digitální struktura používaná k reprezentaci třírozměrných objektů. Skládá se z vrcholů, hran a stěn, které společně tvoří povrch objektu. Pro aplikace doplnění barev je důležité, aby mesh byl dostatečně detailní, aby mohl podporovat složitost a nuance získaných obrazových dat. [1]

4.4.2 Zpracování obrazu pro extrakci barevných dat

Extrakce barevných dat z obrazů zahrnuje analýzu obrazového obsahu k identifikaci a extrakci barevných vzorů, které mají být aplikovány na mesh. Tento proces může zahrnovat segmentaci obrazu, detekci hran a další techniky počítačového vidění. [1]

4.4.3 Techniky doplnění barev

Projekční mapování

Projekční mapování zahrnuje projekci obrazových dat přímo na povrch 3D meshe, což často vyžaduje pečlivé zarovnání mezi obrazem a geometrií meshe. [23]

UV mapování

UV mapování je proces, při kterém jsou 2D obrazové textury mapovány na 3D objekty pomocí UV souřadnic, což umožňuje přesné a efektivní aplikace textur bez deformací. [23]

Vertex coloring

Vertex coloring přiřazuje barvu přímo vrcholům meshu, což umožňuje jednoduché, ale efektivní barevné detaily bez nutnosti složitých textur. [23]

4.4.4 Algoritmy pro mapování barev

Ray casting pro spojení obrazu a meshu

Ray casting algoritmus využívá zpětné sledování paprsků od obrazové roviny k meshu, aby určil, které barvy by měly být aplikovány na specifické vrcholy. [24]

Interpolace barev mezi vertexy

Interpolace barev je technika používaná k vytvoření plynulého přechodu barev mezi vrcholy, což je zvláště užitečné při aplikaci složitých barevných gradientů na mesh. [1]

4.4.5 Výzvy a omezení

Řešení problémů s okluzí a osvětlením

Okluzní a osvětlovací problémy mohou komplikovat přesné mapování barev, zvláště v složitých scénách s mnoha překrývajícími se objekty.

Optimalizace výpočetní náročnosti

Algoritmy pro doplnění barev mohou být výpočetně náročné, což vyžaduje optimalizaci k dosažení reálného výkonu pro praxi.

5 KONVOLUČNÍ NEURONOVÉ SÍTĚ (CNN) A ZPRACOVÁNÍ OBRAZU

Konvoluční neuronové sítě (CNN) jsou hluboké učící modely, které mají zásadní význam zejména ve zpracování obrazu a video analýze. V této sekci zavedeme základní koncepty CNN a prozkoumáme jejich architekturu.

5.1 Hluboké učení

Hluboké učení je podmnožinou strojového učení, která využívá neuronové sítě s mnoha vrstvami (tzv. hluboké neuronové sítě) k modelování složitých vzorců ve velkých objemech dat. Hluboké učení se stalo základem pro mnoho moderních aplikací umělé inteligence, včetně zpracování přirozeného jazyka, rozpoznávání obrazu a autonomních vozidel. [3] [25]

5.2 Matematické principy CNN

Konvoluční neuronové sítě (CNN) využívají matematické operace konvoluce k automatickému učení vlastností z obrazových dat. Tyto operace umožňují síti efektivně zpracovávat obrazové data s přihlédnutím k lokálním vzorcům jako jsou hrany, tvary a textury. [3]

Konvoluce

Konvoluce je matematická operace, která kombinuje dvě funkce a produkuje třetí funkci. Ve smyslu CNN, konvoluce spočívá v aplikaci filtru (nebo jádra) na obraz, což vede k vytvoření tzv. mapy rysů. [3] Matematicky je to vyjádřeno v následující rovnici.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{5.1}$$

V této rovnici je I obrazová matice, K je jádro konvoluce a S je výsledná mapa rysů.

Aktivační funkce

Aktivační funkce jsou klíčové pro přidání nelinearity do neuronové sítě, což umožňuje modelovat složitější funkce. Běžně používaná aktivační funkce v CNN je ReLU (Rectified Linear Unit), která je definována v následující rovnici, kde x je vstup do aktivační funkce a $f(x)$ je výstupní hodnota funkce. Tato funkce přiřazuje hodnotu nula všem negativním vstupům a zanechává kladné hodnoty nezměněné. Tímto způsobem ReLU

přispívá k nelinearitě v neuronové síti a umožňuje modelovat komplexní vzory v datech. [27] [3]

$$f(x) = \max(0, x) \tag{5.2}$$

Pooling

Pooling je proces redukce rozměrů vstupních dat, což zjednodušuje informace v mapách rysů a zmenšuje počet parametrů a výpočtů v síti. Max pooling, což je běžná technika, vybírá maximum z bloku pixelů. [3] [28]

$$P(i, j) = \max_{k,l \in W_{i,j}} X(k, l) \tag{5.3}$$

kde $W_{i,j}$ je blok pixelů a X je vstupní obrazová data.

5.3 Architektura konvolučních neuronových sítí

Architektura konvoluční neuronové sítě je základ úspěchu a účinnosti modelů hlubokého učení v oblasti počítačového vidění, zpracování obrazu, rozpoznávání řeči a mnoha dalších úloh. Důležitost architektury spočívá v tom, že správná volba vrstev, počtu neuronů, konvolučních filtrů a dalších parametrů může dramaticky ovlivnit výkon a schopnost modelu generalizovat na nová data. [3] [28]

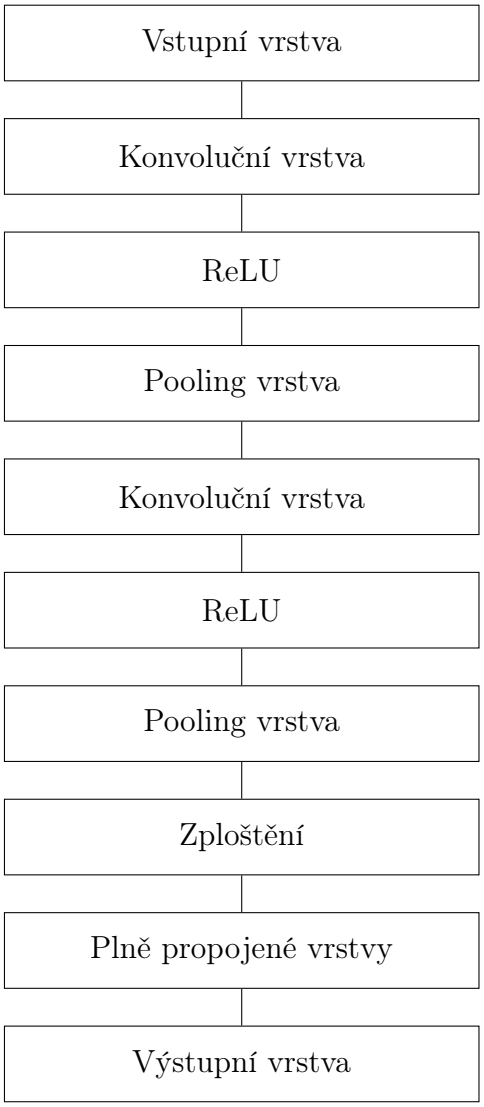
Správně navržená architektura CNN může efektivně extrahovat a hierarchicky reprezentovat různé úrovně abstrakce a funkční prvky v datech. To znamená, že síť dokáže postupně identifikovat jednoduché rysy jako hrany a textury a poté je kombinovat do složitějších struktur, jako jsou tvary a objekty. Tato schopnost hierarchického učení je klíčová pro úspěšné zpracování a porozumění vizuálním datům. [3] [25]

Dalším důležitým aspektem architektury CNN je schopnost se efektivně učit z dat. Moderní architektury CNN jsou často schopny se učit přímo ze surových dat, což eliminuje potřebu ručně navržených funkcí nebo extrakčních metod. To umožňuje modelům lépe se přizpůsobovat různorodým datovým sadám a dosahovat vynikajících výsledků bez potřeby ručního ladění. [3] [28]

Klíčové je pečlivě navrhnout a vybrat vhodnou architekturu v závislosti na konkrétní úloze a dostupných datech.

Ukázka schématu příkladu architektury Konvuluční neuronové sítě

Následující schéma popisuje zjednodušený příklad architektury konvuluční neuronové sítě. Tento příklad obsahuje základní vrstvy, které jsou v následujících sekcích popsané.



Obrázek 5.1 Příklad schémata konvoluční neuronové sítě (CNN) (vlastní tvorba)

Konvoluční vrstvy

Konvoluční vrstvy představují základní stavební blok konvolučních neuronových sítí. Tyto vrstvy aplikují konvoluční operace na vstupní data, což umožňuje sítím učit se vlastnosti z datových sad obrazů na různých úrovních abstrakce. Konvoluční operace vytváří mapy příznaků, které reprezentují detekované vzory nebo rysy v datech. Každý filtr (nebo jádro) v konvoluční vrstvě hledá specifické vzory vstupních dat pomocí konvoluční operace, což umožňuje sítím extrahovat různé úrovně vlastností, jako jsou hrany, textury a objekty. Formálně lze konvoluční operaci vyjádřit jako:

$$y[i,j] = \sum_{u,v} x[i-u, j-v] \cdot w[u,v] + b \tag{5.4}$$

kde x je vstupní mapa příznaků, w je váhová matice filtru, b je zkreslení (bias), a y je výstupní mapa příznaků. Tento proces umožňuje síti učit se rozpoznávat vzory na různých úrovních detailu a abstrakce. [25]

Pooling vrstvy

Pooling vrstvy jsou často používány k redukci rozměrovosti vstupních dat, což pomáhá snižovat počet parametrů a výpočetní složitost modelu. Tento proces snižuje citlivost modelu na malé posuny v datech a umožňuje sítím lépe generalizovat. Nejběžnější typ pooling vrstvy je max pooling, který vybírá maximální hodnotu v dané oblasti (například 2x2 okně) vstupních dat. Formálně lze max pooling vyjádřit jako:

$$y[i,j] = \max_{u,v} x[i+u, j+v] \tag{5.5}$$

kde x je vstupní mapa příznaků a y je výstupní mapa příznaků po aplikaci max pooling. [25]

Plně propojené vrstvy

Na konci architektury konvolučních neuronových sítí jsou plně propojené vrstvy, které slouží k zajištění klasifikace nebo regrese na základě naučených vlastností. Tyto vrstvy spojují všechny příznaky z předchozích vrstev a provádějí lineární transformaci na vstupních datech. Výsledný vektor příznaků je následně zpracován pomocí aktivační funkce, jako je například ReLU, a výstupní vrstva provádí konečnou klasifikaci nebo regresi. Formálně lze plně propojenou vrstvu vyjádřit jako:

$$y = \sigma(Wx + b) \tag{5.6}$$

kde x je vstupní vektor příznaků, W je váhová matice, b je bias a σ je aktivační funkce.[3]

5.4 Trénování konvoluční neuronové sítě

Trénování konvoluční neuronové sítě (CNN) je proces, při kterém se model postupně učí extrahovat relevantní rysy z trénovacích dat a naučit se přiřazovat správné třídy nebo predikovat správné výstupy. Tento proces se skládá z několika klíčových kroků, které zahrnují přípravu dat, definici architektury modelu, volbu ztrátové funkce a optimalizačního algoritmu a samotné trénování a vyhodnocování modelu. [28]

Prvním krokem při trénování CNN je příprava trénovacích dat. To zahrnuje načtení a předzpracování dat, jako je normalizace, změna velikosti obrázků nebo aplikace augmentačních technik pro rozšíření trénovacího souboru. Dále je nutné data rozdělit na trénovací, validační a testovací sady, aby bylo možné efektivně vyhodnocovat výkon modelu. [25]

Poté následuje definice architektury modelu. Tento krok zahrnuje vytvoření instancí konvolučních, pooling a plně propojených vrstev, stejně jako specifikaci hyperparametrů jako je počet filtrů, velikost kernelů a velikost vrstev. Správná architektura je klíčová pro úspěšné trénování modelu a dosažení požadovaného výsledku. [25]

Poté je model zkompilován s pomocí určené ztrátové funkce a optimalizačního algoritmu. Ztrátová funkce měří rozdíl mezi skutečnými a predikovanými hodnotami a slouží jako ukazatel chyby, kterou model musí minimalizovat během trénování. Optimalizační algoritmus pak upravuje váhy modelu tak, aby minimalizoval tuto ztrátu. [25] [3]

Ztrátová funkce měří, jak dobře model predikuje očekávané výstupy. Běžně se používá křížová entropie pro klasifikační úkoly. Křížová entropie měří rozdíl mezi pravdivým rozložením tříd a predikovaným rozložením tříd modelu. Formálně lze křížovou entropii vyjádřit jako:

$$H(y, \hat{y}) = -\sum_{i} y_i \log(\hat{y}_i) \tag{5.7}$$

kde y je pravdivé rozložení tříd a $\hat{y}$ je predikované rozložení tříd modelu. Proces učení konvolučních neuronových sítí zahrnuje optimalizaci parametrů sítě pomocí algoritmů jako je gradientní sestup. Backpropagation je algoritmus používaný k efektivnímu výpočtu gradientů a aktualizaci parametrů sítě. Během trénování je chyba modelu zpětně propagována skrze síť, a gradienty jsou vypočítány pro každý parametr pomocí řetězového pravidla derivací. Gradientní sestup poté aktualizuje váhy sítě tak, aby minimalizoval chybu predikce modelu. Tento proces se opakuje iterativně pro každou dávku dat až do dosažení optimálních váh pro daný úkol učení. [3]

Nakonec probíhá samotné trénování modelu na trénovacích datech. Během tohoto procesu jsou trénovací data postupně předávána modelu, který upravuje své váhy tak, aby minimalizoval chybu na trénovacích datech. Trénování probíhá po určený počet epoch, přičemž v každé epošě je model aktualizován na základě gradientu ztrátové funkce. Po dokončení trénování je model vyhodnocen na validačních a testovacích datech, aby bylo možné získat objektivní měření jeho výkonu. [28] [29]

5.5 Příklad zjednodušené implementace konvoluční neuronové sítě

Tato sekce poskytuje příklad zjednodušenné implementace konvoluční neuronové sítě (CNN) pomocí populární knihovny TensorFlow. Příklady kódu demonstrují postup vytvoření základního modelu CNN a jeho kompilaci pomocí TensorFlow. Implementace je zjednodušená a zaměřuje se na základní prvky architektury CNN, jako jsou konvoluční vrstvy, vrstvy max pooling, plně propojené vrstvy a výstupní vrstva.

V ukázce kódu níže je vytvořen sekvenční model s několika vrstvami, které tvoří základní architekturu CNN. Začínáme konvolučními vrstvami, kde se provádí konvoluce na vstupních datech za účelem extrakce rysů. Každá konvoluční vrstva je následována aktivační funkcí ReLU, která přidává nelinearitu do modelu. Poté jsou aplikovány vrstvy max pooling, které redukují velikost obrazu a zjednodušují výpočetní náročnost. Po konvolučních a pooling vrstvách následují plně propojené vrstvy, které provádějí klasifikaci na základě extrahovaných rysů. Na konci je přidána výstupní vrstva s aktivací softmax, která generuje pravděpodobnostní rozdělení pro klasifikaci do různých tříd.

Po vytvoření modelu je provedena jeho kompilace pomocí optimalizačního algoritmu a ztrátové funkce. Dále je model nastaven tak, aby sledoval metriku 'accuracy' (přesnost) během trénování a vyhodnocování. Tímto způsobem je model připraven k trénování a evaluaci na konkrétních datech.

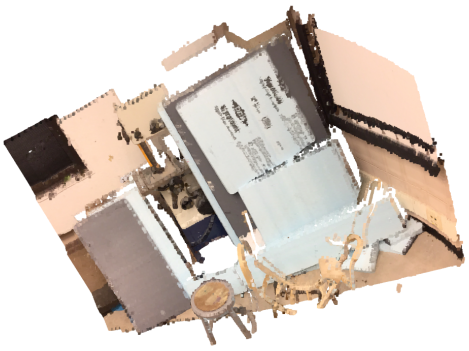
```
import tensorflow as tf
# Vytvoření sekvenčního modelu s následujícími vrstvami:
model = tf.keras.models.Sequential([
    # Konvoluční vrstva: 32 filtrů, velikost kernelu 3x3, aktivace ReLU
    # 'input_shape' určuje rozměry vstupních dat (výška, šířka, kanály)
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
    input_shape=(28, 28, 1)),
    # Max pooling vrstva: redukuje velikost obrazu o
    faktor 2 pomocí 2x2 poolingu
    tf.keras.layers.MaxPooling2D((2, 2)),
    # Druhá konvoluční vrstva: 64 filtrů, velikost
    kernelu 3x3, aktivace ReLU
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    # Druhá max pooling vrstva: dále redukuje velikost obrazu
    tf.keras.layers.MaxPooling2D((2, 2)),
    # Třetí konvoluční vrstva: 64 filtrů, velikost
    kernelu 3x3, aktivace ReLU
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    # Zploštění dat před přechodem do plně propojených vrstev
    tf.keras.layers.Flatten(),
    # Plně propojená vrstva: 64 neurony, aktivace ReLU
    tf.keras.layers.Dense(64, activation='relu'),
    # Výstupní vrstva: 10 neurony pro 10 tříd,
    aktivace softmax pro klasifikaci
    tf.keras.layers.Dense(10, activation='softmax')
])
# Kompilace modelu s optimalizátorem 'adam',
ztrátovou funkcí 'sparse_categorical_crossentropy'
# a sledováním metriky 'accuracy' (přesnost)
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
# Zde by se přidalo trénování modelu a evaluace
```

II. PRAKTICKÁ ČÁST

6 POPIS PROJEKTU A ÚVOD K PRAKTICKÉ ČÁSTI

V praktické části mé diplomové práce se zabývám průzkumem možností transformace dvojice obrázků na 3D model ve formátu 3D mesh za pomocí umělé inteligence a hlubokého učení. Konkrétně využívám konvoluční neuronovou síť, která je součástí algoritmu pro 3D rekonstrukci metodou stereovize. Tato síť slouží k feature matchingu mezi dvojicí obrázků s cílem vytvořit disparity mapu, která je klíčová pro následnou rekonstrukci 3D modelu. Následují dva obrázky, první reprezentuje jeden ze vstupů a druhý obrázek poté ukazuje výsledný 3D model (výstup z programu).

Obrázek 6.1 Vstup (levý obrázek) [31]



Obrázek 6.2 Výstup (výsledný 3D model)
(vlastní tvorba)

Praktická část této diplomové práce je strukturovaná do tří klíčových sekcí. První sekce se soustředí na dokumentaci a podrobný popis zdrojového kódu vytvořeného programu. Zabývám se zde technickými aspekty funkčnosti programu a detailním přehledem využitých knihoven. Strukturu zdrojového kódu dále rozkládám do jednotlivých modulů, které plní rozdílné úkoly, jako je například vytváření modelu konvuluční neuronové sítě (CNN), 3D rekonstrukce či srovnání výsledků. Sekce navíc obsahuje vybrané ukázky kódu, které ilustrují klíčové techniky a postupy. Speciální pozornost je věnována segmentu kódu, který se týká využití umělé inteligence. Podrobně popisuji architekturu implementované konvuluční neuronové sítě, přístupy k trénování, včetně

popisu použitého datasetu, a způsob, jakým je natrénovaný model integrován do celkového programu.

Druhá hlavní část práce je věnována vyhodnocení dosažených výsledků projektu. Provádím zde jak statistické, tak vizuální hodnocení a srovnání výkonu mého přístupu s tradičními a osvědčenými metodami.

Ve třetí a závěrečné části se zaměřuji na budoucí plány pro rozvoj projektu a možnosti jeho aplikace. Detailně popisuji potenciál využití 3D rekonstrukcí s využitím konvulučních neuronových sítí. Tato sekce také reflektuje stávající nedostatky a slabá místa projektu a navrhuje možná řešení pro jejich překonání.

7 DOKUMENTACE A POPIS ZDROJOVÉHO KÓDU A POUŽITÝCH KNIHOVEN

V této kapitole popisuji jednotlivé moduly mého zdrojového kódu. Kód je popisován ve stejném pořadí, v jakém běží. Veškeré ukázky kódu jsou má vlastní tvorba (kromě použitých knihoven). První část této sekce se věnuje popisu jednotlivých knihoven, které používám.

7.1 Výpis použitých knihoven

Knihovna PyTorch

PyTorch je rozsáhlá knihovna pro strojové učení, která poskytuje rozsáhlé možnosti pro práci s multidimenzionálními tenzory a provádění matematických operací. Tenzory v PyTorch fungují jako základní stavební kameny, sloužící nejen k uložení dat, ale také k zachycení gradientů během procesu trénování modelů. Díky podpoře běhu na CPU i GPU umožňuje PyTorch efektivně zpracovávat výpočty, což je klíčové pro rychlé trénování složitých modelů. [33]

Modul torch.nn je speciálně navržen pro definování vysokoúrovňových abstrakcí pro neuronové sítě, včetně různých typů vrstev jako jsou konvoluční vrstvy, pooling vrstvy či vrstvy pro normalizaci dávky. Tyto nástroje usnadňují konstrukci a experimentaci s novými architekturami neuronových sítí, a dále poskytují implementace běžně používaných aktivačních a ztrátových funkcí. [33]

Pro nízkoúrovňové operace je zde modul torch.nn.functional, který je zásadní pro efektivní integraci s autograd mechanismem PyTorch. Nabízí širokou paletu funkcí jako jsou konvoluce, různé aktivační funkce a ztrátové funkce, které jsou optimální pro operace vyžadující minimální udržování vnitřního stavu, například při implementaci forward pass v neuronových sítích. [33]

Tato knihovna a její moduly jsou v této práci použité v definici architektury modelu konvoluční neuronové sítě a v algorytmu jeho účení. Byla velmi přínosná, protože jsem s její pomocí mohl proces učení přenést na GPU, což vše značně urychlilo. Dále mi tato knihovna ušetřila spoustu času díky kvalitní dokumentaci a dobrým přístupem k abstrakcím.

Knihovna NumPy

Tato knihovna je základem pro vědecké výpočty v Pythonu. Poskytuje podporu pro velké, multidimenzionální pole a matice, spolu s rozsáhlou knihovnou matematických funkcí pro práci s těmito datovými strukturami. NumPy je nezbytný pro efektivní manipulaci s daty a matematické operace, které jsou základem všech ostatních používaných knihoven pro zpracování obrazu a strojové učení. [34]

Jedná se o velmi obecnou knihovnu, která je v mém projektu použitá na mnoha místech pro mnoho různých účelů.

Knihovna OpenCV (cv2)

OpenCV je vyspělá, ověřená a velmi rozšířená knihovna pro počítačové vidění, která umožňuje real-time zpracování obrazu. Zahrnuje funkce pro základní i pokročilé operace, jako jsou transformace obrazu, detekce objektů a sledování pohybu. [35]

Knihovnu OpenCV v projektu používám na práci s obrázky a další funkce spojené s počítačovým viděním.

Knihovna Matplotlib

Matplotlib je robustní knihovna pro vizualizaci v Pythonu, která podporuje širokou škálu grafů a diagramů. Její schopnost integrace s NumPy umožňuje komplexní grafické zobrazení datových analýz. [36]

V mém projektu je tato knihovna používaná na vizualizaci průběžných výsledků a jejich ukládání.

Knihovna Open3D

Open3D je knihovna zaměřená na práci s 3D daty, která poskytuje nástroje pro 3D rekonstrukci, zpracování bodových mraků, a vizualizaci 3D modelů. Tato knihovna obsahuje funkce jako jsou registrace bodových mračen, meshing a výpočet normál. [37]

V mém projektu je knihovna Open3D použita na práci s bodovými mračny a 3D Mesh modely. Také je velmi pomocná při vizualizaci výsledků.

Pillow (PIL)

Pillow je knihovna pro manipulaci s obrazem, která podporuje otevření, změnu a ukládání různých formátů obrázků. Tato práce ji využívá pro čtení, zpracování a ukládání obrazových dat, což je nezbytné pro všechny operace zahrnující obrazový vstup a výstup. [38]

Tuto knihovnu požívám například k úpravě vstupních obrázků.

Scikit-learn

SciPy je další základní knihovna pro vědecké výpočty v Pythonu, která je postavena na NumPy a integruje pokročilé matematické funkce a algoritmy. Tato knihovna je nezbytná pro provádění sofistikovaných matematických operací, jako jsou integrace, optimalizace, interpolace, vlastní hodnoty, Fourierova transformace a další. SciPy je

užitečná pro výkonově náročné výpočty a analýzy, což zlepšuje efektivitu a přesnost vědeckých výpočtů, které jsou zásadní pro zpracování a analýzu dat. [40]

S knihovnou SciPy jsem experimentoval nad možnostmi odhadnutí parametrů nutných k 3D rekonstrukci.

Numba

Numba je vysokovýkonná, open source knihovna, která umožňuje kompilaci Python kódu do rychlého strojového kódu s minimálním úsilím na straně programátora. Použitím dekorátorů, jako je *@jit* , Numba dramaticky zrychluje výpočty, které by jinak byly omezeny běžnými výkonovými omezeními interpretovaného Pythonu. Tato knihovna je zvláště užitečná v oblastech, kde jsou potřeba rychlé matematické a vědecké výpočty, jako je analýza dat, simulace a strojové učení. Numba umožňuje uživatelům jednoduše definovat funkce, které mohou být kompilovány (Just-In-Time) pro výkon na CPU nebo GPU, což zvyšuje flexibilitu a výkon aplikací. [41]

7.2 Popis a dokumentace zdrojového kódu

V této části práce se věnuji popisu jednotlivých modulů mého programu. Následující ukázka kódu představuje funkci main v souboru main.py, která iniciuje spuštění všech ostatních komponent programu.

```
from compare_module.compare import compare
from model_module.model_training import start_training
from reconstruction_module.reconstruct_and_vizualize_mesh import (
    reconstruct_and_vizualize_mesh,
)
def main():
    start_training()
    # compare("trained_models/cnn_disparity_generator_model_epoch_20.pth")
    # reconstruct_and_vizualize_mesh(
    #     "data/real-data/pendulum2/im0.png",
    #     "data/real-data/pendulum2/im1.png",
    #     "data/real-data/pendulum2/calib.txt",
    #     "trained_models/cnn_disparity_generator_model_epoch_26.pth",
    #     "model_output_disparity.pfm",
    #     None,
    # )
if __name__ == "__main__":
    main()
```

Tento výpis kódu ukazuje strukturu hlavní funkce *main* v souboru *main.py*, která slouží jako centrální spouštěcí bod pro všechny klíčové komponenty programu. Funkce zahrnuje volání pro zahájení trénování modelu, porovnání přístupů a 3D rekonstrukce a vizualizace výsledků. Uživatel může ovládat chování programu upravením argumentů funkcí a aktivací nebo deaktivací specifických částí kódu prostřednictvím komentování nebo odkomentování příslušných řádků.

7.2.1 Modul 3D rekonstrukce a vizualizace

Vstupní bod komponenty, která se věnuje nejdůležitější části projektu, tedy samotné rekonstrukci 3D modelu ze vstupních dat je funkce *reconstruct_and_vizualize_mesh*, jejíž použití můžeme vidět v předchozí ukázce kódu (funkce *main*). Funkce *reconstruct_and_vizualize_mesh* přijímá několik specifikovaných argumentů, které definují její operace. První dva argumenty specifikují cesty ke stereo obrázkům, které slouží jako vstupní data pro rekonstrukci. Třetí argument určuje umístění textového souboru obsahujícího kalibrační parametry příslušných kamer. Dalším argumentem je cesta k uloženému, již natrénovanému modelu, který je použit pro generování disparity mapy. Předposlední argument specifikuje cílový název souboru, kam bude disparity mapa uložena. Poslední argument je nepovinný a označuje cestu k existující disparity mapě. Pokud je tento argument poskytnut, program použije tuto mapu pro 3D rekonstrukci namísto generování nové mapy.

Následuje ukázka ilustrace definice funkce *reconstruct_and_vizualize_mesh*, která poskytuje přehled o struktuře a funkcionalitě této klíčové funkcionality. Představený kód slouží k objasnění jednotlivých částí funkce. Jedná se o zkrácení reálné implementace s komentáři.

```
def reconstruct_and_viz_mesh(
    left_img,  # Cesta k levému obrázku
    right_img, # Cesta k pravému obrázku
    params_path, # Cesta k souboru s parametry
    model_path,  # Cesta k modelu pro predikci disparity
    disp_out_path, # Cesta pro uložení disparity
    disp_path=None # Volitelná cesta k existující disparity
):
    # Načtení parametrů kamery
    fl, cx, cy, bl, vmin, vmax, w, h, _ = read_params(params_path)
    # Predikce disparity, pokud není cesta k existující
    if not disp_path:
        predict_disp(model_path, left_img, right_img, disp_out_path)
    # Načtení disparity mapy
    disparity = (
        readPFM(disp_path, True, vmax, vmin) if disp_path
        else readPFM(disp_out_path, False, vmax, vmin) if disp_out_path
        else None
    )
    # Vizualizace disparity mapy
    test_visualize_disp_map(disparity)
    # Výpočet hloubky z disparity
    depth = calc_depth(disparity, fl, bl, vmin, vmax)
    # Výpočet bodového mraku
    X_valid, Y_valid, Z_valid = calc_point_cloud(
        depth, disparity, fl, vmin, vmax, cx, cy, w, h
    )
    # Vizualizace bodového mraku
    visualize_point_cloud(X_valid, Y_valid, Z_valid)
    # Načtení barev z obrázku
    colors = read_img_colors(
        left_img, X_valid, Y_valid, Z_valid, fl, cx, cy, (w, h)
```

```
    )
    # Generování 3D mřížky
    pcd = generate_3D_mesh(X_valid, Y_valid, Z_valid, colors)
    # Vizualizace 3D mřížky
    visualize_3D_mesh(pcd)
```

Následuje popis jednotlivých metod, které jsou ve funkci *reconstruct_and_vizualize_mesh* použité.

Čtení parametrů

Extrakce parametrů je základním krokem, který získává nezbytné kalibrační údaje pro převod disparity na hloubku. Parametry zahrnují ohniskovou vzdálenost, pozici středu obrazu, rozměry obrazu a baseline. Tyto hodnoty jsou získány z kalibračního souboru a jsou kritické pro následné výpočty. Následuje ilustrace kódu spolu s komentáři, která popisuje čtení těchto parametrů z textových souborů. Tyto soubory byly součástí dat z datasetu Middlebury.

```
def read_parameters(filepath):
    # Inicializace slovníku pro uchování parametrů
    params = {}
    # Čtení souboru a extrakce klíč-hodnota párů
    with open(filepath, "r") as file:
        for line in file:
            if "=" in line:
                key, value = line.strip().split("=")
                params[key.strip()] = value.strip()
    # Extrahování parametrů kamery cam0 (fokální délka, cx, cy)
    cam0 = params.get("cam0")
    if cam0:
        # Odstranění nepotřebných znaků a rozdělení hodnot
        matrix_values = cam0.replace("[", "")
                            .replace("]", "")
                            .replace(";", "")
                            .split()
        focal_length = float(matrix_values[0])
        cx = float(matrix_values[2])
        cy = float(matrix_values[5])
    else:
        focal_length = cx = cy = None
    # Další specifické parametry pro 3D rekonstrukci
    baseline = float(params.get("baseline", 0))
    vmin = int(params.get("vmin", 0))
    vmax = int(params.get("vmax", 0))
    width = int(params.get("width", 0))
    height = int(params.get("height", 0))
    ndisp = int(params.get("ndisp", 0))
    # Vrácení všech potřebných parametrů
    return (focal_length,
    cx, cy, baseline,
    vmin, vmax, width,
    height, ndisp)
```

Parametry extrahované touto funkcí jsou nezbytné pro realizaci následujících fází procesu 3D rekonstrukce, které zahrnují čtení disparity map, výpočet hloubky, generování bodového mraku a další související operace.

Ve své práci jsem také zkoumal možnosti 3D rekonstrukce bez předem známých nutných parametrů kamery. Tyto nutné parametry jsou baseline, která popisuje vzdálenost od obou kamer a focal length (ohnisková vzdálenost). Toto jsou minimální nutné parametry pro úspěšnou 3D rekonstrukci ze dvou stereo obrázků. V praktickém projektu jsem se tedy snažil odhadnout tyto parametry jen na základě dvou vstupních obrázků. Při mém výzkumu jsem došel k pochopení, že přesnost vygenerování těchto parametrů jen na základě dvou stereo obrázků je velmi komplikované a nespolehlivé. Lepších výsledků jsem dosáhl při odhadování pouze ohniskové vzdálenosti s předem známou vzdáleností kamer, ale i tak jsem tuto logiku nepoužil a zaměřil se na dataset, který obsahuje přesné kalibrační parametry. Následuje ukázka kódu, který odhadoval ohniskovou vzdálenost na základě disparity mapy a baseline.

```
from numba import jit
import numpy as np
from scipy.optimize import differential_evolution
@jit(nopython=True)
def compute_depths(f, baseline, disparity_map):
    # Zamezení dělení nulou
    epsilon = 1e-6
    # Výpočet hloubek z disperzity s ohniskovou vzdáleností a baseline
    depths = (f * baseline) / (disparity_map + epsilon)
    return depths


@jit(nopython=True)
def compute_depth_variance(depths, disparity_map):
    valid_depths = []
    count = 0
    # Sběr validních hloubkových hodnot
    for i in range(depths.shape[0]):
        for j in range(depths.shape[1]):
            if disparity_map[i, j] > 0:
                valid_depths.append(depths[i, j])
                count += 1
    # Převod seznamu na numpy pole pro výpočet variance
    valid_depths_array = np.array(valid_depths)
    depth_variance = np.var(valid_depths_array)
    return depth_variance


@jit(nopython=True)
def objective_function(x, disparity_map, baseline):
    f = x[0]
    # Výpočet hloubek pro danou ohniskovou vzdálenost
    depths = compute_depths(f, baseline, disparity_map)
    # Výpočet variance hloubek pro minimalizaci
    depth_variance = compute_depth_variance(depths, disparity_map)
    return depth_variance


def estimate_focal_length(disparity_map, baseline):
```

```
    # Rozsah možných hodnot ohniskové vzdálenosti
    bounds = [(500, 2000)]
    # Optimalizace pro nalezení ohniskové vzdálenosti
    # minimalizující varianci hloubek
    result = differential_evolution(
        objective_function, bounds, args=(disparity_map, baseline),
        strategy='best1bin', maxiter=100, popsize=15, tol=0.01,
        mutation=(0.5, 1.5), recombination=0.7
    )
    return result.x[0]
```

Tento kód implementuje sérii funkcí určených k odhadu ohniskové vzdálenosti kamery z disparity mapy a základní vzdálenosti (baseline) pomocí knihoven Numba, NumPy a SciPy. Funkce compute_depths vypočítává hloubkovou mapu z disparity, používající geometrický vztah mezi disparitou, ohniskovou vzdáleností a baseline. Funkce, která se jmenuje *compute_depth_variance* následně vypočítává varianci těchto hloubek, zaměřující se pouze na validní hodnoty s kladnou disparitou. Objektivní funkce objective_function pak používá tuto varianci jako metriku pro optimalizaci, a hlavní funkce estimate_focal_length aplikuje metodu diferenciální evoluce pro nalezení hodnoty ohniskové vzdálenosti, která minimalizuje tuto varianci. Kombinace těchto technik a optimalizačních metod poskytuje efektivní nástroj pro přesný odhad ohniskové vzdálenosti, klíčový pro úspěšnou 3D rekonstrukci.

Generování disparity mapy

Ve funkci reconstruct_and_viz_mesh je po extrakci parametrů následující krok generace disparity mapy, který představuje klíčovou fázi celého procesu 3D rekonstrukce. Kvalita výsledků z tohoto kroku má zásadní vliv na celkovou kvalitu rekonstrukce, jelikož hodnoty disparity přímo určují vypočítanou hloubku objektů ve scéně. V závislosti na argumentech funkce reconstruct_and_viz_mesh dochází k rozdílnému přístupu ke generaci disparity mapy. V některých případech se použije již dříve vygenerovaná nebo existující disparity mapa, zatímco v jiných situacích se vytváří nová mapa disparity.

Pokud hodnota argumentu disp_path není None, program načte data z již existující disparity mapy. Tato mapa může být buď „ground truth“ disparity mapa získaná z datové sady, nebo mapa disparity, kterou program vygeneroval při předchozím spuštění. V rámci mého softwaru jsou disparity reprezentovány ve formátu PFM (Portable Float Map), který je preferován pro ukládání obrazových dat s vysokou přesností, jako jsou hloubkové mapy. Obrázky v tomto formátu jsou vertikálně převráceny a obsahují relevantní metadata. Kdykoliv je v programu disparity mapa čtena nebo ukládána, používá se tento formát. Současně se v programu dbá na udržení konzistence hodnot disparity, což zahrnuje její normalizaci na rozsah 0 - 1 a sjednocení rozměrů šířky a výšky disparitních map, aby byly kompatibilní s dalšími procesy a analýzami.

V opačném případě program generuje a ukládá novou disparity mapu na základě vstupních stereo obrázků. Tato logika je zapouzdřená ve funkci predict_disparity_from_model, která je zobrazená v následují ukázce kódu.

```
def predict_disparity_from_model(
    model_path,  # Cesta k souboru s natrénovaným modelem
    left_img_path,  # Cesta k levému obrazu ve stereo páru
    right_img_path,  # Cesta k pravému obrazu ve stereo páru
    output_path  # Cesta pro uložení vygenerované disparity mapy
):
    # Načtení a inicializace modelu pro predikci
    model = EnhancedCustomCNN()
    model.load_state_dict(torch.load(model_path))
    model.eval()  # Přepnutí modelu do evaluačního režimu
    # Načtení a předzpracování vstupních obrazů
    left_img = load_and_preprocess_image(left_img_path)
    right_img = load_and_preprocess_image(right_img_path)
    # Přidání batch dimenze
    left_img = left_img.unsqueeze(0)
    right_img = right_img.unsqueeze(0)
    # Přiřazení obrázků k zařízení, na kterém je model spuštěn
    device = next(model.parameters()).device
    left_img = left_img.to(device)
    right_img = right_img.to(device)
    # Predikce disparity mapy bez vytváření gradientů
    with torch.no_grad():
        output = model(left_img, right_img).squeeze(0).squeeze(0)
    # Konverze výstupu z tensoru na numpy pole
    output_np = output.cpu().numpy()
    # Vizualizace výsledné disparity mapy
    plt.figure()
    plt.imshow(output_np, cmap="hot")
    plt.colorbar()
    plt.title("Disparity Map")
    plt.show()
    # Kontrola statistik před uložením mapy do souboru PFM
    print("Before saving to PFM:")
    analyze_disparity(output_np)
    # Uložení disparity mapy ve formátu PFM
    write_pfm(output_path, output_np)
    # Načtení a kontrola uložené disparity mapy
    loaded_map = readPFM(output_path)
    print("After reading PFM:")
    analyze_disparity(loaded_map)
```

Funkce predict_disparity_from_model je navržena k predikci disparity mapy z dvojice stereo obrazů za pomoci natrénovaného modelu neuronové sítě. Tato funkce začíná načítáním modelu ze souboru specifikovaného argumentem model_path, přičemž model je následně přepnut do evaluačního režimu. Tento režim deaktivuje tréninkové funkce, jako jsou dropout a batch normalization, aby se zajistila konzistence výstupů během inferenční fáze. V dalším kroku jsou z cest left_img_path a right_img_path načteny a předzpracovány levý a pravý obraz. Předzpracování obnáší normalizaci a další úpravy, což zajišťuje kompatibilitu obrazů s vstupem modelu.

Oba obrazy jsou poté transformovány na tenzory a opatřeny batch dimenzí, což umožňuje modelu zpracovat data v dávkách. Tyto tenzory jsou dále přesunuty na zařízení (GPU nebo CPU), na kterém je model spuštěn, aby se optimalizoval výpočetní výkon. S využitím kontextového manažera torch.no_grad(), který zabraňuje tvorbě gradientů během inferenčního procesu, je vygenerována disparity mapa. Tato mapa je následně konvertována z tensoru PyTorch do numpy pole pro usnadnění další analýzy a vizualizace.

Pro vizualizaci disparity mapy je použita knihovna matplotlib, která mapu zobrazí jako teplotní spektrum. Toto zobrazení pomáhá lépe vizuálně posoudit kvalitu a charakteristiky výsledků. Před uložením mapy do souboru ve formátu PFM, jsou prováděny statistické analýzy disparity, aby byla ověřena její kvalita. Po uložení je disparity mapa znovu načtena a podrobena reanalýze, což potvrzuje úspěšnost uložení a integritu dat.

Výpočet hloubky

Po získání mapy disparity se 3D rekonstrukce posouvá do fáze výpočtu hloubky. V teoretické části diplomové práce jsem při popisu metody 3D rekonstrukce pomocí stereovize došel k následující rovnici, která popisuje výpočet hloubky daného bodu. V této rovnici se rozdíl poloh mezi dvěma korespondujícími body na dvou obrázcích vyjadřuje jako jmenovatel.

$$D = \frac{B \cdot f_1 \cdot f_2}{f_1 \cdot \tan(\theta_2) + f_2 \cdot \tan(\theta_1)} \tag{7.1}$$

Této hodnotě se říká disparita. Pro výpočet hloubky v mém progamu zjištuji disparitu pomocí konvoluční neuronové sítě. To je hlavní rozdíl mezi tradičním přístupem a metodou, kterou pro 3D rekonstrukci z dvou stereo obrázků do formátu 3D mesh používám ve svém programu. Umělá inteligence tedy v mém případě neřeší jen feature matching, ale i samotné generování disparity mapy. Výpočet hloubky v mém případě tedy popisuje následující funkce v pythonu, kde hodnota disparity je výstup z použití vytrénovaného modelu konvoluční neuronové sítě.

```
import numpy as np
def calculate_depth(disparity_normalized, focal_length, baseline, vmin, vmax):
    # disparity_normalized: Normalizovaná hodnota disparity (0-1)
    # focal_length: Ohnisková vzdálenost kamery, určuje šíři zorného pole
    # baseline: Vzdálenost mezi optickými středy dvou kamer ve stereo páru
    # vmin: Minimální možná hodnota skutečné disparity
    # vmax: Maximální možná hodnota skutečné disparity
    # Přepočet normalizované disparity na skutečnou hodnotu
    disparity_actual = vmin + disparity_normalized * (vmax - vmin)
    # Výpočet hloubky z aktuální disparity, zamezení dělení nulou
    depth = focal_length * baseline / np.maximum(disparity_actual, 1e-6)
```

```
    # Výpis průměrných hodnot hloubky a disparity pro kontrolu
    print("Depth Average:", depth.mean(),
          "Disparity Average:", disparity_actual.mean())
    return depth
```

Funkce calculate_depth slouží k výpočtu hloubky z normalizovaných hodnot disparity ve stereo obrazových datech. Funkce přijímá pět argumentů: disparity_normalized, které jsou normalizované hodnoty disparity mezi 0 a 1, focal_length, což je ohnisková vzdálenost kamery určující šíři zorného pole, baseline, což je vzdálenost mezi optickými středy dvou kamer ve stereo páru, vmin a vmax, což jsou minimální a maximální možné hodnoty skutečné disparity. Tyto hodnoty disparity jsou použity pro přepočet normalizované disparity na skutečnou hodnotu. Výpočet hloubky je pak realizován jako součin ohniskové vzdálenosti a baseline dělený hodnotou skutečné disparity, přičemž se používá funkce np.maximum pro zamezení dělení nulou. Funkce poskytuje průměrné hodnoty hloubky a skutečné disparity jako kontrolní metriku pro ověření kvality a konzistence výsledků. Tato funkce je klíčová pro aplikace, kde je třeba přesně rekonstruovat 3D prostor z obrazových dat.

Generování Bodového Mraku

Po úspěšném výpočtu hloubkové mapy pokračuje proces 3D rekonstrukce generováním bodového mračna, neboli 3D cloudu. Na základě získané hloubkové mapy jsou pro každý pixel obrázku vypočítány 3D souřadnice. V tomto projektu navíc pixely obarvuji podle jejich hloubkové hodnoty, což umožňuje vizuálně odlišit různé úrovně hloubky v rekonstruované scéně. Následuje ukázka kódu, který bodové mračno generuje.

```
import numpy as np
def calculate_point_cloud(
    depth,  # Hloubková mapa získaná z disparity
    disparity_normalized,  # Normalizovaná disparity mapa
    focal_length,  # Ohnisková vzdálenost kamery
    vmin,  # Minimální možná hodnota skutečné disparity
    vmax,  # Maximální možná hodnota skutečné disparity
    original_cx,  # Původní x-ová souřadnice středu kamery
    original_cy,  # Původní y-ová souřadnice středu kamery
    original_width,  # Původní šířka obrazu
    original_height,  # Původní výška obrazu
    resized_width=960,  # Šířka obrazu po změně rozměru
    resized_height=540  # Výška obrazu po změně rozměru
):
    # Faktory pro změnu měřítka ohniskové vzdálenosti a středu obrazu
    scale_x = resized_width / original_width
    scale_y = resized_height / original_height
    # Upravené ohniskové vzdálenosti a středové souřadnice
    focal_length_adjusted = focal_length * scale_x
    cx_adjusted = original_cx * scale_x
    cy_adjusted = original_cy * scale_y
    # Přepočet normalizované disparity na skutečné hodnoty
```

```
    disparity_actual = vmin + disparity_normalized * (vmax - vmin)
    # Kontrola rozměrů hloubkové a disparity mapy
    height, width = disparity_normalized.shape
    assert depth.shape == (height, width),
    "Depth and disparity maps must have the same dimensions."
    # Vytvoření mřížky pixelových souřadnic
    x = np.tile(np.arange(width), (height, 1))
    y = np.repeat(np.arange(height), width).reshape(height, width)
    # Výpočet 3D souřadnic
    X = (x - cx_adjusted) * depth / focal_length_adjusted
    Y = (y - cy_adjusted) * depth / focal_length_adjusted
    Z = depth
    # Maskování platných hodnot na základě hodnot Z a rozsahu disparity
    valid_mask = (disparity_actual > vmin) & (disparity_actual < vmax)
    # Extrahování platných souřadnic
    X_valid = X[valid_mask]
    Y_valid = Y[valid_mask]
    Z_valid = Z[valid_mask]
    return X_valid, Y_valid, Z_valid
```

Funkce calculate_point_cloud převádí hloubková data získaná z disparity mapy do sady 3D souřadnic, které reprezentují bodové mračno. Funkce přijímá hloubkové hodnoty, normalizované hodnoty disparity, původní ohniskovou vzdálenost, minimální a maximální možné hodnoty disparity, původní souřadnice středu obrazu a jeho rozměry, a rozměry obrazu po změně velikosti. Funkce nejprve přepočítává normalizovanou disparitu na skutečné hodnoty pomocí lineární interpolace mezi vmin a vmax, poté upravuje ohniskovou vzdálenost a souřadnice středu obrazu na základě změněného měřítka. 3D souřadnice X, Y, a Z jsou vypočítány pomocí projektivní geometrie, kde X a Y vznikají z produkce rozdílů pixelových souřadnic a hloubkových hodnot podělené upravenou ohniskovou vzdáleností. Hodnota Z je přímo rovna hloubkovým hodnotám. Výpočet zohledňuje masku platnosti, určující, které body jsou v přijatelném rozsahu disparity a jsou považovány za validní pro další zpracování.

Extrakce hodnot barev

Následně po vygenerování bodového mračna dochází k extrakci barev pixelů, které jsou dále použity pro obarvení 3D mesh. Barvy pro 3D mesh jsou extrahovány z jednoho ze vstupních obrazů. V mém případě se jedná o levý obrázek, protože právě z tohoto pohledu generuji disparity mapu. Tento krok umožňuje přiřazení správné barvy k odpovídajícím 3D bodům, což značně zvyšuje vizuální kvalitu rekonstrukce. Následující ukázka kódu zobrazuje funkci read_image_colors, která extrahuje a připravuje hodnoty barev pixelů pro obarvení 3D meshe.

```
import numpy as np
import cv2
def read_image_colors(
    img_path,  # Cesta k obrázku
```

```
    X_valid,  # 3D x souřadnice platných bodů
    Y_valid,  # 3D y souřadnice platných bodů
    Z_valid,  # 3D z souřadnice (hloubky) platných bodů
    focal_length,  # Původní ohnisková vzdálenost kamery
    cx, cy,  # Původní souřadnice středu obrazu (optického středu)
    original_size,  # Původní rozměry obrázku (šířka, výška)
    resized_size=(960, 540)  # Nové rozměry obrázku,
                             # pokud došlo ke změně velikosti
):
    if resized_size:
        # Přepočet parametrů kamery na základě změny velikosti
        scale_x = resized_size[0] / original_size[0]
        scale_y = resized_size[1] / original_size[1]
        focal_length *= scale_x  # Předpoklad stejného měřítka pro x a y
        cx *= scale_x
        cy *= scale_y
    # Načtení a předzpracování obrázku
    img = cv2.imread(img_path)
    img = cv2.resize(img, resized_size) if resized_size else img
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    # Výpočet pixelových souřadnic v změněném obrázku
    u_valid = np.clip(
        np.round(X_valid * focal_length /
            Z_valid + cx).astype(int), 0, img.shape[1] - 1
    )
    v_valid = np.clip(
        np.round(Y_valid * focal_length /
            Z_valid + cy).astype(int), 0, img.shape[0] - 1
    )
    # Extrakce barev z obrázku vypočtených souřadnicích
    colors = img[v_valid, u_valid]
    # Normalizace barev
    return colors / 255.0
```

Funkce read_image_colors extrahuje barevné hodnoty z obrázku na základě 3D souřadnic a parametrů kamery, přičemž zohledňuje možnou změnu velikosti obrázku. Načítá obrázek ze zadané cesty a případně jej přizpůsobuje novým rozměrům. Parametry kamery, jako je ohnisková vzdálenost a střed obrazu, jsou upraveny na základě poměru změny velikosti. Poté se vypočítají pixelové souřadnice pro platné 3D body pomocí projektivní transformace, kde jsou 3D souřadnice převedeny na 2D pixelové pozice v obrázku. Barvy na těchto pozicích jsou extrahovány a normalizovány na hodnoty mezi 0 a 1, což poskytuje normalizované barevné hodnoty pro každý platný bod v bodovém mračnu. Tento proces je klíčový pro přiřazení správných barev k odpovídajícím 3D bodům v rekonstrukci, což zvyšuje vizuální kvalitu a realismus 3D modelu.

Generování 3D meshe

Posledním krokem v celém procesu 3D rekonstrukce je kromě vizualizací výsledků, samotné generování výsledného 3D meshe z bodového mračna. Pro tuto činost používám metodu Poisson Surface Reconstruction z knihovny open3D. Zkoušel jsem i ostatní metody zmíněné v teoretické části, ale s touto metodou jsem dosáhl nejlepších výsledků.

Následující ukázka kódu s komentáři zobrazuje generování 3D meshe.

```
import numpy as np
import open3d as o3d
def generate_3D_mesh(X_valid, Y_valid, Z_valid, colors):
    """
    Generuje 3D mesh z platných souřadnic X, Y, Z a odpovídajících
    barevných dat s využitím Poisson Surface Reconstruction.
    Args:
        X_valid (numpy.ndarray): Pole souřadnic X.
        Y_valid (numpy.ndarray): Pole souřadnic Y.
        Z_valid (numpy.ndarray): Pole hloubkových souřadnic Z.
        colors (numpy.ndarray): Pole RGB barevných dat s hodnotami
                                od 0 do 1.
    Returns:
        open3d.geometry.TriangleMesh:
            Objekt trojúhelníkového mesh s body a barvami.
    """
    # Vytvoření objektu bodového mračna
    pcd = o3d.geometry.PointCloud()
    points = np.vstack((X_valid, Y_valid, Z_valid)).T
    pcd.points = o3d.utility.Vector3dVector(points)
    # Ověření rozsahu a tvaru barev
    if colors.max() > 1:
        colors = colors / 255.0  # Normalizace barev,
            pokud nejsou v rozmezí 0 až 1
    assert (
        colors.shape[0] == len(Z_valid) and colors.shape[1] == 3
    ), "Nesprávný tvar pole barev nebo neplatný rozsah."
    # Přiřazení barev k bodovému mračnu
    pcd.colors = o3d.utility.Vector3dVector(colors)
    # Provedení Poisson surface reconstruction
    mesh, densities = o3d.geometry
                          .TriangleMesh
                          .create_from_point_cloud_poisson(pcd, depth=9)
    # Volitelné odstranění vertexů s nízkou hustotou
    vertices_to_remove = densities < np.quantile(densities, 0.01)
    mesh.remove_vertices_by_mask(vertices_to_remove)
    return mesh
```

Funkce generate_3D_mesh slouží k vytváření 3D mesh modelu z platných 3D souřadnic X, Y, Z a odpovídajících barevných dat s použitím techniky Poisson Surface Reconstruction. Funkce nejprve vytváří objekt bodového mračna, do kterého přidává 3D body a barvy. Pokud nejsou barvy v rozsahu od 0 do 1, provádí se jejich normalizace dělením 255. Po vytvoření bodového mračna se na něj aplikuje Poissonova metoda pro vytvoření trojúhelníkového mesh. Během tohoto procesu je možné volitelně odstranit body s nízkou hustotou, což může zlepšit vizuální kvalitu a snížit počet vertexů v mesh. Výsledný model je poté nakonec vizualizován.

7.2.2 Modul trénování a architektury modelu CNN

Další možnou funkcí, která se dá spustit z hlavní funkce kódu (*main*) je *start_training*. Výstupem této funkce je uložení natrénovaného modelu, jehož použití bylo naznačeno v předchozí kapitole (Generovaní disparity mapy). V této kapitole se soustředím právě na

tento model. Tato sekce je rozdělena do tří částí. První se věnuje architektuře modelu. Následující část popisuje přípravu dat a poslední sekce se věnuje procesu trénování. Tento CNN model tvoří jádro celého programu.

Architektura modelu CNN

V této sekci popisuji architekturu mého CNN modelu. Budu zde popisovat jednotlivé vrstvy a důvod pro jejich implementaci. Popis architektury modelu je zároveň popis třídy, která tento model definuje. Popis architektury bude rozdělen do tří částí, podle metod této třídy. Tyto metody jsou inicializace, forward pass metoda a nakonec metoda (*crop_and_concat*). U všech těchto metod budou zobrazeny ukázky kódu s komentáři.

Inicializační metoda

Třída EnhancedCustomCNN je implementována jako potomek třídy nn.Module, což je standardní základ pro všechny neuronové sítě v knihovně PyTorch. Dědičnost z této třídy umožňuje modelu efektivně integrovat metody pro zpracování dat, učení a predikce, což je klíčové pro vysokou flexibilitu a rozšiřitelnost neuronových sítí. Inicializační metoda EnhancedCustomCNN je navržena tak, aby nastavila všechny potřebné vrstvy a parametry potřebné pro efektivní trénink a aplikaci modelu. Následující kód ukazuje, jak je tato metoda strukturována, včetně inicializace různých konvolučních, poolingových a normalizačních vrstev, které jsou základem pro kódovací a dekódovací fáze modelu konvoluční neuronové sítě.

```
import torch
import torch.nn as nn
import torch.nn.functional as F
class EnhancedCustomCNN(nn.Module):
    def __init__(self, disparity_shape=(540, 960)):
        super(EnhancedCustomCNN, self).__init__()
        self.leaky_relu_slope = 0.1  # Koeficient úniku pro LeakyReLU
        self.dropout_p = 0.5  # Pravděpodobnost dropoutu
        # Vrstvy kódování: Konvoluce s aktivací a normalizací
        self.conv1 = nn.Sequential(nn.Conv2d(6, 32
                                             ,kernel_size=3, padding=1),
        nn.LeakyReLU(self.leaky_relu_slope), nn.BatchNorm2d(32))
        self.conv2 = nn.Sequential(nn.Conv2d(32, 64
                                             ,kernel_size=3, padding=1),
        nn.LeakyReLU(self.leaky_relu_slope), nn.BatchNorm2d(64))
        self.conv3 = nn.Sequential(nn.Conv2d(64, 128
                                             ,kernel_size=3, padding=1),
        nn.LeakyReLU(self.leaky_relu_slope), nn.BatchNorm2d(128))
        self.conv4 = nn.Sequential(nn.Conv2d(128, 256
                                             ,kernel_size=3, padding=1),
        nn.LeakyReLU(self.leaky_relu_slope), nn.BatchNorm2d(256))
        self.conv5 = nn.Sequential(nn.Conv2d(256, 512
                                             ,kernel_size=3, padding=1),
        nn.LeakyReLU(self.leaky_relu_slope), nn.BatchNorm2d(512))
```

```
self.conv6 = nn.Sequential(nn.Conv2d(512, 1024
                                      ,kernel_size=3, padding=1),
nn.LeakyReLU(self.leaky_relu_slope), nn.BatchNorm2d(1024))
self.pool = nn.MaxPool2d(2, 2)
# Pooling vrstva redukuje rozměry o polovinu
# Vrstvy dekódování: Transponované konvoluce s
                              # aktivací a normalizací
self.upconv1 = nn.Sequential(nn.ConvTranspose2d(1024, 512,
                                      kernel_size=2, stride=2),
nn.LeakyReLU(self.leaky_relu_slope), nn.BatchNorm2d(512),
                                    nn.Dropout(self.dropout_p))
self.upconv2 = nn.Sequential(nn.ConvTranspose2d(1024, 256,
                                      kernel_size=2, stride=2),
nn.LeakyReLU(self.leaky_relu_slope), nn.BatchNorm2d(256),
                                    nn.Dropout(self.dropout_p))
self.upconv3 = nn.Sequential(nn.ConvTranspose2d(512, 128,
                                      kernel_size=2, stride=2),
nn.LeakyReLU(self.leaky_relu_slope), nn.BatchNorm2d(128),
                                    nn.Dropout(self.dropout_p))
self.upconv4 = nn.Sequential(nn.ConvTranspose2d(256, 64,
                                      kernel_size=2, stride=2),
nn.LeakyReLU(self.leaky_relu_slope), nn.BatchNorm2d(64),
                                    nn.Dropout(self.dropout_p))
self.upconv5 = nn.Sequential(nn.ConvTranspose2d(128, 32,
                                      kernel_size=2, stride=2),
nn.LeakyReLU(self.leaky_relu_slope), nn.BatchNorm2d(32),
                                    nn.Dropout(self.dropout_p))
self.upconv6 = nn.Sequential(nn.ConvTranspose2d(64, 32,
                                      kernel_size=2, stride=2),
nn.LeakyReLU(self.leaky_relu_slope), nn.BatchNorm2d(32),
                                    nn.Dropout(self.dropout_p))
# Konečná konvoluce pro generování mapy disparity
self.final_conv = nn.Conv2d(32, 1, kernel_size=3, padding=1)
self.disparity_shape = disparity_shape
```

Inicializační metoda ve třídě EnhancedCustomCNN inicializuje vrstvy neuronové sítě pro generování disparity map z dvojice stereo obrazů. Tato třída využívá konvoluční vrstvy pro kódování a transponované konvoluční vrstvy pro dekódování. Konvoluční vrstvy jsou konfigurovány s aktivací LeakyReLU a batch normalizací, což zlepšuje konvergenci během tréninku. MaxPooling je použit v kódovací části pro snížení rozměrů a zvýšení abstrakce. Dekódovací část obsahuje vrstvy dropoutu pro redukci přetrénování a zlepšení generalizace modelu. Konečná konvoluce vrací disparity mapu ve specifikovaných rozměrech. Celá architektura je navržena tak, aby efektivně zpracovávala stereo obrazy a produkovala přesné disparity mapy, které jsou klíčové pro 3D rekonstrukci scény.

Metoda forward pass

Metoda forward je základním stavebním prvkem každého modelu implementovaného v PyTorch a definuje cestu, kterou data procházejí skrze síť. Ve třídě EnhancedCustomCNN tato metoda aktivně řídí tok dat od vstupních obrazů k výstupní disparity mapě. Po-

užití této metody je nezbytné vždy, když je model vyžádán k predikci, což zahrnuje fáze trénování, validace i testování. Metoda začíná spojením dvou vstupních obrazů, levého a pravého, a postupně aplikuje konvoluční, poolingové a transponované konvoluční vrstvy. Integrace vstupů a jejich postupné zpracování je klíčové pro vytváření přesných disparity map, které jsou následně použity pro 3D rekonstrukci. Tato metoda je také zásadní pro zpětné šíření chyb, což umožňuje modelu učit se a adaptovat se na základě poskytnutých tréninkových dat. Následuje ukázka kódu mé definice metody forward spolu s komentáři.

```
def forward(self, left_img, right_img):
    # Spojení levého a pravého obrazu
    x = torch.cat((left_img, right_img), dim=1)
    # Kódování vstupního obrazu
    x1 = self.conv1(x)
    x2 = self.pool(self.conv2(x1))
    x3 = self.pool(self.conv3(x2))
    x4 = self.pool(self.conv4(x3))
    x5 = self.pool(self.conv5(x4))
    x6 = self.pool(self.conv6(x5))
    # Dekódování a skládání pro zpětné získání rozměrů
    x = self.upconv1(x6)
    x = self.crop_and_concat(x5, x)
    x = self.upconv2(x)
    x = self.crop_and_concat(x4, x)
    x = self.upconv3(x)
    x = self.crop_and_concat(x3, x)
    x = self.upconv4(x)
    x = self.crop_and_concat(x2, x)
    x = self.upconv5(x)
    x = self.crop_and_concat(x1, x)
    x = self.upconv6(x)
    # Generování finální disparity mapy
    # a její přizpůsobení cílovým rozměrům
    disparity_map = self.final_conv(x)
    disparity_map = F.interpolate(disparity_map,
                                  size=self.disparity_shape,
                                  mode='bilinear',
                                  align_corners=False)
    return disparity_map
```

crop_and_concat

Metoda crop_and_concat je specifický mechanismus používaný v architektuře U-Net, který umožňuje efektivně kombinovat rysy z různých vrstev neuronové sítě. Tato metoda se uplatňuje během dekódovací fáze v modelu EnhancedCustomCNN a hraje zásadní roli v procesu zpětného získávání prostorové informace, která byla redukována v kódovacích vrstvách. Její hlavní úlohou je spojit výstupy z transponovaných konvolučních vrstev s odpovídajícími výstupy z předchozích konvolučních vrstev. Tímto způsobem se do procesu dekódování neztrácí detaily a kontextové informace, což je klíčové pro rekonstrukci detailních a přesných disparity map. Metoda crop_and_concat

je využívána při každé iteraci modelu během trénování, validace i testování, což zvyšuje schopnost modelu generalizovat a přesně modelovat požadované výstupy. Následuje ukázka kódu, který definuje metodu crop_and_concat.

```
def crop_and_concat(self, bypass, upsampled):
    """
    Slouží ke spojení vrstev z kódovací a dekódovací části modelu.
    Args:
        bypass (torch.Tensor):
            Tensor obsahující výstupy z kódovací vrstvy,
            které mají být reintegrovány.
        upsampled (torch.Tensor):
            Tensor obsahující výstupy z dekódovací vrstvy,
            ke kterým mají být přidány informace z bypass.
    Returns:
        torch.Tensor:
            Tensor obsahující spojené vrstvy pro další zpracování.
    """
    # Získání rozměrů tensoru z dekódovací vrstvy
    _, _, H, W = upsampled.size()
    # Přizpůsobení rozměrů bypass tensoru na rozměry
    # upsampled tensoru pomocí bilineární interpolace
    bypass_cropped = F.interpolate(bypass,
                                   size=(H, W),
                                   mode='bilinear',
                                   align_corners=True)
    # Spojení upraveného bypass tensoru s tensorem
    # z dekódovací vrstvy podél dimenze kanálů
    return torch.cat((upsampled, bypass_cropped), dim=1)
```

Příprava dat pro trénink modelu

Kvalita tréninkových dat má zásadní vliv na výkonnost a spolehlivost neuronových sítí. Tréninková data by měla být reprezentativní, různorodá a bez chyb, aby model mohl generalizovat na nová data v reálném světě. Níže jsou uvedeny klíčové aspekty kvalitních dat a možné chyby, které mohou ovlivnit učení modelu a na které jsem dbal při výběru vhodného datasetu pro učení mé neuronové sítě.

Data by měla pokrývat celé spektrum situací, které model ve skutečném použití potká. Nedostatečná variabilita v tréninkové sadě může vést k přetrénování, kde model perfektně funguje na tréninkových datech, ale selhává na datech nových nebo mírně odlišných. Middlebury a Sintel datasety poskytují širokou škálu scén, které pomáhají modelu učit se efektivně zpracovávat různé typy vizuálních informací.

Kvalita obrazů a přesnost disparity map jsou klíčové. Rozlišení obrázků musí být dostatečně vysoké, aby model mohl rozlišit jemné detaily. Formát dat musí také uchovávat hloubkovou informaci s vysokou přesností, jak je tomu u formátu PFM u disparity map. Chyby v datech, jako jsou šum, artefakty komprese nebo nekonzistence v disparity mapách, mohou model vést k nesprávným predikcím.

Nedostatečný objem dat může vést k podtrénování, kde model nemá dostatek pří-

kladů pro naučení efektivních a robustních reprezentací. Naopak, nerovnováha v datových třídách může způsobit, že model bude předpojatý k častějším třídám. Je důležité mít vyváženou sadu dat pro všechny kategorie, které model má rozpoznávat.

Popis datasetů Middlebury Stereo Dataset a Sintel Stereo Dataset

Middlebury Stereo Dataset je uznávaná datová sada v oblasti stereovize. Obsahuje obrázky a disparity mapy z různých scén, což zahrnuje interiéry i exteriéry s různým osvětlením a texturami. Obrázky jsou ve formátu PNG s vysokým rozlišením, což umožňuje zachytit jemné detaily a textury. Disparity mapy jsou poskytovány ve formátu PFM (Portable Float Map), což umožňuje uchování přesné hloubkové informace v plovoucí desetinné čárce. Každá stereo sada obsahuje levý a pravý obrázek s odpovídající ground-truth disparity mapou. Jedná se tedy o data z reálných fotografií. Korespondenční disparity mapy jsou generované pomocí metody, která využívá LIDAR. [31]

Sintel Stereo Dataset je odvozen z projektu Open Movie Project a obsahuje stereo obrazové páry z animovaného filmu "Sintel". Data jsou charakteristická svými dynamickými scénami, které obsahují rozsáhlé pohyby a různé scénáře osvětlení. Obrázky jsou rovněž ve formátu PNG, což zajišťuje vysokou kvalitu vizuálních detailů. Disparity mapy jsou dostupné ve formátu PFM, což poskytuje vysokou přesnost pro kvantitativní vyhodnocení modelů. Tato datová sada je ideální pro testování algoritmů schopných zpracovávat komplexní scény s různými typy pohybů a textur. Jedná se tedy o syntetická data. [32]

V mém programu využívám na učení tyto dva datasety. Hlavní rozdíl mezi nimi je, že jeden dataset (Middlebury Stereo Dataset) je tvořen fotografiemi z reálného světa a disparity mapami měřenými poměrně přesnou metodou LIDARu. Druhý dataset je syntetického rázu, vygenerovaný v blenderu. To reálně znamená, že disparity mapy jsou naprosto přesné. Synteticky vygenerovaná data však neobsahují žádný šum a reálně se i u složitých scén vyznačují jednodušším a přímočařejším zpracováním. Pro kvalitní učení mého modelu jsem musel tyto datasety sjednotit. Bez tohoto sjednocení jsem nedosahoval dobrých výsledků, protože model se zlepšoval na jednom typu dat a zhoršoval se na datech z druhého datasetu. Tohoto sjednocení jsem dosáhl díky implementaci augumentací a transformací při načítání datasetu. Dalším rozdílem je, že Middlebury Stereo Dataset obsahuje přesné kalibrační parametry kamer. Toto je velká výhoda a proto jsem se nakonec rozhodl soustředit se převážně na tento dataset. Model je ale plně schopen učit se a transformovat data i ze Sintel Stereo datasetu.

Načítání dat - třída StereoDataset

Za načítání a předpřípravu dat pro model při učení se stará třída `StereoDataset`, která je odvozena od `torch.utils.data.Dataset` a slouží k efektivnímu zpracování dat pro stereo vizi. V následující části této sekce popíšu jednotlivé metody této třídy.

Tato struktura třídy umožňuje modelu provádět souběžné operace zpracování dat, učení a predikce. V konstruktoru třídy jsou inicializovány cesty k obrázkům a nastavení pro augmentaci a normalizace. Cesty k obrázkům jsou uloženy jako seznamy a používají se pro dynamické načítání dat během tréninku, což efektivně šetří paměť. Inicializují se také základní transformační funkce, jako jsou převod na tensor, změna velikosti a normalizace, které zajišťují konzistentní formát a rozsah hodnot vstupních dat pro síť:

```
self.to_tensor = ToTensor()
self.resize = Resize(target_size)
self.normalize = Normalize(**normalize_params)
```

Nastavení augmentačních transformací, které zahrnují horizontální převrácení, rotaci a úpravu barev, zvyšují rozmanitost tréninkových dat, což pomáhá modelu lépe generalizovat a zlepšuje jeho odolnost proti přetrénování. Změna velikosti obrazů na jednotný rozměr je zásadní pro zajištění konzistence mezi tréninkovými daty a pro efektivní zpracování sítí. Tato augmentace slouží ke zmenšení nebo zvětšení obrazu a může ovlivnit pixely a detaily, což je důležité při extrakci příznaků pro neuronové sítě. Následující kód ukazuje, jak jsou augmentace nastaveny:

```
self.horizontal_flip_prob = horizontal_flip_prob
self.rotation_range = rotation_range
self.color_jitter =
ColorJitter(**color_jitter_params)
if color_jitter_params else lambda x: x
```

Převod obrazů z PIL formátu do tensorů PyTorch, který je nutný pro zpracování dat pomocí PyTorch frameworku, umožňuje efektivní výpočty na GPU, což značně zrychluje proces trénování. Normalizace dat na standardní rozsah je kritická pro kvalitní konvergenci při trénování. Tato augmentace stabilizuje učení modelu.

Metoda `__getitem__` získává data, aplikuje transformace a vrátí tensorové reprezentace obrázků a disparity mapy. Tato metoda je klíčová pro zajištění správného zpracování vstupních dat před jejich předáním do neuronové sítě. Augmentace jsou aplikovány konzistentně na všechny související obrázky, což je důležité pro udržení správného vztahu mezi nimi. Data jsou nakonec převedena do tensorů a normalizována, aby odpovídala očekáváním vstupní vrstvy neuronové sítě. Výsledkem je trojice tensorů připravených pro trénink nebo validaci modelu.

Tím, že model vidí data ve více formách, se snižuje riziko přeučení na tréninkových datech a zvyšuje se schopnost modelu fungovat spolehlivě v reálném světě. Použitím

těchto technik se zvyšuje pravděpodobnost, že model bude schopen správně interpretovat nová, neviděná data, což je obzvláště důležité v aplikacích 3D rekonstrukce, kde se scény a objekty mohou významně lišit.

Trénování modelu CNN

V této sekci popisuji trénovací algoritmus pro danou neuronovou síť. Výstupem je zde kód spustitelný z main funkce. Výstupem z této funkce je natrenovaný uložený model ve formátu pth.

Ukládání modelu během tréninku do formátu .pth představuje základní mechanismus v knihovně PyTorch, který umožňuje zachování stavu modelu pro jeho pozdější použití. Tento formát souboru, specifický pro PyTorch, obsahuje všechny váhy a parametry modelu a je esenciální pro efektivní obnovu trénovaného modelu bez nutnosti opakování celého tréninkového procesu. Uložené modely mohou být následně načteny k další evaluaci, pokračování v tréninku nebo pro aplikaci. V případě mého projektu je uložený model načten a použit na generování disparity map.

Jedná se o dozorované učení (supervised learning), což je metoda, při které se model učí z tréninkového datasetu obsahujícího vstupní data a přesně označené výstupy (v tomto případě disparity mapy).

Funkce train_custom_cnn zajišťuje celý tréninkový proces modelu EnhancedCustomCNN. Začíná načítáním tréninkových a validačních dat pomocí funkce load_data a vytváří instance StereoDataset, které jsou následně využívány pro iterativní trénink a validaci modelu v rámci datových načítačů DataLoader.

Model je přenesen na dostupné výpočetní zařízení (GPU pokud je dostupné, jinak CPU) a je nastaven s optimalizátorem AdamW a kritériem ztráty SmoothL1Loss, což je vhodné pro regresní úlohy jako je predikce disparity. Scheduler OneCycleLR je použit pro adaptivní úpravu učícího koeficientu během epoch, což pomáhá zlepšit konvergenci.

Tréninková smyčka iteruje přes tréninková data, kde každý obrázek je zpracován modelem a porovnáván s cílovými disparity mapami. Ztráta je počítána a propagována zpět pro aktualizaci vah modelu. Efektivita modelu je monitorována pomocí metriky pixel_accuracy, která hodnotí procento pixelů, jejichž disparity je v rámci definovaného prahu od skutečné hodnoty.

Validační fáze testuje model na neviděných datech a ukládá vizualizace výsledků pro lepší interpretaci modelového výkonu. Nejlepší model je ukládán pouze pokud jeho výkon překoná dosavadní nejlepší výsledky, což zajišťuje, že ukládaný model je ten s nejnižší validační ztrátou.

Trénink může být předčasně ukončen pomocí metody early stopping, která sleduje

počet epoch bez zlepšení a přeruší trénink, pokud se model nepodaří zlepšit po definovaný počet epoch. Tato strategie pomáhá zabránit přetrénování modelu na tréninkových datech.

Inicializace a nastavení

Model je inicializován a trénován na CPU pro zjednodušení a eliminaci potenciálních problémů s CUDA. Používá se Smooth L1 loss funkci, která je robustnější proti výchylkám než tradiční L2 loss.

$$L_{\text{smooth}_{L1}}(x, y) = \begin{cases} 0.5 \cdot (x - y)^2 & \text{pro } |x - y| < 1, \\ |x - y| - 0.5 & \text{jinak.} \end{cases} \tag{7.2}$$

Optimalizátor Adam je zvolen pro efektivní adaptivní optimalizaci vah, jehož aktualizace váh je dána rovnicí:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{7.3}$$

kde $\hat{m}_t$ a $\hat{v}_t$ jsou odhady prvních momentů a druhých momentů gradientů. Scheduler Cosine Annealing upravuje learning rate během tréninku podle funkce kosinu pro dosažení jemnější konvergence.

```
device = torch.device("cpu")
model = EnhancedCustomCNN().to(device)
criterion = nn.SmoothL1Loss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(optimizer,
T_max=num_epochs)
```

Načítání dat

Data jsou načítána pomocí funkce `load_data`, která rozděluje dostupné obrázky na tréninkovou a validační sadu. `DataLoader` je použit pro efektivní zpracování datových sad s možností augmentace.

```
train_left, train_right, train_disp, val_left, val_right,
val_disp = load_data(data_dir)
train_dataset = StereoDataset(train_left, train_right,
train_disp, augment=True)
train_loader = DataLoader(train_dataset, batch_size=1,
shuffle=True, num_workers=0)
val_dataset = StereoDataset(val_left, val_right, val_disp)
val_loader = DataLoader(val_dataset, batch_size=1, num_workers=0)
```

Trénovací cyklus

Během každé epochy model prochází fázemi tréninku a validace. Tréninková fáze zahrnuje zpětnou propagaci chyb pro aktualizaci vah modelu.

```
for epoch in range(num_epochs):
    model.train()
    for i, (left_img, right_img, disparity) in
    enumerate(train_loader):
        left_img, right_img, disparity = [x.to(device)
        for x in [left_img, right_img, disparity]]
        optimizer.zero_grad()
        outputs = model(left_img, right_img)
        loss = criterion(outputs, disparity)
        loss.backward()
        optimizer.step()
```

Validace a ukládání modelu

Během validační fáze se neaktualizují váhy modelu, pouze se vyhodnocuje loss na validační sadě a ukládají se modely s nejlepšími výsledky. Generování heatmap pomáhá vizuálně ověřit výsledky predikce disparity.

```
model.eval()
with torch.no_grad():
    for i, (left_img, right_img, disparity) in enumerate(val_loader):
        left_img, right_img, disparity = [x.to(device)
        for x in [left_img, right_img, disparity]]
        outputs = model(left_img, right_img)
        val_loss = criterion(outputs, disparity)
        if i == 0:
            save_heatmap(outputs.cpu().squeeze().numpy(),
            f"result_disparity_epoch_{epoch+1}.png")
            save_heatmap(disparity.cpu().squeeze().numpy()
            ,f"ground_truth_disparity_epoch_{epoch+1}.png)
```

Výstupy a metriky

Metrika pixelové přesnosti měří procento pixelů, kde odchylka disparity od skutečné hodnoty je menší než zadaný práh, což je kritické pro hodnocení kvality predikcí.

$$\text{Pixel Precision} = \frac{1}{N}\sum_{i=1}^{N} 1(|p_i - t_i| < \text{threshold}) \tag{7.4}$$

kde p_i je predikovaná hodnota, t_i je skutečná hodnota a 1 je indikátorová funkce.

Ukončení trénování

Trénování může být předčasně ukončeno na základě nedostatečného zlepšení validační loss, což pomáhá předejít přetrénování modelu.

```
if epochs_since_improvement >= patience:
    print('Early stopping triggered!')
    break
```

7.2.3 Modul pro hodnocení a srovnávání metod

Tento modul porovnává přístup, kde se ke generování disparity mapy používá umělá inteligence s tradičnějším přístupem ke 3D rekonstrukci ze stereo obrázků. Jedná se o poslední modul, který v této části práce popisuji. Tento modul se skládá z několika částí, ale hlavním výstupem je funkce *compare*, která je spustitelná z funkce *main* a jako argument přijímá cestu k danému naučenému CNN modelu, který chceme porovnávat. Porovnávání funguje tak, že se hledá hodnota RMSE (průměrná kvadratická chyba) u referenční metody (bm) a i u metody, která používá CNN. RMSE se počítá vůči *ground_truth* disparity mapě.

První částí tohoto modulu je funkce, která generuje disparity mapu z definovaného natrénovaného modelu. Následuje ukázka kódu, který toto zprostředkovává.

```
def generate_cnn_disparity(model, left_img_path, right_img_path):
    """Generuje disparity mapu pomocí CNN modelu."""
    # Načtení a předzpracování obrazů
    left_img = load_and_preprocess_image(left_img_path)
    right_img = load_and_preprocess_image(right_img_path)
    # Převod obrazů na tensor a přesun na zařízení
    left_img = left_img.unsqueeze(0)
                       .to(next(model.parameters()).device)
    right_img = right_img.unsqueeze(0)
                       .to(next(model.parameters()).device)
    # Predikce modelu
    model.eval()
    with torch.no_grad():
        output = model(left_img, right_img).squeeze(0).squeeze(0)
    return output.cpu().numpy()
```

Funkce generate_cnn_disparity využívá předtrénovanou konvoluční neuronovou síť ke generování map disparity z dvojice stereo obrazů. Obrazy jsou nejprve načteny a předzpracovány pro zajištění kompatibility se vstupy modelu. Následuje predikce disparity, která je vykonávána v režimu evaluace, aby se zajistilo, že se během predikce neuplatňují prvky tréninku jako dropout. Výstupní disparity mapa je převedena z tensoru na numpy pole pro další zpracování a analýzu.

Referenční metoda generování disparity map je zapouzdřená ve funkci generate_bm_disparity. Následuje ukázka kódu s komentáři.

```
def generate_bm_disparity(
    left_img_path, right_img_path, target_size=(960, 540),
    numDisparities=64, blockSize=15, minDisparity=0, maxDisparity=1000):
    # Načtení obrazů a kontrola jejich existence
    imgL = cv2.imread(left_img_path, cv2.IMREAD_GRAYSCALE)
    imgR = cv2.imread(right_img_path, cv2.IMREAD_GRAYSCALE)
    if imgL is None or imgR is None:
        raise FileNotFoundError("One or both image paths are incorrect.")
    # Změna velikosti obrazů na cílový rozměr
    imgL
 = cv2.resize(imgL, target_size)
    imgR = cv2.resize(imgR, target_size)
    # Inicializace StereoBM a nastavení parametrů
    stereo = cv2.StereoBM_create(numDisparities=numDisparities,
                                 blockSize=blockSize)
    stereo.setMinDisparity(minDisparity)
    # Výpočet disparity mapy
    disparity = stereo.compute(imgL, imgR)
    # Normalizace disparity mapy do rozsahu 0 až 1
    min_disp, max_disp = np.min(disparity), np.max(disparity)
    if max_disp - min_disp != 0:
        disparity_normalized = (disparity - min_disp) /
            (max_disp - min_disp)
    else:
        disparity_normalized = np.zeros_like(disparity)
    return disparity_normalized
```

Funkce generate_bm_disparity používá tradiční metodu Stereo Block Matching (StereoBM) pro generování disparity map z dvojice černobílých obrazů. Po načtení a přizpůsobení velikosti obrazů se inicializuje StereoBM s parametry, které řídí detailnost a rozsah hledání disparity. Disparity mapa je následně vypočítána a normalizována, aby byla v rozmezí od 0 do 1, což umožňuje snadné porovnání s jinými metodami a jednodušší vizualizaci. Tato metoda poskytuje základní porovnání s výsledky získanými pomocí pokročilejších CNN modelů.

Stereo Block Matching (StereoBM) je tradiční metoda pro výpočet disparity map ze stereo obrazů, která je široce používána díky své relativní jednoduchosti a efektivitě. Tato technika je založena na porovnávání bloků pixelů z levého obrazu s odpovídajícími bloky v pravém obrazu.

Funkce generate_bm_disparity implementuje StereoBM s využitím knihovny OpenCV, což je populární nástroj pro zpracování obrazu. V této funkci jsou vstupní obrazy nejprve převedeny do černobílého formátu, což redukuje výpočetní náročnost. Následně jsou obrazy přizpůsobeny na cílový rozměr, který je definován vstupním parametrem. Tato změna velikosti zajišťuje, že bloky pixelů budou správně porovnány napříč celými obrazy.

Inicializace objektu StereoBM zahrnuje nastavení počtu disparity, které mají být vyhledány, a velikosti bloku, který definuje, kolik sousedních pixelů bude zahrnuto v každém porovnání. Tyto parametry mají zásadní vliv na kvalitu výsledných disparity map a jejich výběr by měl být přizpůsoben specifikům zpracovávaných dat. StereoBM dále zahrnuje možnost nastavit minimální a maximální disparity, což omezuje rozsah hledání a může zlepšit rychlost zpracování.

Po výpočtu disparity mapy je důležité provést její normalizaci. Tento krok transformuje vypočtené disparity na normalizovaný rozsah, typicky od 0 do 1, což usnadňuje další analýzu a zpracování mapy. Normalizace je zvláště důležitá, pokud jsou výsledky disparity mapy vizualizovány a porovnávány.

Samotné porovnávání dvou disparity map probýhá ve funkci calculate_rmse. Funkce calculate_rmse slouží k výpočtu průměrné kvadratické chyby (RMSE) mezi predikovanou a skutečnou disparity mapou. Před výpočtem RMSE jsou mapy přizpůsobeny, aby měly stejné rozměry, a RMSE je vypočítáno pouze pro ty oblasti, kde je disparity známá (nejsou to tedy okraje nebo oblasti bez informace o disparity). Následuje ukázka kódu s komentáři.

```
def calculate_rmse(predicted_map, ground_truth_map):
    """Vypočítává RMSE mezi predikovanou a skutečnou disparity mapou."""
    # Přizpůsobení rozměrů predikované mapy, pokud je potřeba
    if predicted_map.shape != ground_truth_map.shape:
        predicted_map = cv2.resize(
            predicted_map, (ground_truth_map.shape[1],
            ground_truth_map.shape[0]),
            interpolation=cv2.INTER_LINEAR)
    # Výpočet RMSE pouze pro platné oblasti
    mask = ground_truth_map > 0  # Vyloučení neplatných oblastí
    mse = mean_squared_error(predicted_map[mask], ground_truth_map[mask])
    return sqrt(mse)
```

Poslední částí modulu, který se zabývá porovnáváním metod je funkce compare_disparity_maps. V rámci funkce jsou nejprve inicializovány seznamy pro uchování hodnot RMSE (Root Mean Square Error) pro obě metody. Následuje vytvoření souboru markdown, do kterého jsou zapisovány výsledky jednotlivých porovnání. Funkce iteruje přes prvních 30 složek validačních dat.

Pro každou složku jsou generovány disparity mapy pomocí CNN a StereoBM. Tyto mapy jsou následně přizpůsobeny stejným rozměrům jako skutečná disparity mapa pro správný výpočet RMSE. RMSE je vypočítáno pro každou metodu a zaznamenáno do seznamů a markdown souboru pro další analýzu.

Vizualizace disparity map společně se skutečnými hodnotami je realizována pomocí knihovny Matplotlib, kde jsou výsledky prezentovány vedle sebe pro snadné porovnání. Všechny vygenerované grafy jsou uloženy do předem definovaného adresáře.

Na konci procesu funkce vypočítá průměrné hodnoty RMSE pro obě metody a zapisuje je do markdown souboru. Tyto průměrné hodnoty poskytují ucelený přehled o výkonnosti obou metod přes všechny testované datasety, což je klíčové pro objektivní hodnocení a rozhodování o dalším využití konkrétních technologií v aplikacích.

Tento přístup k porovnání metod umožňuje nejen kvantifikovat rozdíly v přesnosti disparity map, ale také vizuálně posoudit, jak dobře jednotlivé metody zvládají různé typy scén a úrovně detailů. Výsledky tohoto porovnání mohou být klíčové pro výběr nejvhodnější techniky pro specifické aplikace v oblasti počítačového vidění.

8 PŘEDSTAVENÍ A VYHODNOCENÍ VÝSLEDKŮ

V této sekci představím výsledky a srovnání mého programu, který využívá konvoluční neuronovou síť (CNN) ke generování disparity mapy, s tradičním a osvědčeným řešením. Pro tradiční přístup jsem zvolil metodu Stereo Block Matching (StereoBM). Tato metoda byla vybrána díky své relativní jednoduchosti implementace a robustnosti.

Porovnávání na základě disparity mapy mi připadá ideální, protože kroky následující po vygenerování disparity jsou u obou přístupů stejné. Navíc je velkou výhodou, že mám k dispozici "ground truth"disparity mapy, které slouží jako přesné referenční body pro srovnání.

Výsledné disparity mapy jsou srovnávány na validačních datech s ground truth disparity mapami, které slouží jako referenční body. Pro vyhodnocení výsledků používám metriku Root Mean Squared Error (RMSE), která měří průměrnou kvadratickou chybu mezi predikovanou a skutečnou mapou. Tato metrika poskytuje přesný a konzistentní způsob měření rozdílů mezi mapami.

V závěru této sekce prezentuji ukázky výsledků z jednotlivých fází transformace dvou obrázků do 3D modelu spolu s vizuálními vstupy. K těmto ukázkám přidávám také své subjektivní vizuální hodnocení, které reflektuje celkový dojem a kvalitu zpracování.

8.1 Popis vybrané tradiční metody

Pro tradiční metodu jsem zvolil postup založený na Stereo Block Matching (StereoBM). StereoBM je efektivní a široce užívaná metoda pro výpočet disparity map ze stereo obrazů, založená na principu porovnávání bloků pixelů mezi levým a pravým obrazem. Tato technika identifikuje relativní posun mezi bloky, který je interpretován jako disparity, a tím umožňuje rekonstrukci 3D informací ze 2D obrazů. Disparity je inversně proporcionální ke vzdálenosti objektů od kamery.

Implementace StereoBM využívá knihovnu OpenCV a zahrnuje několik klíčových kroků. Nejprve jsou vstupní obrazy převedeny do černobílého formátu a následně přizpůsobeny cílovému rozměru, což zajišťuje, že porovnávané bloky pixelů mají správnou velikost a jsou konzistentně distribuovány napříč celými obrazy. Tato úprava velikosti je nezbytná pro efektivní a správné porovnání bloků.

Iniciace StereoBM objektu zahrnuje nastavení počtu disparity, které mají být vyhledány, a velikosti bloku pixelů, která určuje, kolik sousedních pixelů je zahrnuto v každém porovnání. Tato konfigurace má zásadní vliv na kvalitu a přesnost výsledných disparity map. Dále lze nastavit minimální a maximální hodnoty disparity, což omezuje rozsah hledání a může významně zlepšit efektivitu zpracování.

8.2 Popis vyhodnocovací funkce

Pro vyhodnocení disparity mapy používám metodu Root Mean Squared Error (RMSE), která měří průměrnou kvadratickou chybu mezi predikovanou a skutečnou (ground truth) disparity mapou.

8.3 Popis vyhodnocovacích dat

Obě metody budou testované na validačním datasetu, ze kterého se konvuluční neuronová síť neučila. Tento validační dataset bude obsahovat 30 obrázků. V tomto datasetu budou obsaženy reálné obrázky.

8.4 Prezentace získaných výsledků

Výsledky porovnání

Tabulka obsahuje výsledky porovnání mezi dvěma metodami: CNN a StereoBM.

Tabulka 8.1 Porovnání Disparitních Metod

Test Folder	**CNN RMSE**	**StereoBM RMSE**
Backpack-perfect	0.241	0.318
chess1	0.327	0.473
curule1	0.271	0.437
curule2	0.283	0.395
Umbrella-imperfect	0.323	0.486
Sword2-imperfect	0.280	0.453
Cable-perfect	0.315	0.460
Classroom1-perfect	0.376	0.227
Mask-imperfect	0.310	0.590
Playroom-imperfect	0.261	0.467
Sticks-imperfect	0.181	0.256
ladder1	0.264	0.358
ladder2	0.317	0.317
Pipes-perfect	0.293	0.357
octogons2	0.295	0.388
Sword1-imperfect	0.246	0.461
podium1	0.377	0.458
Vintage-perfect	0.277	0.422
Backpack-imperfect	0.256	0.331
Jadeplant-imperfect	0.286	0.504
Motorcycle-perfect	0.282	0.352
traproom2	0.289	0.418
Cable-imperfect	0.289	0.472
Classroom1-imperfect	0.359	0.280
Sword1-perfect	0.244	0.471
skiboots1	0.282	0.504
Mask-perfect	0.312	0.594
Couch-perfect	0.257	0.462
Flowers-imperfect	0.425	0.382
Piano-imperfect	0.245	0.393
Průměr	**0.292**	**0.416**

Interpretace výsledků

Výsledky v tabulce ukazují jednotlivé hodnoty RMSE (Root Mean Squared Error) pro dvě metody generování disparity map: CNN a StereoBM. Průměrné hodnoty RMSE pro každou metodu jsou uvedeny ve spodní části tabulky.

RMSE měří průměrnou kvadratickou chybu mezi predikovanými hodnotami a ground truth disparity mapou. Nižší RMSE naznačuje přesnější model nebo metodu, protože je menší rozdíl mezi předpovězenou a skutečnou disparity mapou. Vyšší RMSE ukazuje větší chybu, což znamená méně přesnou metodu.

Interpretace pro ukázkovou hodnotu RMSE 0.2:

- **Praktický význam:** RMSE 0.2 znamená, že v průměru je kvadratický rozdíl mezi předpovězenými a skutečnými hodnotami disparity 0.2 pixelů. Po odmocnění je průměrný rozdíl zhruba 0.447 pixelů (protože 0.2 0.447).
- **Důsledky:** Menší rozdíl ukazuje na přesnější model, což vede k lepší kvalitě disparity map a přesnější rekonstrukci 3D modelů.

Tabulka ukazuje, že metoda CNN má průměrnou hodnotu RMSE 0.292, což je nižší než u tradiční metody StereoBM 0.416, což znamená, že metoda CNN je přesnější v tomto srovnání.

Porovnání poskytuje jak kvantitativní, tak kvalitativní měřítko pro hodnocení těchto přístupů, což je důležité pro pochopení, která metoda je vhodnější pro daný úkol.

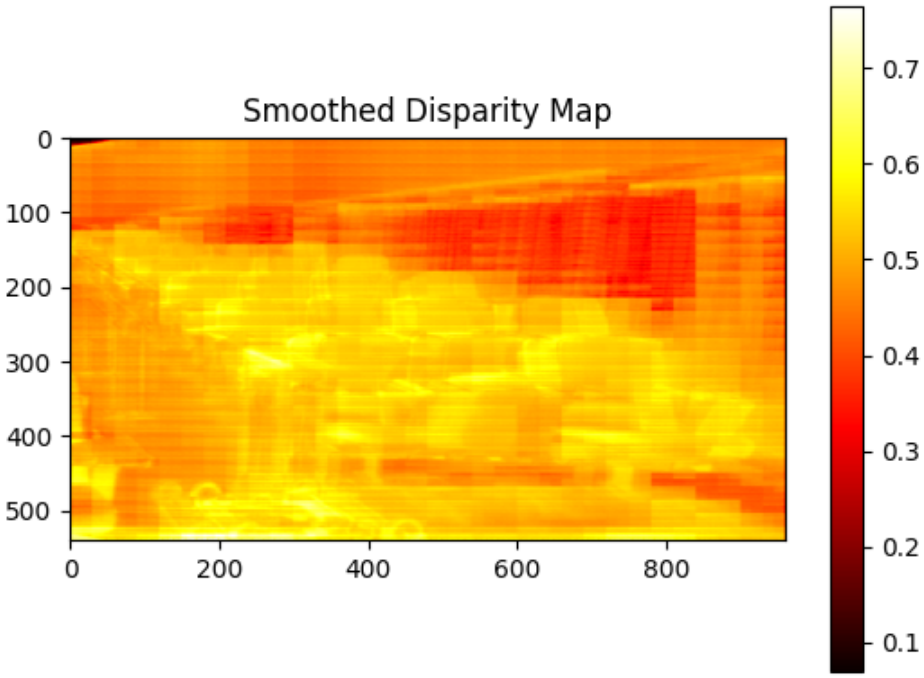
Nedovolím si nyní porovnávat rychlost těchto dvou přístupů. Můj program je psaný v pythonu a není optimalizovaný (mnoho ukládání, načítání, kontrol, kreslení grafů, atd.). U metody StereoBM také velmi rychlost ovlivňuje dané nastavení parametrů.

8.5 Vizuální prezentace průběhu transformace do 3D modelu

První obrázek zobrazuje jeden ze vstupních stereo-obrázků. V tomto případě se jedná se o levý obrázek.

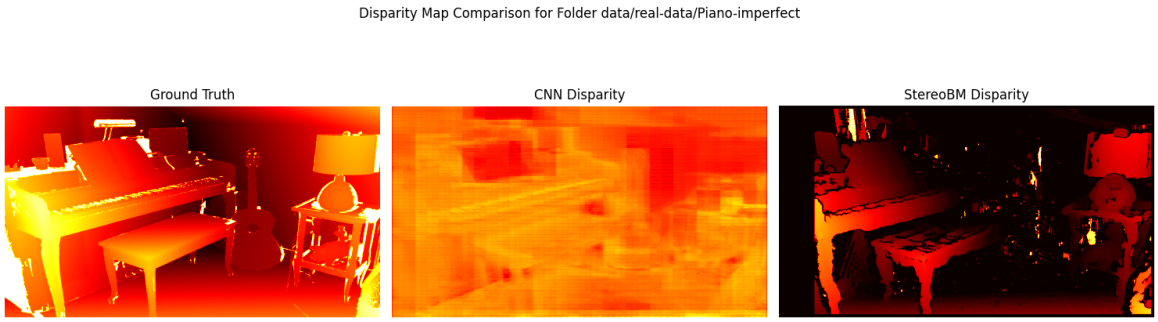
Obrázek 8.1 Vstupní obrázek [31]

Dalším mezikrokem a zároveň také výstupem z mého CNN modelu je disparity mapa, která v sobě už má i informaci o tom, které pixely jsou blíž a které dál. Světlejší barvy značí bližší pixely a tmavší barvy značí větší vzdálenost od kamery.



Obrázek 8.2 Disparity mapa (vlastní zdroj)

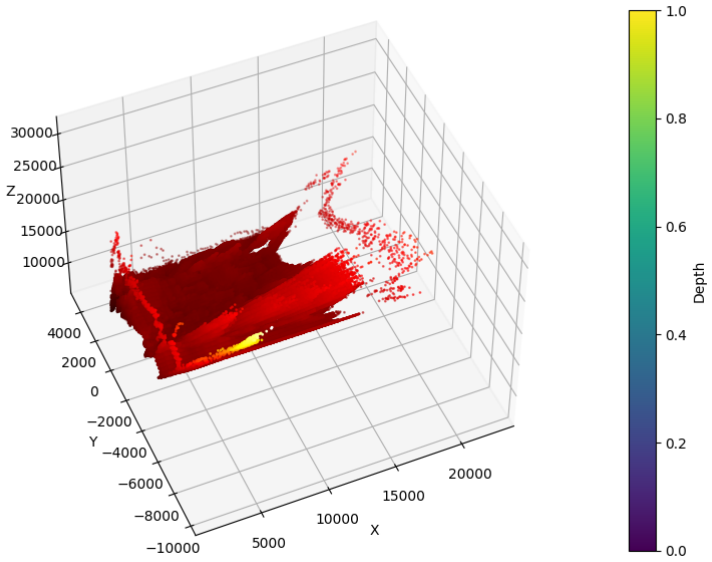
Následuje porovnání mezi ground truth disparity mapou a disparity mapou z CNN modulu a referenční metody.



Obrázek 8.3 Porovnání vygenerovaných disparity map s ground truth (vlastní zdroj)

StereoBM dokáže lépe než aktuální metoda založená na CNN generovat ostré hrany. Metoda založená na neuronové síti má zase hlubší pohled a obecně lepší rozložení disparitních hodnot, i když s ní vzniká šum.

Po vygenerování disparity mapy následuje tvorba bodového mraku, který má na vstupu disparity mapu a k úspěšnému dokončení tohoto kroku jsou také nutné přesné parametry kamery. V zobrazeném příkladě využívám předem získané parametry z datasetu. Jednotlivé vnější pixely jsou obarveny na základě barev ze vstupního obrázku. Bodový mrak je dále transformován do formátu 3D mesh.



Obrázek 8.4 Bodové mračno (vlastní zdroj)

Obrázek 8.5 3D model (vlastní zdroj)

Celkově jsem osobně s výsledky mého programu u většiny příkladů spokojený. Na výsledném 3D modelu jsou ale vidět chyby. Jedno ze základních omezení 3D rekonstrukce je neexistující informace o objektech skrytých za jinými objekty (průhledná místa). Dále si na těchto ukázkách můžeme také všimnout mírného rozmazání a deformace. Deformace je největší ve směru hloubky.

9 BUDOUCÍ VÝVOJ A ZLEPŠENÍ

Při práci na projektu mé diplomové práce jsem nestihl tento projekt dostat do stavu, jaký jsem původně plánoval. Mám v plánu tento projekt dále rozvíjet. Hlavní budoucí plánované změny spočívají v začistění kódu a k převodu projektu do open-source knihovny s kvalitním api, tak aby se tato knihovna dala používat jako součást jiných projektů. Součástí dalšího vývoje bude také zkvalitnění výsledků a optimalizace efektivnosti a rychlosti zpracování. Abych tohoto dosáhl, tak budu muset dále rozšiřovat a zlepšovat architekturu CNN modelu a také budu muset získat rozšířenější a komplexnější data. Dále bych se rád zamyslel nad možnostmi optimalizace rychlosti zpracovávání dat a generování výsledků.

ZÁVĚR

Tato diplomová práce představila komplexní analýzu metod pro transformaci 2D obrazů na 3D modely a praktickou implementaci tohoto procesu. První část práce se zabývá literární rešerší, která představila současné technologické možnosti v této oblasti, včetně využití strojového učení, umělé inteligence a technik počítačového vidění. Tato rešerše také zahrnovala metody pro převod 2D obrazů na 3D modely, se zaměřením na generování disparity map a rekonstrukci povrchu.

Druhá část práce představila praktickou implementaci řešení pro převod 2D do 3D pomocí stereovize a umělé inteligence. Tento projekt byl realizován v jazyce Python s využitím knihoven OpenCV a PyTorch, což umožnilo vytvoření programu pro generování disparity map a 3D modelů ve formátu 3D mesh. Výsledky této implementace byly vyhodnoceny na základě porovnání s tradičňější metodou StereoBM (Root Mean Squared Error).

Výsledky byly porovnávány s ground truth disparity mapami na validačním datasetu, přičemž pro vyhodnocení byla použita metrika Root Mean Squared Error (RMSE), která poskytuje přesný a konzistentní způsob měření rozdílů mezi mapami. Tato práce také představila vizuální prezentaci 3D mesh modelů, která spolu s RMSE hodnotami poskytuje ucelený pohled na kvalitu a přesnost implementovaného řešení.

Výsledky ukazují, že model generuje disparity mapy konzistentně a s odpovídajícím rozložením hodnot, což umožňuje spolehlivou rekonstrukci hloubky scén. Tento úspěch naznačuje potenciál pro další praktické aplikace v oblastech, jako je 3D tisk, autonomní řízení a lékařská diagnostika, kde jsou 3D modely nezbytné.

Z praktického hlediska tato práce navrhuje několik kroků pro zlepšení, včetně porovnání s dalšími metodami pro generování disparity map, testování modelu v praktických scénářích, jako je 3D tisk a lékařská diagnostika, a vylepšení architektury modelu, hyperparametrů a augmentace dat, což může vést k přesnějším a robustnějším výsledkům.

Celkově podle mého názoru tato práce přispívá k rozšíření poznatků v oblasti 3D rekonstrukce pomocí metod počítačového vidění a umělé inteligence. Implementace praktického příkladu a jeho analýza poskytuje hlubší vhled do technik a postupů pro transformaci 2D na 3D, což má široký potenciál pro další výzkum a aplikace v různých oborech. Důkladná analýza implementovaného řešení navíc přináší poznatky pro další rozvoj v oblasti zpracování obrazu, strojového učení a umělé inteligence.

SEZNAM POUŽITÉ LITERATURY

[1] BOTSCH, Mario et al. *Polygon Mesh Processing.* CRC Press, 2010. ISBN 978-3-540-35427-6.

[2] HARTLEY, Richard a ZISSERMAN, Andrew. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2003. ISBN 978-0-521-54051-3.

[3] GOODFELLOW, Ian et al. *Deep Learning.* MIT Press, 2016. ISBN 978-0-262-33737-9.

[4] SZELISKI, Richard. *Computer Vision: Algorithms and Applications.* Springer, 2010. ISBN 978-1848829343.

[5] MOONS, T., VAN GOOL, L., a VERGAUWEN, M. 3D Reconstruction from Multiple Images: Part 1-Principles [online]. In: *Foundations and Trends in Computer Graphics and Vision.* 2009, Vol. 4, No. 4, str. 287–404. Dostupné z: http://dx.doi.org/10.1561/0600000007. [cit. 2024-04-04].

[6] WYATT-SPRATT, Simon. *Scanning vs Photogrammetry: the pros and cons of different 3D modelling techniques* [online]. 2018. Dostupné z: http://dx.doi.org/10.13140/RG.2.2.28510.87366. [cit. 2024-04-04].

[7] LINDER, Wilfried. *Digital Photogrammetry: A Practical Course.* 4th Edition. Springer, 2016. ISBN 978-3-319-25967-7.

[8] WEITKAMP, Claus. *Lidar: Range-Resolved Optical Remote Sensing of the Atmosphere.* Springer, 2005. ISBN 978-3540244704.

[9] CASHMAN, Edel. *The Engineering Essentials Behind LiDAR* [online]. In: Electronic Design. 9.4.2021. Dostupné z: `https://www.electronicdesign.com/markets/automotive/article/21160813/onsemi-the-engineering-essentials-behind-lidar`. [cit. 2024-04-04].

[10] Santiago Royo a Maria Ballesta-Garcia. An Overview of Lidar Imaging Systems for Autonomous Vehicles [online]. *Applied Sciences.* Roč. 9, č. 19, 2019, č. článku 4093. ISSN 2076-3417. Dostupné z: `https://doi.org/10.3390/app9194093`. [cit. 2024-04-04].

[11] Bitfab. *3D Scanning With Structured Light* [online]. c2024 . Dostupné z: `https://bitfab.io/blog/3d-structured-light-scanning/`. [cit. 2024-04-04].

[12] ANDREWS L. David. *Structured Light and Its Applications* [online]. Academic Press, 2008. ISBN 978-0-12-374027-4. Dostupné z: https://doi.org/10.1016/B978-0-12-374027-4.X0001-1. [cit. 2024-04-04].

[13] MI, Qingwei a GAO, Tianhan. *3D Reconstruction Based on the Depth Image: A Review.* In: Barolli, Leonard (ed.). *Innovative Mobile and Internet Services in Ubiquitous Computing.* Cham: Springer International Publishing, 2022, s. 172–183. ISBN 978-3-031-08819-3.

[14] S.P. Sahasrabuddhe, B. Lohani a P. Mittal. *Single Camera Techniques for 3D Perception.* Elsevier, 2018. ISBN 978-0-12-374027-4.

[15] SOLEM, Jan Erik. *Programming computer vision with Python.* Beijing: O'Reilly, 2012. ISBN 9781449316549.

[16] SAKAI, Kentaro a YASUMURA, Yoshiaki. *Three-dimensional shape reconstruction from a single image based on feature learning* [oline] In: 2018 International Workshop on Advanced Image Technology (IWAIT). Thailand: IEEE, 2018, s. 1-4. Dostupné z: https://doi.org/10.1109/IWAIT.2018.8369636. [cit. 2024-04-04].

[17] HASSNER, T. a BASRI, R. *Example Based 3D Reconstruction from Single 2D Images.* [online]. In: 2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06). 2006, s. 15-15. Dostupné z: http://dx.doi.org/10.1109/CVPRW.2006.76. [cit. 2024-04-04].

[18] CHEN, Xu, WU, Qingfeng a WANG, Shengzhe. *Research on 3D Reconstruction Based on Multiple Views* [online]. In: 2018 13th International Conference on Computer Science & Education (ICCSE). 2018. Dostupné z: https://doi.org/10.1109/ICCSE.2018.8468705. [cit. 2024-04-04].

[19] LU, Cheng-Hao a CHEN, Xiu-Hong. *Improved iterative Poisson point cloud surface reconstruction* [online]. In: 2023 3rd International Conference on Digital Society and Intelligent Systems (DSInS). 2023. Dostupné z: https://doi.org/10.1109/DSInS60115.2023.10455616. [cit. 2024-04-04].

[20] P. A. Nugroho, D. K. Basuki a R. Sigit. *3D heart image reconstruction and visualization with marching cubes algorithm* [online]. In: 2016 International Conference on Knowledge Creation and Intelligent Computing (KCIC), Manado, Indonesia, 2016. Dostupné z: https://doi.org/10.1109/KCIC.2016.7883622. [cit. 2024-04-04].

[21] Y. S. Elshakhs, K. M. Deliparaschos, T. Charalambous, G. Oliva a A. Zolotas. *A Comprehensive Survey on Delaunay Triangulation: Applications, Algorithms, and Implementations Over CPUs, GPUs, and FPGAs* [online]. In: IEEE Access. 2024. Dostupné z: `https://doi.org/10.1109/ACCESS.2024.3354709`. [cit. 2024-04-04].

[22] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva a G. Taubin. *The ball-pivoting algorithm for surface reconstruction* [online]. In: IEEE Transactions on Visualization and Computer Graphics. Roč. 5, č. 4, Oct.-Dec. 1999. Dostupné z: `https://doi.org/10.1109/2945.817351`. [cit. 2024-04-04].

[23] S. M. Indumathi, N. Prajwala a N. Pushpa. *Implementation of Vertex Colouring Using Adjacency Matrix* [online]. In: 2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT), Mysuru, India, 2021. Dostupné z: `https://doi.org/10.1109/ICEECCOT52851.2021.9708024`. [cit. 2024-04-04].

[24] B. Dong a X. Shen. *The Study of the Terrain Rendering Method Based on Ray Casting* [online]. In: 2014 International Conference on Virtual Reality and Visualization, Shenyang, China, 2014. Dostupné z: `https://doi.org/10.1109/ICVRV.2014.71`. [cit. 2024-04-04].

[25] CHOLLET, François. *Deep learning v jazyku Python: Knihovny Keras, TensorFlow.* Grada, 2019. ISBN 978-80-247-3100-1.

[26] GÉRON, Aurélien. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems.* Third edition. Beijing: O'Reilly, [2023]. ISBN 978-1-098-12597-4.

[27] POUYANFAR Samira, et al. *A survey on deep learning: Algorithms, techniques, and applications* [online]. In: ACM Computing Surveys (CSUR), 2018, s. 1-36. Dostupné z: `http://dx.doi.org/10.1145/3234150`. [cit. 2024-04-04].

[28] ROSEBROCK, Adrian. *Deep learning for computer vision with Python.* [Místo vydání není známé]: PyImageSearch, 2017. ISBN 9781986538138.

[29] JIN Yiwei, JIANG Diqiong, CAI Ming. *3d reconstruction using deep learning: a survey* [online]. In: Communications in Information and Systems, 2020, s. 389-413. Dostupné z: `https://www.intlpress.com/site/pub/files/_fulltext/journals/cis/2020/0020/0004/CIS-2020-0020-0004-a001.pdf`. [cit. 2024-04-04].

[30] YUNIARTI Anny, SUCIATI Nanik. *A review of deep learning techniques for 3D reconstruction of 2D images* [online]. In: 2019 12th International Conference on Information Communication Technology and System (ICTS). IEEE, 2019. s. 327-331. Dostupné z: `https://doi.org/10.1109/ICTS.2019.8850991`. [cit. 2024-04-04].

[31] Middlebury Stereo Datasets. Middlebury College [online]. c2021. Dostupné z: `https://vision.middlebury.edu/stereo/data/`. [cit. 2024-04-04].

[32] D. J. Butler, J. Wulff, G. B. Stanley, M. J. Black, *A naturalistic open source movie for optical flow evaluation* [online]. In: *European Conf. on Computer Vision (ECCV)*, A. Fitzgibbon et al. (Eds.), Springer-Verlag, Part IV, LNCS 7577, Oct. 2012, pp. 611-625. Dostupné z: `http://sintel.is.tue.mpg.de/stereo`. [cit. 2024-04-04].

[33] PyTorch, *PyTorch Documentation* [online]. Dostupné z: `https://pytorch.org/docs/stable/index.html`. [cit. 2024-04-04].

[34] NumPy, *NumPy Documentation* [online]. Dostupné z: `https://numpy.org/doc/`. [cit. 2024-04-04].

[35] OpenCV, *OpenCV Documentation* [online]. Dostupné z: `https://docs.opencv.org/master/`. [cit. 2024-04-04].

[36] Matplotlib, *Matplotlib Documentation* [online]. Dostupné z: `https://matplotlib.org/stable/contents.html`. [cit. 2024-04-04].

[37] Open3D, *Open3D Documentation* [online]. Dostupné z: `http://www.open3d.org/docs/release/`. [cit. 2024-04-04].

[38] Pillow, *Pillow Documentation* [online]. Dostupné z: `https://pillow.readthedocs.io/en/stable/`. [cit. 2024-04-04].

[39] Scikit-learn, *Scikit-learn Documentation* [online]. Dostupné z: `https://scikit-learn.org/stable/documentation.html`. [cit. 2024-04-04].

[40] SciPy, *SciPy Documentation* [online]. Dostupné z: `https://docs.scipy.org/doc/scipy/reference/`. [cit. 2024-04-04].

[41] Numba, *Numba Documentation* [online]. Dostupné z: `https://numba.pydata.org/numba-doc/latest/index.html`. [cit. 2024-04-04].

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

CPU	Central Processing Unit
GPU	Graphics Processing Unit
CNN	Convolutional Neural Network
RMSE	Root Mean Square Error

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I. CD S DIPLOMOVOU PRACÍ VČETNĚ ZDROJOVÉHO KÓDU

Přiložené CD obsahuje:

Diplomovou práci ve formátu .pdf (fulltext.pdf)

Komprimovaný zdrojový kód (prilohy.zip)