Západočeská univerzita v Plzni

AVIN

Diplomová práce

Návrh systému bezznačkové rozšířené reality

Autor:
Andrea Varáčková

FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Obsah

1 Úvod

Počítačové vidění je vědní obor často využívaný v mnoha odvětvích a aplikacích usnadňujících každodenní život. Kamery a video senzory se využívají například k detekci překážek, navigaci, automatické identifikaci předmětů i lidí a k řešení dalších užitečných úloh.

Tato práce se věnuje především bezznačkové rozšířené realitě. Rozšířená realita neboli Augmented Reality (AR) označuje pojem, kde je reálné prostředí kombinováno s virtuálními objekty. Realita není zcela nahrazována, ale je pouze doplněna o počítačově vytvořené objekty, které uživatel vidí přes konkrétní nástroj, kterým může být například monitor počítače, mobilu, nebo třeba brýle vybavené prostředky k zobrazení rozšířené reality. Tato technologie funguje na základě počítačového vidění, kdy zkoumá okolní prostředí za použití kamer. Hledá v něm významné body, které jsou dostatečně odlišitelné od prostředí, a pomocí nich se orientuje při vkládání virtuálního objektu do okolního prostředí. Pro AR mohou být tyto významné body předem známé. Je možné do prostředí vložit například QR kód, výrazné puntíky nebo jiné snadno identifikovatelné symboly.

Toto označení předem však není možné vždy vytvořit. Proto existuje druh AR zvaný bezznačková rozšířená realita, která hledá významné body v libovolném předem neoznačkovaném prostoru.

Pro správnou funkci AR je nutné dokázat se správně orientovat v prostoru. K řešení této úlohy je potřeba splnit několik kroků, kterými jsou

- kalibrace kamer,
- nalezení významných bodů v prostoru,
- sledování a párování významných bodů napříč obrazy,
- návrh virtuálního objektu,
- vkládání objektu do reálného prostředí.

O jednotlivých bodech se blíže píše v následujících kapitolách. Pro dobrou orientaci v prostoru, která je nezbytnou podmínkou správného fungování AR, se využívají algoritmy jako například vizuální odometrie (VO) nebo současná lokalizace a mapování (SLAM).

Druhá kapitola se zabývá výhradně rozšířenou realitou. Nejprve přiblíží pojem AR, poté popíše základní princip fungování a nakonec se zaměří na bezznačkovou AR. Vysvětlí, jak se liší oproti značkové AR a podrobněji popíše její princip.

Tato práce se kromě rozšířené reality věnuje také vizuálnímu SLAMu, který je detailně popsán v Kapitole 3. Je v ní popsán princip jeho fungování a také různé druhy SLAM systémů. Některé jsou pouze stručně zmíněny, protože byly důležité pro vývoj současného SLAMu, jiné více používané jsou pak vysvětleny podrobněji.

Čtvrtá kapitola se zabývá již existujícími systémy. Zmiňuje jak AR systémy obecně, tak SLAM a Vizuální Odometrii. Dále také konkrétní algoritmy věnující se trackování. Popisuje i možné využití neuronových sítí pro tento typ úloh.

V další kapitole je pak řešena praktická část práce. Jsou tam popsány všechny kroky pro vytvoření AR systému na základě SLAMu. Vysvětluje, jak byly jednotlivé části naprogramovány a ukazuje výsledky.

Poslední kapitolou je závěr. V něm je shrnuta celá práce, jak teoretická, tak praktická část. Jsou tam také nastíněny cíle do budoucna a možná zlepšení systému.

2 Rozšířená realita

Variace virtuální reality, která však člověka plně nepohltí do virtuálního prostředí, ale místo toho pouze doplňuje skutečné prostředí počítačově vytvořenými obrazy, se nazývá rozšířená realita [1]. Ať už je k jejímu použití potřeba Head-Mounted display, speciální brýle, nebo obrazovky mobilu, tabletu či jiného zařízení, musí splňovat tři základní požadavky.

Musí **kombinovat reálné a virtuální**, přičemž uživatel vidí oba tyto světy naráz. Je možné do reálného prostředí vkládat pouze symboly sloužící pro orientaci, nápisy, jednotlivé objekty, nebo třeba nahradit i velký kus scenérie. Tímto stylem je možné například z reality "odstranit"nějaký předmět, protože ho uživatel přes virtuální obraz nevidí.

Dále musí být **interaktivní v reálném čase**. Interakce s virtuálními objekty musí probíhat dostatečně rychle, aby to pro uživatele působilo přirozeně a nemusel čekat. V mnoha aplikacích je tato rychlost klíčová.

A konečně musí být **registrované ve 3D**, což znamená, že virtuální objekty, ať už dvou nebo trojrozměrné, jsou vkládány do reálného prostředí, které je 3D. Proto je třeba dbát na to, aby byly vkládány do přesné hloubky a pozice, aby nekazily pocit přirozenosti.

2.1 Základní rozdělení AR

Jedno z hlavních dělení AR systémů je na základě toho, jaký typ technologií využívají. Dělí se na optické a video technologie.

Optické technologie nechávají uživatele vidět reálné prostředí a pouze do něj vkládají virtuální objekty. Využívají k tomu optické kombinéry, které vlastně slouží jako polopropustná zrcadla. Uživatel skrz ně vidí své okolí a na nich se odráží promítaný obraz vkládaného virtuálního objektu. Oproti druhému typu technologií je tato metoda jednodušší a levnější. Jejich implementace je jednoduší, ale zato se mohou potýkat s problémy zkreslení, které vzniká tím, že obraz prochází optickou soustavou, aniž by bylo možné ho upravovat. Další výhodou je, že uživatel stále alespoň částečně vnímá své prostředí, takže nehrozí, že by si mohl ublížit o překážku. Optické technologie také neovlivňují rozlišení reálného světa.

Tento přístup má však i své nevýhody, kterými jsou například to, že nemusí vždy dokonale zakrýt reálný svět a virtuální objekty se mohou zdát průsvitné. Aby bylo možné tomuto předejít, design zobrazovacího zařízení je složitější než u video technologií. Dalším problémem je předejití zkreslení, které je tím výraznější, čím víc širokoúhlý displej je použit. Toto je možné řešit větším množstvím výpočtů, což vede ke zvýšení náročnosti úlohy. Optické technologie mají pouze jedinou informaci o poloze uživatelovy hlavy a to přímo z headsetu, což také může přinášet jisté nepřesnosti.

Video technologie nahrávají okolní prostředí, takto vzniklé video pak digitálně kombinují s virtuálními objekty a výsledek ukazují uživateli. Nevidí tedy skutečné reálné prostředí, ale pouze jeho digitální formu. Headset nemusí mít tak komplexní design protože se vyhýbá problémům působeným optickými kombinéry. Tento přístup může všechna zkreslení řešit digitálně v obrazu, není třeba vytvářet složitá optická řešení. Vyhýbá se také možnou časovou nesouhrou mezi reálným prostředím a virtuálními objekty. Když má vkládání objektu zpoždění, video reálného světa se také zkrátka ukáže s mírným zpožděním. Tyto technologie mají obecně více informací o poloze kamery. Znají polohu headsetu, ale navíc mají k dispozici kompletně celý obraz reálného světa v digitální formě.

I tyto technologie mají ovšem své nevýhody. Může se zde vyskytovat zkreslení, pro-

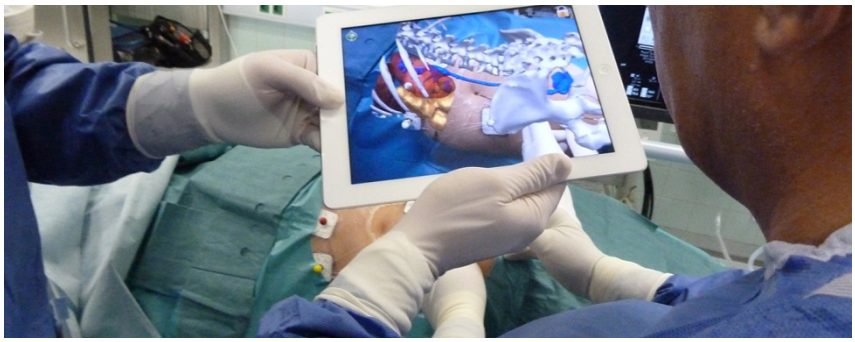
tože pracují s videem okolí i virtuálními předměty v digitální formě. Oba tyto proudy musí být správně synchronizované, aby výsledek působil přirozeně. Celková implementace těchto systému je náročnější než u video technologií. Dále ovlivňuje rozlišení reálného světa, protože ho uživatel vidí jen v digitální formě. Hrozí také větší nebezpečí pro uživatele, protože je reálný svět zakrytý svým digitálním obrazem. Když se přístroj vypne, uživatel přes něj nic nevidí. Může stát, že je realita v obraze malinko posunutá. Oči totiž nejsou postaveny přesně na stejném místě jako kamery. Tento posun je však možné řešit dodatečnými výpočty.

Pro zobrazování AR je ovšem možné používat i jiné přístroje než headsety. Časté využití najdou monitory počítače, tablety nebo mobilní telefony. Záleží na konkrétní aplikaci. V případě počítače dostává systém informace v podobě videa z externí statické nebo pohyblivé, jedné nebo více kamer. Ty pak digitálně zkombinuje s virtuálními objekty a na obrazovce uživateli ukazuje výsledek. V případě mobilu nebo tabletu je možné používat přímo kameru daného zařízení. V praktické části této práce se budu věnovat především využití počítače a mobilního telefonu.

2.2 Využití AR

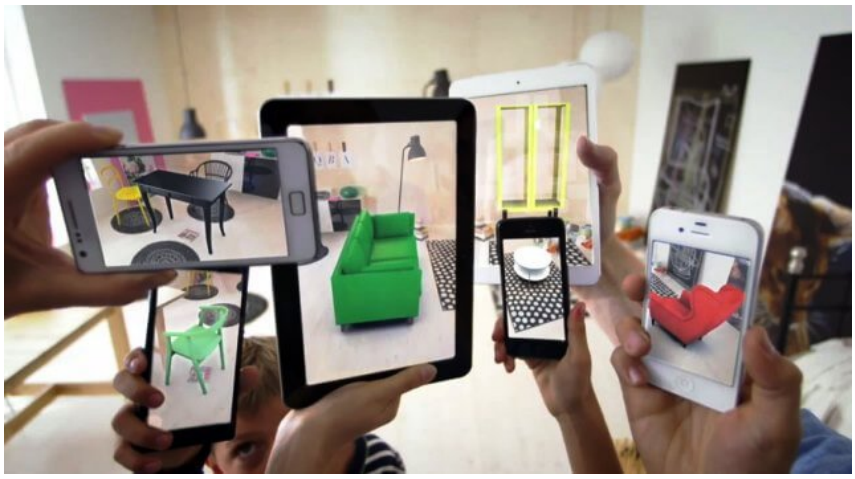
AR nachází své využití ve velkém množství odvětví a ve spoustě různých typů úloh. V následujících odstavcích jsou některé z nich shrnuty [2].

Jedno z velmi důležitých odvětví, které AR využívá stále více je medicína. AR v tomto oboru slouží například pro trénink mediků, kteří se díky headsetu na zobrazování rozšířené reality mohou důkladně seznámit s lidskou anatomií a tak podobně. Anotace také může pomoci při lékařských zákrocích, kde se s velkou přesností například ukáže chirurgovi místo zákroku, viz Obrázek 1.



Obrázek 1: Využití AR v medicíně

Další užitečnou aplikací je schopnost AR zobrazovat věci, které ještě neexistují. Tato vlastnost se využívá při mnoha situacích, viz Obrázek 2. Může například do scenérie vykreslit navrženou budovu a tak pomáhat architektům. Dále ji mohou používat uživatelé při koupi různých předmětů, u kterých si mohou vizualizovat, jak by vypadaly s různými úpravami nebo třeba v jiných barvách. Podobně mohou AR využívat také umělci nebo designéři.

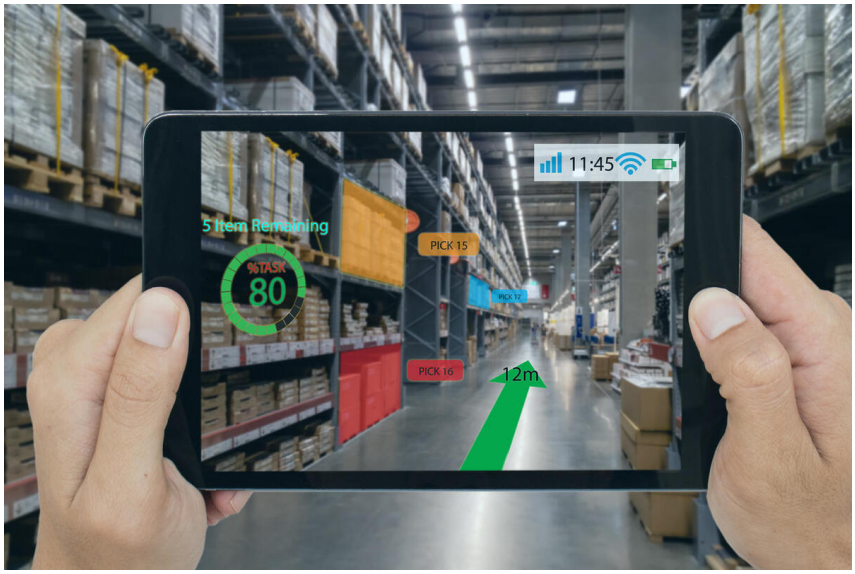


Obrázek 2: Využití AR ve vizualizaci

AR najde velké využití i při opravách nebo údržbě komplexních strojů. Může sloužit jako virtuální, interaktivní návod, přičemž uživatel stále vidí stroj, na kterém musí pracovat. Pomocí headsetu pak mohou vidět jak psané pokyny bez používání obrazovek navíc, nebo třeba zvýrazněná problémová místa, kterým je třeba se věnovat. Všechny potřebné informace se jim zobrazují tehdy, kdy jsou třeba a uživatelé mají obě ruce volné k práci.

V průmyslové logistice bývá AR využíváno například ve skladištích, viz Obrázek 3. Pomocí něj mohou uživatelé zlepšit svou orientaci po skladu, vykreslením map nebo značek do prostoru, zvýrazněním hledaných objektů a tak dále. Tímto se zlepší výkonnost práce, protože se ušetří čas případného hledání nebo bloudění po rozlehlých skladištích.

AR se může využívat i pro navigaci a orientaci v prostoru obecně. Neomezuje se pouze na skladiště, ale má mnohem širší využití.



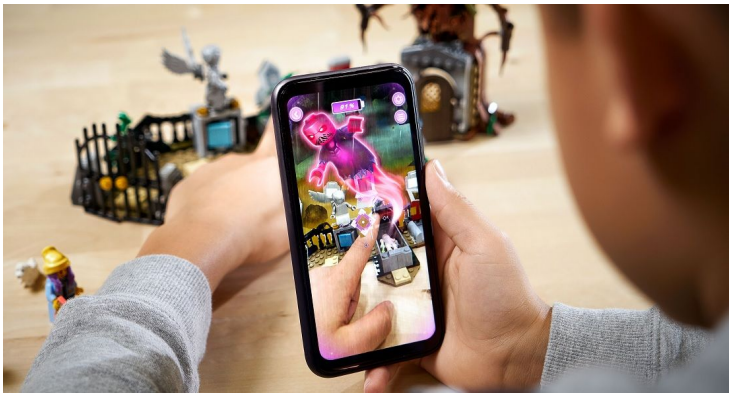
Obrázek 3: Využití AR ve skladištích

Vizualizační schopnosti AR nacházejí velké využití i ve školství nebo turistickém průmyslu, například pro usnadnění orientace v neznámém městě, viz Obrázek 4. Žáci a studenti mohou během výuky sledovat virtuální obrazy toho, o čem se zrovna učí. Turisté si také mohou projít například malým náhledem místa, kam si chtějí koupit letenky.

Obrázek 4: Využití AR v turistice

Vizualizace virtuálních předmětů a anotace předmětů reálných je obecně hojně využívána pro ulehčení mnoha pracovních nebo každodenních aktivit. Vývoj jde stále dopředu, tak se AR využívá ve stále větším množství aplikací a odvětví.

Dalším odvětvím, kde je rozšířená realita často využívána je zábavní průmysl. Ať už jde o využívání AR technologií ve filmařství nebo třeba při tvorbě her. Hry postavené na principu AR jsou čím dál tím rozšířenější. Přes mobilní obrazovku nebo headset je možné do okolí umístit virtuální předměty či například postavičky a interagovat s nimi, viz Obrázek 5.



Obrázek 5: Využití AR pro zábavu

2.3 Bezznačková AR

Bezznačková AR nemá v prostředí k dispozici předem známé body. Úloha AR se tak stává výrazně složitější. Tím se však velmi rozšiřuje pole jejího využití. Hlavní výhodou tohoto přístupu je rozšířený rozsah AR - odpadá omezení na označkovaný prostor. Následující podkapitoly popisují fungování systému AR.

2.3.1 Kalibrace kamery

AR je úloha spadající do počítačového vidění. Pracuje se vstupem v podobě proudů snímku, což dělá celý proces závislý na kamerách. To vede k prvnímu kroku, kterým je kalibrace kamery. Nejprve je nutné provést výpočty kompenzující zakřivení, které způsobuje čočka, jíž obraz prochází. Kalibrace je proces, díky kterému je možné získat parametry kamery, dále používané k dalším výpočtům. Jedním z nich je matice kamery popisující ohniska f_x a f_y a optické středy c_x a c_y. Matice pak vypadá následovně:

$$C_M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Vstupem algoritmu pro získání parametrů je několik snímků, na kterých je možné nalézt body, jejichž obrazové 2D a světové 3D souřadnice jsou známy. K tomuto účelu slouží například fotografie šachovnice umístěné v prostoru. Výstupem algoritmu pak je vlastní matice kamery.

Algoritmus uvedený a blíže vysvětlený na stránkách OpenCV[1] se dělí na 4 kroky. Prvním je definování světových 3D souřadnic bodů za použití šachovnice o známých rozměrech. Poté se pořídí několik snímků zmíněné šachovnice. Dále se naleznou souřadnice pixelů pro každý 3D bod v odlišných obrazech a nakonec se na základě těchto informací vypočítají samotné parametry kamery.

2.3.2 Trackování

Aby bylo možné porozumět obrazu a sledovat pohyb prostředí na po sobě jdoucích snímcích, je nutné nalézt v obraze nějaké orientační body. Musí být dostatečně odlišitelné od svého okolí a při nalezení stejných bodů v různých snímcích je třeba mít možnost určit, jedná-li se o stejný bod. Za tímto účelem vzniklo označení významné body. Mezi ně spadají například rohy - místa, v nichž existují dva dominantní směry, viz Obrázek 6 - nebo oblasti dostatečně jasově odlišitelné od okolí. Ploché regiony ani hrany, po kterých je možné se posunout a nepoznat rozdíl v umístění, s určováním pozice nepomáhají.

Obrázek 6: Ukázka plochého regionu, hrany a rohu

Pro hledání významných bodů se využívají takzvané detektory.To jsou algoritmy, které mají jako vstup obraz prostředí a jako výstup pak lokaci významných bodů, které se v něm nacházejí. Nepodávají žádné bližší informace o nalezených bodech, pouze jejich polohu. Mezi tyto algoritmy patří například:

[1] https://learnopencv.com/camera-calibration-using-opencv/

- **Moravcův detektor rohů** [3] testuje podobnost okolí pixelu, měřenou pomocí sumy absolutních rozdílů. Jako rohy jsou označena lokální maxima sum těchto rozdílů. Pro definici okolí využívá pravoúhlého okénka.

- **Harrisův detektor rohů** [3] je založen na Moravcovu detektoru, ale místo pravoúhlého používá Gaussovské okénko, čímž potlačuje šum. Algoritmus pro každý bod v obraze počítá Harrisovu matici ve tvaru $M = \sum_{x,y} w(x,y) \begin{bmatrix} I_u^2(x,y) & I_u I_v(x,y) \\ I_u I_v(x,y) & I_v^2(x,y) \end{bmatrix}$, která slouží k určení, o jaký typ okolí pixelu se jedná. Detektor rozlišuje tři typy a to plochý region, hranu a roh, viz Obrázek 6. Z matice se aproximací výpočtu vlastních čísel ve tvaru $R = det(M) - k(trace(M))^2$, kde k je konstanta v rozmezí 0.04 až 0.1, dostane hodnota R. Ta spadá do jedné ze tří zmiňovaných kategorií. Je-li vyšší než 10 000, jedná se o roh, je-li nižší než -10 000, značí hranu a vyskytuje-li se mezi, jde o plochý region.

- **Shi-Tomasi detektor rohů** [4] je algoritmus, který modifikuje Harrisův detektor a poskytuje lepší výsledky. Výpočet hodnoty R je nahrazen vzorcem $R = min(\lambda_1, \lambda_2)$, Překoná-li R stanovenou prahovou hodnotu, je bod považován za roh.

- **FAST** [5] je rychlý algoritmus pro detekci významných bodů. Jako další deskriptory je založen na zkoumání okolí pixelu. Pracuje s intenzitami v obraze, které porovnává. Nejprve zvolí pixel s intenzitou I_p k prozkoumání a prahovou hodnotu t. Poté vezme kružnici šestnácti pixelů v okolí zvoleného bodu. Zkoumaný pixel je označen jako roh, je-li v kružnici soubor n po sobě jdoucích pixelů, které jsou jasnější než $I_p + t$, nebo tmavší než $I_p - t$. n bylo zvoleno jako 12. Algoritmus je možno urychlit tím, že se provede rychlý test, který nejprve sleduje pouze pixely, které jsou v kružnici umístěny na pozicích 1, 5, 9 a 13. Pokud je sledovaný pixel roh, alespoň tři z těchto čtyř bodů musí být jasnější než $I_p + t$, nebo tmavší než $I_p - t$.

- **MSER** detektor [6] neboli metoda Maximálně Stabilních Extrémních Regionů (Maximally Stable Extremal Region) se využívá k detekci blobů v obraze. Narozdíl od výše zmiňovaných detektorů, tento nehledá pouze významné body, ale celé oblasti dobře odlišitelné od svého okolí. Algoritmus pracuje s intenzitou pixelů. Nejprve pixely podle intenzity roztřídí a poté určuje, které by mohly patřit do stejného blobu. Takto nalezené oblasti se pak protřídí. Jsou odstraněny například ty, jež jsou moc malé, velké nebo nestabilní. Stabilita regionu je pak určena jako opak relativní variance oblasti poté, co je intenzita zvýšena o předem zvolenou hodnotu.

Příznakové deskriptory se od detektorů liší tím, že významné body v obraze nejen lokalizují, ale vracejí i jejich deskriptory. To jsou takzvané příznakové vektory obsahující dodatečné informace, které mají využití mimo jiné právě v úloze trackování. Deskriptory obsahují vektory, do nichž je zakódovaná geometrie oblasti kolem daného bodu. To je významné pro identifikaci bodů. Díky tomu je možné od sebe jednotlivé body odlišit a provádět tak jejich porovnávání a sledování napříč obrazy. Detektory by ideálně měly být invariantní vůči transformaci, tedy translaci, rotaci i změně měřítka. Mezi algoritmy, hledající a popisující významné body patří například:

- **SIFT** [7] je škálově invariantní algoritmus, který extrahuje klíčové body a vypočítá jejich deskriptory. Nejprve detekuje extrémy za použití prostorového měřítka. Využívá různé velikosti oken k nalezení různě velkých rohů. Pro obraz je nalezen

Laplacián Gaussiánu s hodnotou σ, jejíž velikost vypovídá o velikosti rohů. Poté se přejde k lokalizaci klíčových bodů. Body nalezené v prvním bodě se musí protřídit, aby se vyřadily hrany či nepřesnosti. K tomu se používá rozvoj Taylorovou řadou a Harrisův detektor rohů. Každému bodu, který touto filtrací projde se přiřadí orientace výpočtem směru a velikosti gradientu jeho okolí. Tím se zajistí invariance vůči rotaci obrazu. Dalším krokem je vytvoření samotného deskriptoru. Ten je ve formě vektoru o délce 128 hodnot reprezentujících hodnoty histogramu orientace okolí bodu. Posledním krokem je nalezení shod klíčových bodů. Deskriptory bodů jsou porovnávány a podle metody nejbližších sousedů jsou hledány nejlepší shody.

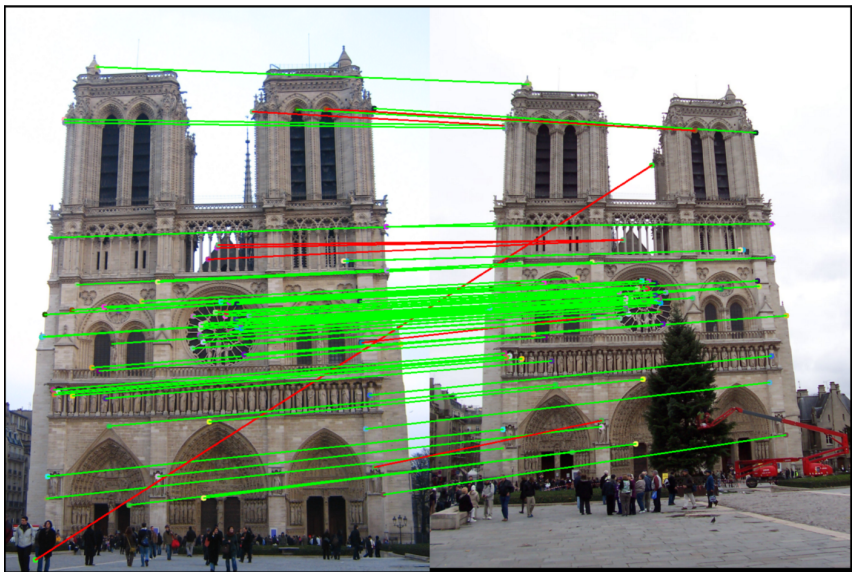
- **SURF** [8] je v porovnání se SIFTem rychlejší, protože pro aproximaci Laplaciánu Gaussiánu používá čtvercového filtru, který algoritmus urychluje. Čtvercový filtr je volen proto, že je jeho konvoluce snadno vypočitatelná pomocí integrálních obrazů. Vzorec integrálního obrazu má tvar: $I_\Sigma(x,y) = \sum_{i=0}^{i\leq x}\sum_{j=0}^{j\leq x} I(i,j)$. Pomocí integrálních obrazů je také možné snadno určit vlnové odezvy okolí, využívané pro určení orientace, ve velkém i malém měřítku. Základní verze SURF deskriptoru má 64 hodnot popisující příznaky daného bodu. Existuje však i verze vracející 128 hodnot. Příznaky zaznamenané ve vektorech představují vypočtené horizontální a vertikální odezvy. SURF je výrazně rychlejší než SIFT i přes minimální zvýšení výpočetní náročnosti. Je vhodný při zpracování obrazů obsahující rozmazání, ale změna úhlu pohledu a osvětlení může působit problémy.

- **ORB** [9] je kombinací FAST detektoru a BRIEF deskriptoru. Tento algoritmus je blíže popsán v kapitole **3.5.4**. Jedná se o metodu rychlejší než SIFT i SURF a je základem pro ORB-SLAM, který je popsán ve stejné kapitole.

- **KAZE** [10] je detektor využívající nelineární difúzi. Je schopný zachovat důležité detaily obrazu a odstranit šum při práci v prostorech s nelineárním měřítkem. Algoritmus je založen na škálově normalizovaném determinantu Hessiánu. Využívá pohyblivé okénko pro výpočet maxima odezvy detektoru. Významné body a jejich deskriptory jsou invariantní vůči měřítku i rotaci. V různých měřítkách mají lepší rozlišitelnost, ale jsou časově mírně náročnější. Existuje i urychlený KAZE algoritmus, AKAZE, který se liší tím, že využívá rychlou explicitní difúzi. Kvalita rotační invariance je zvýšena použitím Scharrových filtrů. Výsledky jsou srovnatelné, AKAZE je však výrazně rychlejší.

- **LBP** (Local Binary Patterns) [11] neboli lokální binární vzory jsou deskriptory textury. Počítají lokální reprezentaci textury porovnáváním pixelu se všemi v jeho okolí. Metoda převede obraz do odstínů šedé, vezme osmi-okolí pixelu, jeho intenzitu zaznamená a okolí porovná s prahem. Pixely, které mají vyšší hodnotu, než práh, jsou označeny nulou a zbytek jedničkou. Poté se provádí binární testy okolí, které jsou uloženy do osmi-bitového pole a převedeny do desítkové soustavy. Takto získané hodnoty se ukládají do 2D pole. Posledním krokem je pak výpočet histogramu, který umožňuje získat kód histogramu a finální příznakový vektor.

Trackování je úloha využívající nalezených významných bodů a jejich deskriptorů. Pro orientaci v prostoru je nutné mít stále informaci o pozici konkrétních bodů napříč jednotlivými snímky videa. Proto je nutné porovnávat body v po sobě jdoucích obrazech a hledat shody, viz Obrázek 7. K tomu slouží úloha párování významných bodů.



Obrázek 7: Ukázka párování bodů napříč obrazy

Na Obrázku 7 je možné vidět ukázku párování významných bodů napříč dvěma obrazy. Zelené čáry značí dvojice bodů, které byly spárovány správně a červené jsou chybná propojení. Následují příklady algoritmů, které tento problém řeší:

- **Metoda nejbližšího souseda** porovnává deskriptivní vektory získané pomocí algoritmů zmíněných výše. Hledá mezi nimi nejlepší shodu. Deskriptory se porovnávají podle $arg\,min\|x_i - x_j\|$. Jedná se o velmi přesnou metodu, která označí bod s nejlepší shodou za nejbližšího souseda. Existují však výrazně rychlejší metody, ovšem rychlost může být na úkor přesnosti.

- **RANSAC algoritmus** (RAndom SAmple Consensus) [12] slouží k odhadu parametrů modelu za účelem odstranění outlierů - bodů, které jsou velmi odlišné od ostatních dat v systému, mohly se do databáze dostat například chybou měření či odhadu. Jedná se o iterační algoritmus, který se opakuje, dokud není nalezen dostatek inliers - bodů dobře vyhovujících modelu.

  Kroky algoritmu jsou následující. Nejprve se náhodně zvolí N dat z datasetu. Poté se odhadnou parametry, aby model co nejlépe odpovídal zvoleným bodům. Data, která spadají do zvoleného okolí modelu jsou označeny jako inliery. Je-li jich dostatek, algoritmus končí. Pokud jich není dost, první tři kroky se opakují, dokud se buď nedojde ke zdárnému konci, nebo neproběhne hraniční počet opakování bez uspokojivého výsledku. V takovém případě algoritmus selhal.

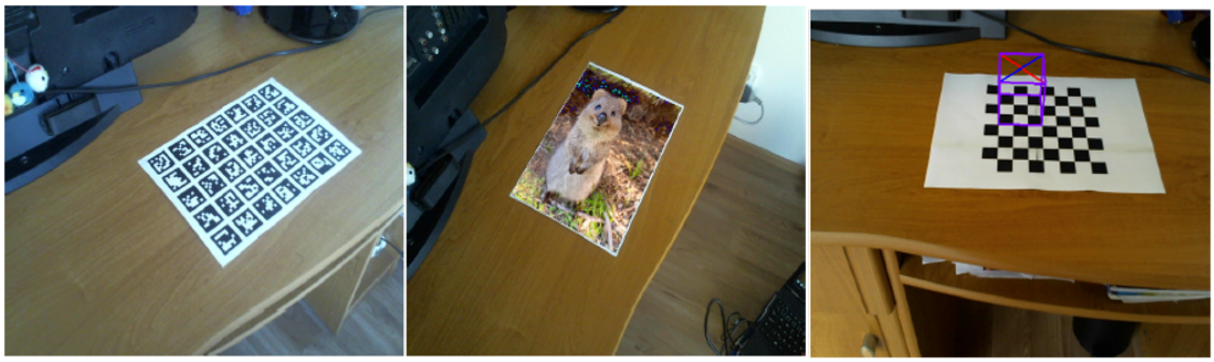
- **PROSAC** (PROgressive SAmple Consensus) [13] je další z metod užívaných k párování, ale na rozdíl od RANSACu nevybírá náhodné vzorky z datasetu, čímž proces urychluje. PROSAC využívá lineární uspořádání definované na množině korespondencí pomocí funkce podobnosti používané při stanovení předběžných shod. Vzorky jsou pak vybírány z postupně větších sad korespondencí s nejvyšším hodnocením. Podle provedených experimentů je někdy i stokrát rychlejší než RANSAC.

V případě značkové AR není třeba porovnávat všechny významné body navzájem. Stačí v obraze nalézt předem dané značky, které mohou mít různé podoby. Využívají se

například výrazné body nalepené či vložené do prostředí, aruco kódy nebo šachovnicové vzory. Bezznačková AR se však musí obejít bez tohoto ulehčení, což dělá úlohu trackování výrazně náročnější.

2.3.3 Návrh virtuálního objektu

Typů virtuálních objektů je nepřeberné množství, mohou být 2D, 3D, jednoduché či nahrazující velikou část reálného prostředí, atd. Volba daných objektů záleží na konkrétní aplikaci AR. Pro některé anotační úlohy či úlohy orientace stačí do prostoru vložit samotný nápis nebo třeba jen 2D obraz. Jindy je však nutné navrhnout i velmi přesné 3D modely objektů, například v případě medicíny či opravy nebo výroby strojů, kde je vysoká přesnost zásadní.



Obrázek 8: Ukázky virtuálních objektů v reálném prostředí

Existuje velká spousta možností, jak objekty navrhovat. V nejjednodušších případech - pro 2D modely - stačí najít nebo vytvořit potřebný obrázek. Pro výrobu 3D modelů pak ve snadnějším případě, jako například na Obrázku 8, stačí použít linky či plochy vytvořené přímo v kódu algoritmu. Pro složitější modely pak existují platformy jako například OpenGL[2] nebo Unity[3], které jsou přímo vytvořeny za účelem 3D modelování, ať už pro tvorbu her, objektů, animací či pro jiná využití.

2.3.4 Registrace

Registrace je pro AR velmi důležitou úlohou. Jedná se o jeden ze základních problémů limitujících AR. Pro uživatele je zásadní, aby bylo reálné a virtuální správně srovnáno. Přesným umístěním objektů je možné docílit toho, že AR působí přirozeně. Jindy je přesnost přímo vyžadována, jak již bylo zmíněno výše. Aby registrace fungovala správně, je nutné předcházet statickým a dynamickým chybám. Mezi statické, které se projevují, když je uživatel i prostředí v klidu, se řadí optické zkreslení, chyby sledovacího systému, mechanické posunutí a nesprávné sledovací parametry. Dynamické se pak ukazují, když je prostředí, uživatel nebo obojí v pohybu, a jsou způsobeny systémovým zpožděním či zaostáváním.

Optické zkreslení je způsobeno čočkou kamery a optikou používanou k zobrazování AR. Větší problémy s ním mají hlavně širokoúhlé displaye. Většinou se jedná o systémové chyby, které se dají při správném zmapování kompenzovat dodatečnou optikou nebo digitálně.

[2]https://www.opengl.org/
[3]https://unity.com/

Chyby sledovacího systému jsou obsaženy přímo ve výstupu ze sledovacího systému. Většinou však nejsou systematické a je velmi složité je řešit. Je k tomu potřeba 3D pravítka přesnějšího než testovaný sledovač.

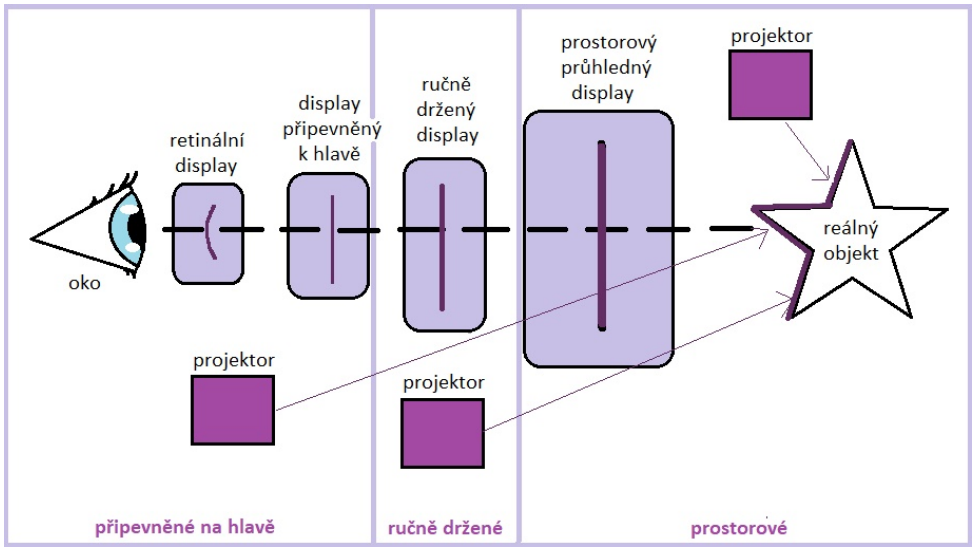
Mechanické posunutí zahrnuje chyby způsobené neshodou mezi modelem a reálnými fyzikálními modely systému. Tyto problémy lze jen složitě kompenzovat. Pokud je to možné, je lepší rovnou opravit a předělat model.

Nesprávné sledovací parametry jsou dalším významným problémem, který se vyskytuje při sledování polohy kamery nebo hlavy. Lze však řešit přeměřováním parametrů nebo kalibrací.

U dynamických chyb jde především o takzvané end-to-end zpoždění, což je rozdíl mezi okamžikem, kdy je měřena poloha a orientace kamery, a okamžikem, kdy se obrazy generované na základě naměřených hodnot promítnou. Každá část systému totiž potřebuje k vykonání své úlohy a potřebným výpočtům nějaký čas. Ke snižování těchto chyb se používají čtyři hlavní typy metod a to redukce systémových zpoždění, redukce zjevných zpoždění, shodnost časových toků a předvídání budoucích lokací.

2.3.5 Displaye

Důležitou volbou pro implementaci AR systému je i způsob zobrazování. Existují prostředky, které slouží pro virtuální rozšíření dalších smyslů kromě zraku. AR může upravovat i sluchové, hmatové či méně rozšířené čichové nebo chuťové vjemy. Tato práce se však věnuje pouze vizuální AR. Následující odstavce popisují různé druhy zobrazovacích technik [14], které lze využít. Na Obrázku 9 je možné vidět jednoduché schéma a prostorové rozložení několika přístupů.



Obrázek 9: Vizuální zobrazovací techniky

Existují tři hlavní přístupy k zobrazování virtuálních objektů do reálného prostředí. Prvními dvěma jsou optické a video technologie, které jsou popsané v kapitole **2.1**. Třetím způsobem je vkládat virtuální objekty přímo na reálné předměty, k čemuž slouží projektory. Tím vzniká třetí typ a to projektivní displaye. Nevyžadují žádné přístroje, které uživatel musí nosit na hlavě nebo v ruce a mohou pokrýt výrazně větší plochu. Většinou jsou však limitované na užití uvnitř budov kvůli problémům s nedostatečným jasem promítaných objektů.

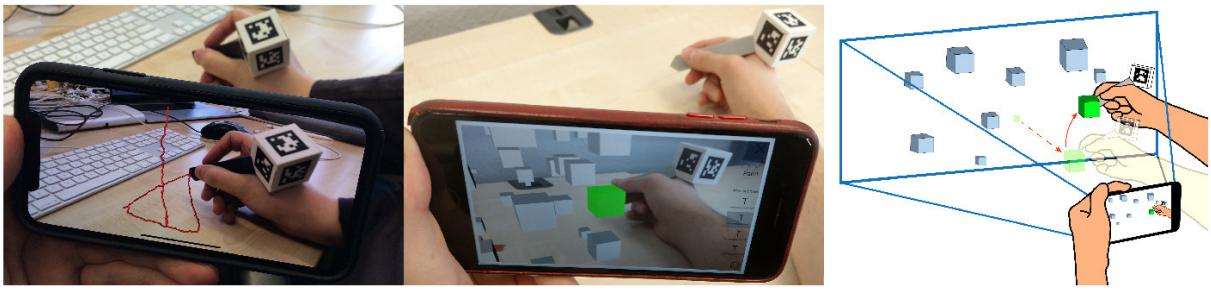
Dalším dělením vizuálních displayů je způsob umístění. Existují ty, které se připevní k hlavě, držené v ruce a nakonec prostorové. Do první kategorie spadají například optické nebo video průhledné dispaye na hlavu (optical/video see-through head-mounted displays - HMD), virtuální retinální displaye (VRD) a k hlavě přidělané projektivní displaye (HMPD). Tyto typy potřebují nějaké spojení s počítačem, který vykonává výpočty. Je ovšem možné provádět výpočty dálkově a propojit se se zobrazovacím zařízením například pomocí BlueTooth nebo jiných technologií.

Do druhé kategorie patří například optické či video průhledné displaye držené v ruce, nebo ruční projektory. Výhodou těchto technologií je levnější výrobní cena v porovnání s předchozí kategorií a snadné použití. Uživatel může AR využívat například pomocí obrazovky svého mobilního telefonu.

Poslední kategorie pak zahrnuje obrazovkově založené video průhledné displaye, prostorové optické průhledné displaye nebo displaye projektivní. Ty jsou využívány především v případě velkých ploch pro více uživatelů s omezenou interakcí.

2.3.6 Interakce

Interakce či manipulace s některým z virtuálních objektů je častá úloha řešená v AR. Umožní tak uživateli větší propojení s virtuálním světem. V mnoha aplikacích dokonce existuje nutnost určitého typu interakce. Někdy stačí označit nějaký z objektů, jindy je jím třeba pohnout, otočit ho či měnit jeho velikost a někdy se ho stačí dotknout, aby se spustila některá z naprogramovaných akcí. Na Obrázku 10 je možné vidět ukázku manipulace s virtuálním objektem.



Obrázek 10: Manipulace s virtuálním objektem

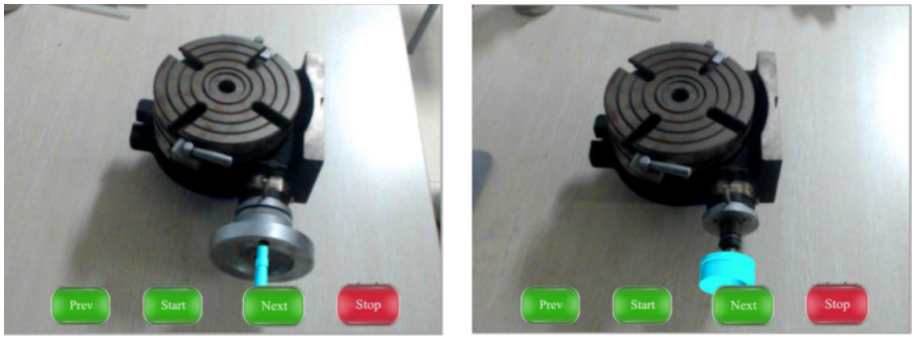
Jednou z možností je reagovat přímo na akci kliknutí myši, zobrazuje-li se AR na obrazovce počítače. Podobný princip platí i v případě mobilního telefonu nebo tabletu, kde objekt reaguje na dotyk.

Dalším typem je systém sledující přímo uživatelovu ruku [15]. Identifikuje ruku v prostoru a sleduje její pohyb, nebo třeba jen pohyb prstů. Virtuální objekt poté reaguje na polohu ruky. Podobného výsledku je možné dosáhnout i bez využití automatického vizuálního rozpoznávání a to přístrojem, který se na ruku nasadí a monitoruje pohyb prstů přímo fyzicky.

Existuje ještě několik způsobů, jak s AR interagovat. Možností je mnoho, jako například složitější úlohy sledování úhlu pohledu uživatelových očí, atd. V principu jde však vždy o to, že systém čeká na daný signál a poté spustí požadovanou akci virtuálního objektu. Spouštěč těchto akcí závisí na konkrétní implementaci a potřebách či možnostech dané aplikace.

2.4 Příklady existujících bezznačkových AR systémů

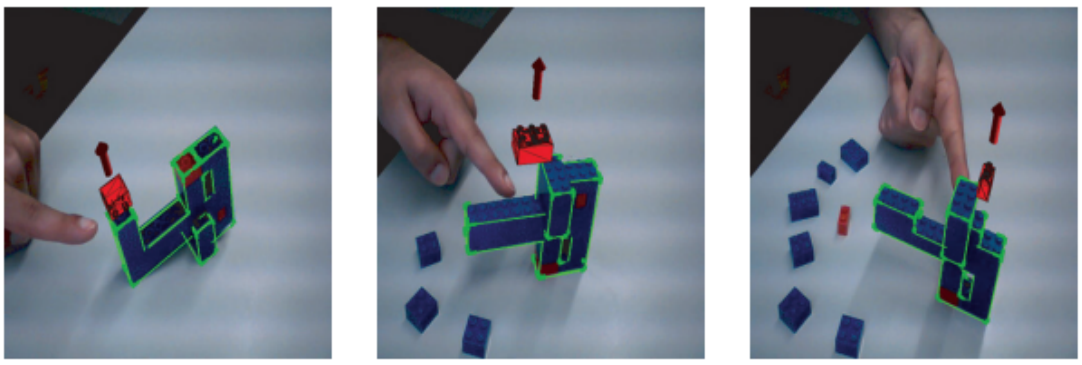
Autoři Wang a spol ve své práci **Mechanical assembly assistance using marker-less augmented reality system** [16] představili systém bezznačkové rozšířené reality využívaný jako pomoc při mechanické montáži. Systém se automaticky přizpůsobuje požadavkům na sledování v momentě, kdy se po provedení montážních kroků změní topologická struktura soustavy. Systém poskytuje asistenci v podobě zobrazování virtuálních objektů představujících reálné díly potřebné při montáži, viz Obrázek 11. Ukazuje je na místech, kam patří ty skutečné, i se způsobem jejich připojení.



Obrázek 11: Ukázka mechanické montáže doplněné o AR

Tento systém, jako velká část následujících, je rozdělen do dvou částí a to ofline příprav a online provedení. V přípravné fázi jsou nejprve získány referenční multi-obrazy za pomoci virtuální kamery a CAD softwaru. Pro hledání shod se pak využívá modifikovaný LINE-MOD, kterému je přidána větší rotační invariance. K získání příznaků je zvolen ORB. V online fázi se pak počítá pozice kamery, která se nejprve odhaduje a pak upřesňuje. Pro odstranění nežádoucích outlierů z dat slouží RANSAC algoritmus. Vizualizace je prováděna za pomoci enginu Unity3D.

Providing Guidance for Maintenance Operations Using Automatic Markerless Augmented Reality System [17] je práce Alvaréze a spol, ve které představují využití AR jako nástroje k usnadnění demontáže během údržby. Systém poskytuje pracovníkům detailní pokyny za účelem efektivnějšího provádění úkonů údržby. Systém obsahuje také modul pro plánování trasy. Konkrétní sekvence pro montáže nebo demontáže jsou odvozovány automaticky z 3D modelu objektu. Za využití těchto informací jsou pak generovány pokyny pro pracovníky, viz Obrázek 12.



Obrázek 12: Ukázka demontáže s AR pokyny

I tento systém je rozdělen na dvě části. Ve fázi příprav obdrží systém 3D model předmětu určeného k údržbě. Ten obsahuje jednotlivé části reprezentované 3D sítí a jejich polohu v celkovém modelu. Na základě něj je určen postup montáže/demontáže. Pro potřebu trackování jsou vyhledány orientační významné body. V online fázi se pak počítá pozice a na reálný objekt jsou vizualizovány instrukce, viz Obrázek 12. Uživatel poté stisknutím tlačítka upozorní systém, že krok byl proveden, a objeví se následující instrukce z generovaného pořadí.

Ve své práci **Markerless Augmented Reality with Light Source Estimation for Direct Illumination** [18] Frahm a spol představuje AR systém, který nejen zobrazuje virtuální objekty, ale navíc hledá zdroje světla v obraze a přizpůsobuje jim osvětlení modelu pro dosažení větší realističnosti, viz Obrázek 13. Systém najde využití v televizní produkci. K analýze scény a jejího okolí a lokalizaci světelných zdrojů jsou použity dvě kamery. Televizní snímá obraz prostředí určený ke zpracování a fish-eye kamera se širokým zorným polep pak pořizuje záznam stropu, kde jsou vyhledávány světelné zdroje.

Obrázek 13: Ukázka virtuálního objektu doplněného o osvětlení

Systém se opět dělí na offline část, kde jsou získány 3D příznaky ze záznamů obou kamer, a online část, která provádí trackování a odhad pozice. K analýze snímků je využit přístup struktury z pohybu (structure-from-motion). Poté se hledají shody mezi snímky za použití KLT-trackeru a k upřesnění slouží RANSAC algoritmus. V druhé fázi probíhá počáteční lokalizace první kamery a dále se sleduje poloha kamery na základě předem získaných 3D významných bodů. Informace z obou kamer se musí zarovnat a poté se přechází do bodu registrace a vizualizace virtuálního objektu za použití hloubkové mapy. Na základě dat z fish-eye kamery, jejíž úkol je snímat zdroje světla, je pak na model přidáno osvětlení tak, jak by byl nejspíš osvícen reálný objekt.

Sato a spol představují ve své práci **A Marker-less Augmented Reality System Using Image Processing Techniques for Architecture and Urban Evironment** [19] systém využívající AR za účelem použití při venkovních rekonstrukcích a projektech údržby budov. Navrhovaný systém využívá technologii registrace obrazu založenou na příznacích a metodách struktury z pohybu (Structure from Motion, SfM), která konstruuje 3D modely pomocí fotografií z více úhlů pohledu. AR může sloužit k vizualizaci projektů budov v plném měřítku ještě ve fázi návrhu před výstavbou. Lze ji také použít k vizualizaci některých částí budovy pro účely údržby a provozu, čímž se práce Satoa zabývá. Na Obrázku 14 je možné vidět příklad vložení virtuálních markýz na reálnou budovu.

Obrázek 14: Ukázka promítnutí virtuálních markýz na reálné budově

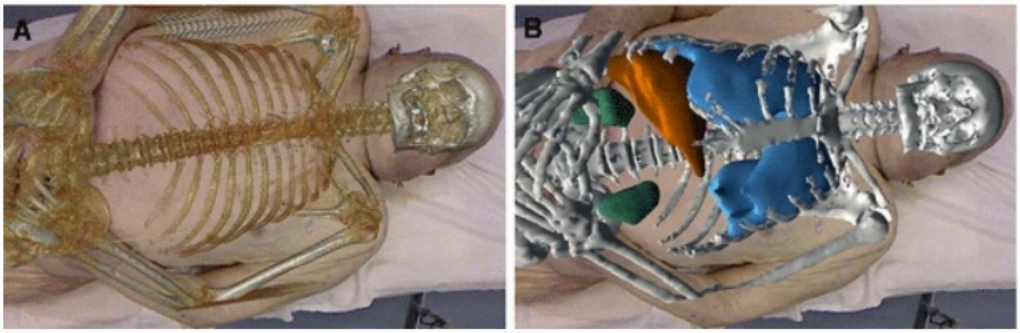
Ve fázi předzpracování je pomocí SfM metody ze snímků získaných z několika úhlů pohledu vytvořen 3D model budovy. Informace o pozici a orientaci všech úhlů pohledu je uchována. Poté se určí souřadnice 3D virtuálního objektu určeného k augmentaci vzhledem k souřadnicím modelu prostředí. Nakonec se uloží příznaky nalezené v prostředí pomocí SURF deskriptoru. Ve fázi zpracování probíhající v reálném čase se pak importují předem získané informace a ze snímků videa se opět za použití SURFu extrahují důležité příznaky. Ty jsou porovnávány s modelem a nakonec jsou na display do videa promítnuty virtuální objekty.

V práci **Cultural Heritage in Marker-Less Augmented Reality: A Survey** [20] provádí Kolivand a spol rešerši systémů schopných vizualizovat kulturní dědictví ve virtuální formě. Podobných systémů existuje mnoho s různým rozsahem pro zobrazování různě velkých objektů, uvnitř budov nebo venku v otevřeném prostoru. Slouží k účelům výuky nebo jen pro představu, jak některé stavby nebo artefakty vypadaly. Systémy jsou například schopny do prostředí, ať už vnitřního nebo venkovního, promítat 3D modely budov vytvořené z obrazů a fotografií, viz Obrázek 15.

Obrázek 15: Ukázka promítnutí virtuální budovy ro reálného prostředí

V práci jsou zmíněny a porovnány různé systémy, pracující v reálném čase či nikoliv, vizualizujících přímo na mobilních telefonech nebo jiných zobrazovacích zařízeních, atd.

Kilgus a spol v práci **Mobile markerless augmented reality and its application in forensic medicine** [21] představují systém sloužící jako pomoc v medicínském odvětví. Umožňuje patologům v průběhu pitvy sledovat data získaná z CT za pomoci tabletu nebo počítače. Toto usnadňuje analýzu a práci s tělem, protože patolog má možnost například přiřadit vnější rány k vnitřním zraněním ještě před otevřením těla. Na Obrázku 16 je možné vidět ukázku z tohoto systému.



Obrázek 16: Ukázka promítnutí části anatomie na tělo zesnulého

Kamera snímá hloubku obrazu a barvu těla. Server odhaduje polohu kamery na základě povrchové registrace CT a hloubkových dat, čímž umožňuje vizualizaci anatomie přímo na tabletu. Prvním krokem je tedy provedení CT, dále segmentace informací z tohoto vyšetření, pak umístění kamery a nakonec vizualizace virtuálních dat v tabletu přímo na tělo zemřelého.

Existuje i spousta systémů, která se zaměřuje například na učení nebo zábavu. Huang ve své práci **Piano AR: A Markerless Augmented Reality Based Piano Teaching System** [22] představuje pomocníka při učení se na piáno. Místo markerů vyhledává systém v obraze geometrii kláves a poté na ně promítá virtuální prsty, které má uživatel sledovat, viz Obrázek 17. Projekt je programován v C++ za použití OpenCV.

Obrázek 17: Ukázka systému podporujícího výuku na klávesy

Systém se skládá ze čtyř hlavních modulů. Prvním je příprava, kde se získají parametry kamery a geometrie reálných kláves, popřípadě další fyzikální vlastnosti. Po snímání obrazu se zbylé tři moduly opakují ve smyčce. Prvním je rozpoznávání a trackování, obsahující předzpracování obrazu, extrakci a analýzu kontur, rozpoznání kláves a nakonec získávání geometrických příznaků. Poté následuje modul odhadu pozice, který obsahuje výpočty matic rotace a translace spolu s určením 3D souřadnic. Posledním modulem je pak kompozice scény, kde se promítá virtuální model na klávesy.

2.5 Přístup k AR v této práci

Pro kalibraci kamery jsem v tomto případě využila přímo kódu ze stránek OpenCV[4]. Jeho pomocí jsem získala parametry kamery a ty budu využívat pro správný chod zbytku programu.

Pro tvorbu 3D modelů jsem zvolila platformu Vectary[5]. Ta umožňuje tvorbu 3D objektu a jeho import v různých formátech. Lze využít některé šablony, nebo si vytvořit vlastní model. Do projektu je pak možné přidat i barvy, textury nebo dokonce animaci.

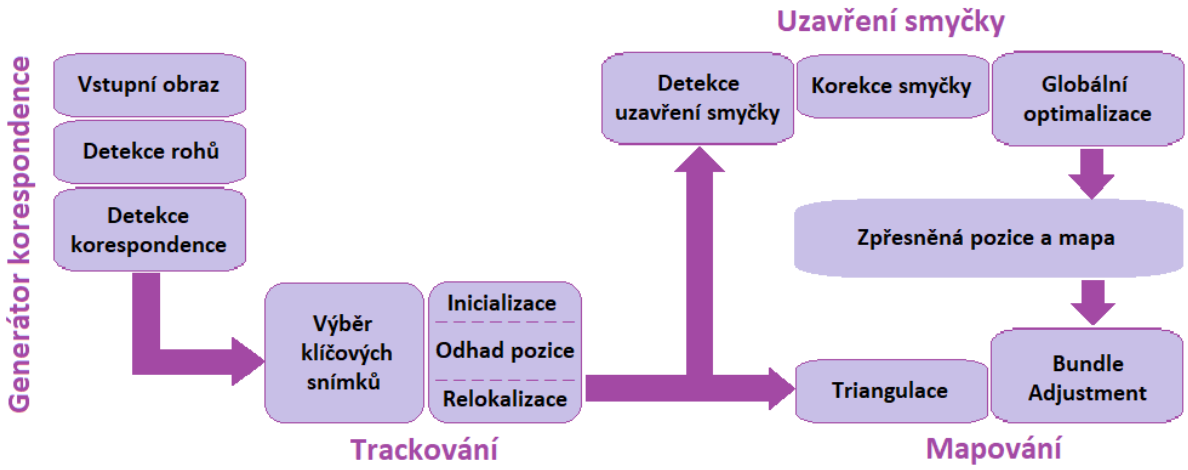
Jako display jsem zvolila plochu počítače, která později slouží i jako rozhraní pro interakci uživatele s virtuálním objektem. Objekt reaguje na akci kliknutí myši, načež se vymění s druhým modelem. Bez této akce setrvává ve stejné pozici.

[4] https://learnopencv.com/camera-calibration-using-opencv/
[5] vectary.com

3 Současná lokalizace a mapování

Současná lokalizace a mapování neboli Simultaneous Localization and Mapping (SLAM) [23] je označení procesu, kdy lokalizace a mapování probíhá současně. Lokalizací rozumíme určování polohy kamery a mapováním pak vytváření mapy. Výsledkem SLAMu bývá 3D reprezentace prostředí nebo mapa okolí, po kterém se sledovaný objekt pohybuje. Záleží na konkrétní řešené úloze. V případě této práce bude SLAM využíván převážně k pozorování pohybu kamery a jejím úhlu natočení - orientaci v prostoru.

SLAM obecně je často prováděn za použití jiných senzorů než jen kamer. Je možné použít například lasery nebo radar. V této práci se však bude psát konkrétně o jednom druhu SLAMu, kterým je vizuální SLAM (vSLAM). Jako senzor pro jeho fungování se využívá jedna nebo více kamer. Na Obrázku 18 je zobrazeno schéma principu SLAM systému.

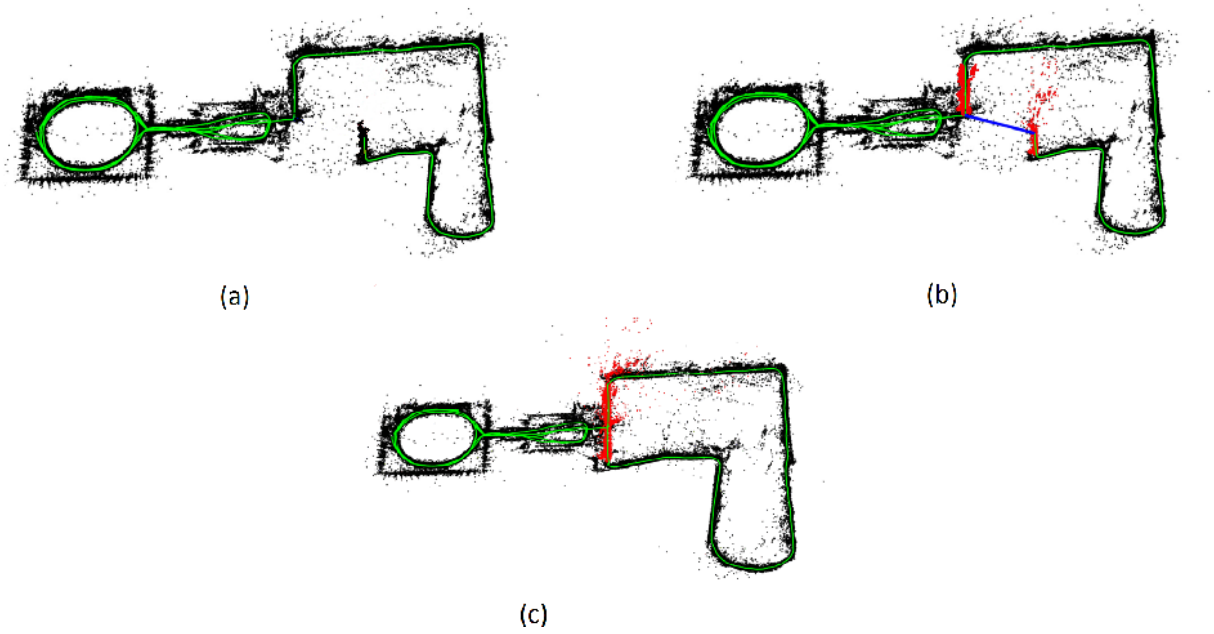


Obrázek 18: Schéma principu SLAM systému

Vizuální odometrii, zmíněnou v úvodu, je možné použít jako základ pro SLAM systém. VO, blíže popsaná v kapitole 3.2, je proces odhadu polohy a pohybu kamery nebo robota z vizuální dat v prostředí, které nemusí být předem známé. Stačí jí proud obrazů z kamery, který analyzuje a na jeho základě získává potřebné informace. V počátečním okamžiku se ocitá v konkrétním zvoleném bodě X_0 souřadnicového systému. Celkový odhad pohybu se skládá ze všech transformací T v časových okamžicích 0 až k, složených z translace t a rotace R, od tohoto počátečního bodu.

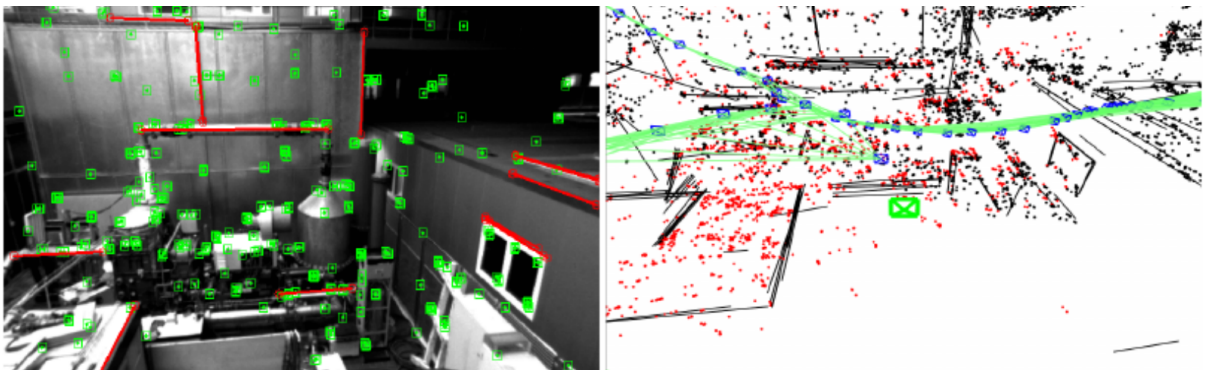
$$X_k = T_k \cdot T_{k-1} \cdot T_{k-2} \cdot ... \cdot T_0 \cdot X_0 \tag{1}$$

Pro SLAM systém však musí být VO rozšířena o vytváření digitální reprezentace prostoru. SLAM totiž narozdíl od VO nemapuje pouze trajektorii. Je proto výpočetně náročnější. Uchovává si v paměti celou cestu a na základě toho později upravuje tvar trajektorie sledovaného objektu. Když později rozpozná místo, na kterém se již nacházel, přepočítá mapu tak, aby se totožné reálné místo nacházelo ve stejném bodě i v reprezentaci prostředí. Uzavírá tak smyčku, zohledňuje konzistenci celé trajektorie. Na Obrázku 19 je uzavírání smyčky ilustrováno.



Obrázek 19: Uzavírání smyčky

Tato technologie se často využívá například pro ovládání a sledování autonomních robotů, sond, vozidel a jiných bezpilotních prostředků. Využívá se také v případě rozšířené reality. SLAM se používá k vytvoření digitalizované reprezentace okolního prostředí. To je nezbytná funkce pro všechny výše zmíněná využití. Jelikož je SLAM nezávislý na externím signálu z družic - stačí mu informace o prostředí, které získá ze senzorů - je možné jím v některých případech nahradit i GPS. Využití SLAMu by pak řešilo problém s lokalizací uvnitř budov nebo v tunelech. Na Obrázku 20 je příklad výstupu SLAM systému.



Obrázek 20: Příklad výstupu vizuálního SLAM

3.1 vSLAM

Vizuální SLAM (vSLAM) [24] jako hlavní zdroj získávání informací o okolním prostředí využívají jednu nebo více kamer. Na základě informací získaných z vizuálních senzorů pak dokáží určit orientaci a polohu kamery v prostoru. Cílem vSLAM systémů je lokalizovat pozici kamery a zmapovat své okolí. Často se využívají za účelem navigace. 3D reprezentace prostředí nebo mapa je vytvářena současně se zpracováváním obrazů, nečeká až budou všechna data zpracována.

Tento přístup má své výhody, díky kterým je tolik používaný. Jednou z nich je finanční dostupnost kamer jakožto používaných senzorů. Další výhodou je, že vSLAM systémy mají k dispozici přímo obrazy reálného prostředí, ne pouze informace, které by mohly poskytnout senzory jako jsou například radary nebo ty, které využívají lasery. S těmito daty je pak možné dále pracovat a využívat je k rozšíření úlohy z pouhé lokalizace a vytváření 3D reprezentace prostředí. Je možné využít nejen schopnost vnímat okolní objekty, ale popřípadě je i rozeznávat, což umožňuje spoustu dalších zajímavých aplikací tohoto systému.

Nevýhodami vSLAMu je pak především fakt, že mohou nastávat chyby v datech. Existuje spousta možných zdrojů těchto chyb, ale nejzávažnější jsou například nedostatečné rozlišení kamery, rozmazání obrazů zapříčiněné rychlým pohybem, nebo chyby způsobené náhlou změnou jasu v okolním prostředí.

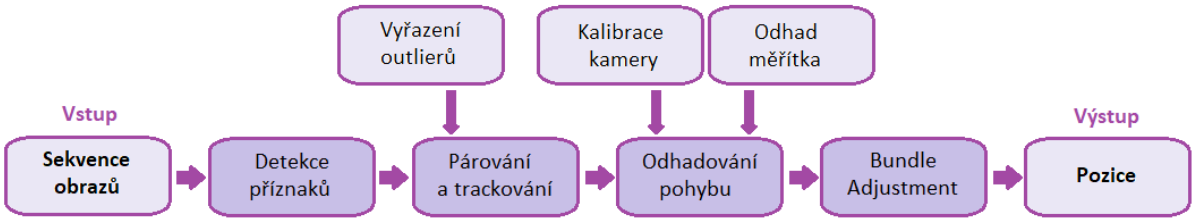
Algoritmy SLAM systémů se skládají z několika kroků [25] nezbytně nutných pro jejich správnou funkci. Tyto kroky budou detailněji popsány v kapitole **3.3**, níže je uvedeno pouze stručné shrnutí.

- **Inicializace** spočívá v počátečním nastavení globálních souřadnic a kalibraci kamery.
- **Lokalizace** je akce, která má za úkol určit a sledovat pozici kamery. Je pevně spjata s mapováním.
- **Mapování** vytváří na základě vizuální informace a informace o pozici kamery 3D reprezentaci prostředí.
- **Relokalizace** znamená sledování pohybu a upravování jeho odhadu.
- **Optimalizace globální mapy** má pak za úkol především uzavírat smyčku - upravovat mapu na základě její celkové konzistence.

3.2 Vizuální Odometrie

Klasická odometrie určovala polohu robotu za pomoci otáček kol, polohových senzorů a podobných zdrojů informací. To však přinášelo značné nedostatky a problémy. Aby bylo možné předejít problémům způsobeným například nerovnostmi terénu, začala se využívat vizuální odometrie. Ta zpracovává obraz z kamer, ve kterém hledá významné body. Některé aplikace nemusejí spoléhat na tyto body a na základě přímých metod pracují přímo s intenzitou obrazu.

Algoritmus vizuální odometrie se skládá z několika kroků. V daných bodech existují rozdíly při použití jedné (mono) a dvou (stereo) kamer. Další odstavce jsou však jen stručné shrnutí principu VO, které se těmito rozdíly detailně nezabývá. Na schématu na Obrázku 21 je znázorněn průběh algoritmu VO.



Obrázek 21: Princip VO

- **Detekce a popis významných bodů**

  V obrazu je provedena detekce významných bodů a snadno odlišitelných míst s různou intenzitou, barvou a texturou vůči svému okolí. Takto nalezeným významným bodům, které by měly být robustní a invariantní vůči geometrickým a fotometrickým změnám, jsou přiřazeny deskriptory.

- **Párování významných bodů**

  Na základě nalezených bodů a deskriptorů se přechází ke kroku párování napříč dvojicemi obrazů. Deskriptory bodů v původním obraze se porovnávají s těmi v obraze novém, buď po celé ploše, nebo jen v daném okolí - záleží na nastavení. V případě nalezení více shod se provádí vzájemná kontrola konzistence, kdy se pro spárování vybere bod s největší normalizovanou korelací s prvkem v původním obraze.

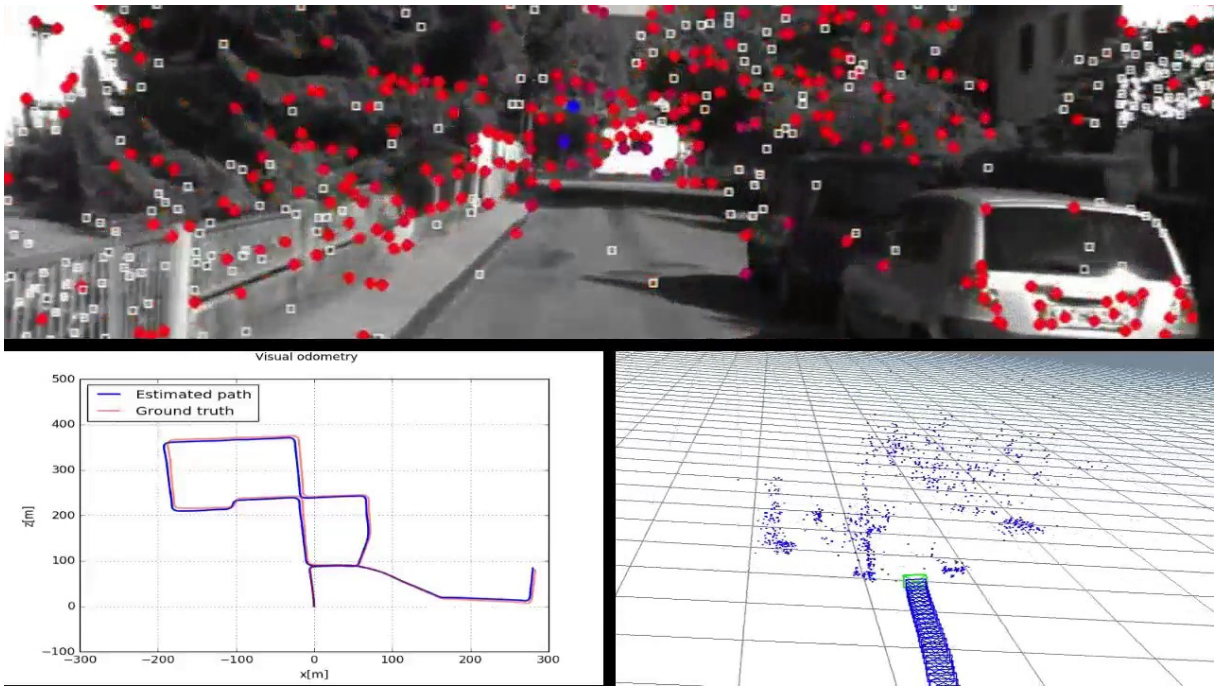
- **Trackování významných bodů**

  V dalším kroku se významné body sledují napříč všemi obrazy, přičemž se odhaduje pozice kamery. V případech monokulárního i stereo systému se často používá například RANSAC [12] algoritmus. Body se sledují napříč po sobě jdoucími obrazy a triangulací se získává jejich 3D pozice. U monokulárního systému chybí 3D informace z dvojice kamer (na rozdíl od stereo VO), což vede k nutnosti výpočtu relativní pozice ze třech po sobě jdoucích obrazů. V případě stereo systému je úloha snazší, protože je možné provádět triangulaci a pozicování pomocí dvou kamer. Vypadne tedy problém výpočtu relativní pozice.

- **Bundle Adjustment**

  Jedná se o optimalizaci zahrnující minimalizaci chyby reprojekce mezi sledovanými a předvídanými body v obraze. Matematicky se jedná o součet čtverců velkého počtu nelineárních funkcí s reálnou hodnotou. Optimalizace tedy využívá nelineární metodu nejmenších čtverců a zpřesňuje odhad pozice.

Na Obrázku 22 je ukázka VO. Zobrazuje nalezené body na vstupním obrázku, trajektorii získanou pomocí VO (modře) v porovnání se skutečnou trasou (červeně) a nakonec bližší pohled na odhady polohy kamery a rozmístění významných bodů v prostoru.



Obrázek 22: Ukázka VO

3.2.1 Metody založené na geometrii

Tyto metody [26] jsou v praxi zatím běžněji používané. Odhad polohy se řídí podle geometrické informace získané ze vstupních obrazů. Tato kategorie se může dále dělit na metody s řídkými (sparse) příznaky a metody přímé.

Druhá skupina metod k určení odhadu pohybu využívá geometrii vstupních obrázků, díky které extrahuje, porovnává a sleduje významné body ze sekvence snímků. Kvůli přítomnosti šumů, outlierů a jiných rušivých elementů však tyto algoritmy trpí kumulací chyb, které se postupem času mohou stát velmi výraznými.

Přímé metody se vyznačují menší výpočetní náročností. Jsou schopny pro odhad pozice využít všechny pixely v po sobě jdoucích snímcích za předpokladu fotometrické konzistence. Tyto metody bývají přesnější a dokáží lépe pracovat v prostředí bez výrazné textury.

3.2.2 Metody založené na učení

Tyto metody se snaží naučit pohybový model a odvozují VO ze snímačů pomocí technik strojového učení bez explicitního použití geometrické teorie. Jedné se o přístup založený na datech. Optický tok se používá k trénování regresních algoritmů. Běžné techniky strojového učení nejsou dostatečně efektivní, když se setkají například s RGB obrázkem, který obsahuje velká nebo vysoce nelineární mnohorozměrná data. Proto se přechází k využití takzvaného deep-learningu, neboli hlubokého učení. To dokáže učením automaticky nalézt vhodnou reprezentaci prvků z rozsáhlé datové sady, čímž umožňuje užití těchto metod ve VO.

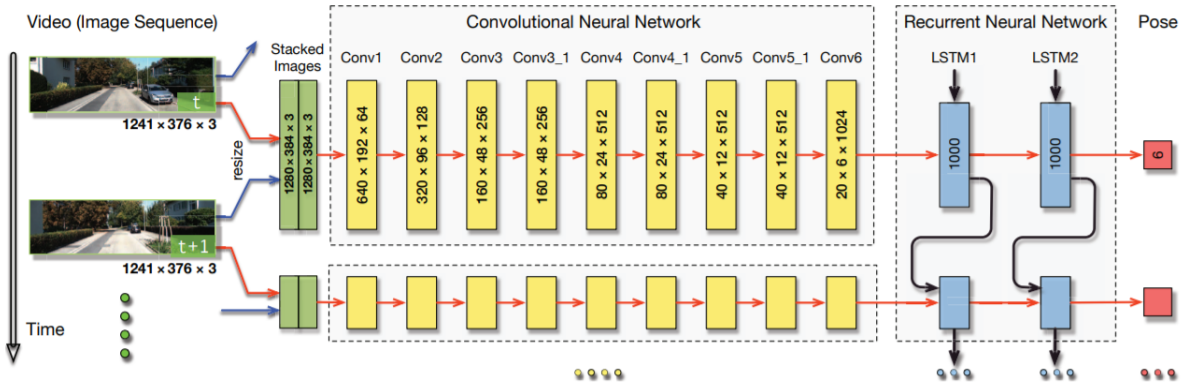
Hluboké učení se dále může dělit na tři podkategorie [27]: s dohledem učitele (Supervised), částečně s dohledem (Semi-Supervised) a bez učitele (Unsupervised). Hlavním rozdílem mezi nimi je dostupnost skutečného, požadovaného výsledku, takzvané ground truth. Metody s s učitelem znají ground truth, učitel jim ji poskytne. Je však obtížné

získat dostatečně kvalitní předem označené soubory dat. Učení bez dozoru se pokouší automaticky najít strukturu v datech extrahováním užitečných příznaků bez jasných a konkrétních pokynů ohledně požadovaných výsledků. Používají se k tomu například metody shlukování příznaků. Mezi těmito dvěma extrémy je kategorie takzvaného polořízeného učení. U tohoto typu učení se trénovací datové sady obvykle skládají z označených i neoznačených dat, odkud se neuronové sítě mohou učit.

3.2.3 Vizuální odometrie a neuronové sítě

V posledních letech se objevují systémy a přístupy, které řeší problém VO pomocí hlubokého strojového učení, respektive pomocí neuronových sítí [28]. Neuronové sítě jsou v tomto přístupu používány k nahrazení deskriptorů. Jsou schopny extrahovat velké množství různých abstraktních příznaků z obrazů, kterými nahrazují geometrické deskriptory příznaků. Na základě těchto informací je pak vytvořena transformační matice mezi dvěma obrazy, jež se využívá k výpočtu trajektorie.

Využívá se k tomu například hlubokých rekurentních konvolučních neuronových sítí (Reccurent Convolutional Neural Network - RCNN), které kombinují více základních typů sítí, které jsou blíže popsány v Kapitole 3.6. Jak už název napovídá, propojují konvoluční (CNN) a rekurentní (RNN) síť. CNN je využívána pro extrakci příznaků z dvojic RGB obrazů. Tyto obrazy pak shrne do kompaktního popisu, který dále pokračuje do RNN. Ta slouží k sekvenčnímu učení modelu, který určuje dynamiku a vztahy mezi příznaky získanými z CNN.



Obrázek 23: Architektura RCNN pro VO

Na Obrázku 23 je možné vidět architekturu sítě pro VO. Jako vstup slouží obrazy z kamer. Dvojice obrazů poté putují do CNN, která má v tomto případě devět vrstev. Síť dále přechází do RNN, ze které informace putují jak do výstupního odhadu polohy, tak i do části RNN zpracovávající následující dvojici obrazů.

3.3 Obecný princip SLAM systému

Následující odstavce blíže popíší jednotlivé kroky SLAM systému, zmíněné v Kapitole 3.1 a vysvětlí jejich princip.

3.3.1 Inicializace

Inicializace je prvním nezbytným krokem pro fungování SLAM i VO systému. Do tohoto bodu je zahrnuta kalibrace kamery, kterou je nutno provést, aby systém nadále dostával přesné informace a mohl správně zpracovávat obrazy, jež z kamery proudí. Další nezbytností je pak definování souřadnicového systému, který bude pro odhad pozice i pohybu kamery dále používán. V tomto bodě se také vytvoří počáteční mapa okolí.

3.3.2 Lokalizace

Mapování a lokalizace spolu úzce souvisí, přestože se na tyto dva úkoly dříve pohlíželo zvlášť. V obrazech okolního prostředí se vyhledávají významné body za užití detektorů a deskriptorů popsaných v kapitole **2.3.2**. Poté se provede jejich párování a trackování. Na základě toho se určuje korespondence mezi obrazy a vytvářenou mapou, pomocí čehož se určuje pozice kamery.

3.3.3 Mapování

Při mapování pak vzniká mapa pomocí výpočtů 3D struktury neznámého prostředí. Výstupem SLAM systémů může být více typů map. Dělí se na 2D a 3D mapy a dále na metrické a topologické. Metrické mapy vznikají triangulací významných 2D bodů za vzniku 3D mapy. Tato úloha se však s rostoucím prostředím stává příliš složitá a dochází k selhání systémů. Proto vznikl jiný typ, topologické mapy. Tento přístup odbourává potřebu tolika geometrických informací jako jsou vzdálenost, měřítko a směr, a místo toho prostředí reprezentuje neuspořádaným grafem uzlů - pozicový graf. Ten představuje klíčové snímky spojené hranami, pokud toto spojení existuje. Klíčovými snímky (KS) rozumíme vybrané snímky, které nadále reprezentují paměť - ostatní se vypouští. Tím se docílí výrazného snížení výpočetní náročnosti a ušetření času. Volba klíčových snímků záleží na konkrétí aplikaci. Jelikož je však ve většině případech nutné postihnout větší plochu, ale zároveň kvůli odhadu polohy kamery mít stále nějaký přehled o metrické informaci, začaly se používat takzvané hybridní mapy - kombinace dvou výše zmiňovaných.

Jak už bylo řečeno, mapování a lokalizace jsou spolu spjaty a mapa se neustále upravuje na základě odhadu polohy. Takto pozměněná mapa pak naopak hraje roli v novějších odhadech polohy.

3.3.4 Relokalizace

Relokalizace se používá v případě, že systém náhle ztratí pojem o své pozici. K tomuto dochází především při selhání VO, když není k dispozici dostatek shod mezi po sobě jdoucími obrazy. To může být způsobeno následkem hned několika problémů, jako jsou chvilkové výpadky přijímání obrazů, zaclonění zorného pole, rozmazání vstupu z kamery vlivem rychlého pohybu a tak dále. Řešení problému je pak nalezeno přepočítáváním pozice kamery vzhledem v mapě, kterou systém vytváří a uchovává.

3.3.5 Optimalizace globální mapy

V tomto bodě se systém zaměřuje na upravování mapy jako celku, na zajištění její celkové konzistence. Snaží se minimalizovat odchylku od reálného prostředí. To je zajištěno především takzvaným uzavíráním smyčky.

V průběhu pohybu kamery se může stát, že se drobné chyby postupně kumulují, až vznikne výrazná nepřesnost mezi skutečným prostředím a jeho virtuální reprezentací. Tomuto jevu se dá předcházet zvážením konzistence informací o celé mapě. Pokud se například v nějakém místě během pohybu kamera dostane do již známého bodu, mapa se upraví tak, aby se trajektorie ve zmiňovaném místě protínala. Tímto se uzavře smyčka a upravená mapa se dále používá jako zdroj informací pro optimalizaci.

3.4 Otevřené problémy SLAM systémů

- **Nepřesnosti lokace**

V průběhu pohybu kamery nebo robota se vyskytují drobné chyby, které se časem kumulují. Jak již bylo zmíněno, tento problém je řešen přepočítáváním polohy a uzavíráním smyčky. V některých případech se však může stát, že systém nerozezná místo, na kterém se nachází, jako to, kde už byl. Tím by nedošlo k uzavření smyčky a mapa by se stala nepřesnou. Stejně tak může nastat problém v repetetivním prostředí, kde by mohl algoritmus podobné místo vyhodnotit jako některé, kudy procházel dříve, a mapa by se upravila do nesprávné podoby, což by mohlo celou mapu rozbít.

- **Šumy senzorů**

Při použití jakýchkoli senzorů - v případě vizuálního SLAMu především kamer nebo popřípadě snímačů hloubky - se může projevit chyba přístrojů. To je problém, který se netýká jen SLAM systémů, ale obecně jakékoliv úlohy, které jsou založeny na informacích z těchto senzorů. Pro jeho řešení je třeba zahrnout další opatření, která by byla schopná tyto šumy kompenzovat. K tomu slouží například dříve zmíněný krok relokalizace.

- **Výpočetní a časová náročnost**

Zpracování obrazů a všechny výpočty potřebné pro správné fungování systému jsou výpočetně náročné. S vyšším množstvím informací, větším prostorem určeným k orientaci a mapování nebo se zvyšující se robustností celého systému výrazně stoupá jak výpočetní, tak časová náročnost. Některé stroje nejsou vybaveny na takové množství výpočtů, nebo nejsou schopny je vytvářet v reálném čase.

Je mnoho možností pro řešení těchto typů problémů. Jedním z nich je například nezpracovávat všechny snímky, ale zvolit jen některé, výše zmíněné, klíčové snímky. Další možností, která velmi urychluje výpočetní čas a umožňuje systému pracovat v reálném čase přesto, že je dál velmi spolehlivý, je rozdělit algoritmus na jednotlivé úlohy a ty nechat zpracovávat různá vlákna vícejádrového procesoru.

3.5 Druhy SLAM systémů

Existuje řada implementací SLAM [29]. Nejdůležitější z nich jsou společně se základními charakteristikami použitých metod [30] uvedeny v následující tabulce.

Název	Přístup	Metoda	Hustota mapy	Globální optimalizace mapy	Uzavření smyčky
MonoSLAM	Filtr	Nepřímá	Řídká	Ne	Ne
PTAM	Optimalizace	Nepřímá	Řídká	Ano	Ne
ORB-SLAM	Optimalizace	Nepřímá	Řídká	Ano	Ano
RGB-D SLAM	RGB-D	RGB-D	Hustá	Ano	Ano
OpenVSLAM	Optimalizace	Nepřímá	Řídká	Ano	Ano

Toto je pět druhů vizuálních SLAM systémů, které jsou v této práci zmíněny. Další odstavce podrobněji popisují jednotlivé charakteristiky.

Přístup vypovídá o tom, jaký přístup SLAM systémy využívají pro odhad polohy kamery. Dvě základní kategorie jsou SLAMy, které používají filtry a SLAMy, které ne [31]. Systémy používající filtry jsou založeny na pravděpodobnostních metodách. Filtrování propojuje informace a měření ze všech snímků postupně tím, že aktualizuje rozdělení pravděpodobnosti mezi prvky a parametry pozice kamery. Metody založené na BA provádějí optimalizaci vybraných obrázků ze vstupního proudu. Pro tyto optimalizace využívají například posuvné okno nebo prostorově rozmístěné klíčové snímky, které umožňují dlouhodobý provoz bez akumulace chyby. Další speciální skupina spadající do tohoto dělení je získávání informací z RGB-D kamer. Tyto kamery mají informaci i o hloubce prostředí, která je využívána pro usnadnění výpočtů - není třeba hloubku explicitně počítat.

Metody SLAMu můžeme rozdělit do třech kategorií podle toho, jakým způsobem využívají informace z přijatého obrazu. Klasifikovat je můžeme do přímých, nepřímých a RGB-D SLAMů [32]. Nepřímé systémy extrahují významné body a ty využívají k určení polohy kamery, získávání trajektorie a vytvoření mapy. Systémy založené na přímých metodách pracují s intenzitou pixelů, nevyhledávají významné body. Tyto systémy na základě znalosti parametrů kamery a optimalizace mapy získávají informace o hloubce obrazu a struktuře prostředí. Ušetří tedy čas, který by spotřebovaly hledáním významných bodů. Mohou ho pak využít k řešení jiných úloh. Výhodou nepřímých metod je pak tolerance vůči změnám osvětlení v obrazech nebo výrazně menší náročnost výpočtů. Přímé metody jsou náročnější, protože se nevěnují pouze vybraným bodům, ale pracují se všemi pixely v obraze. Poslední RGB-D kategorie pracuje s informacemi o hloubce, jak je zmíněno výše. Její fungování je blíže vysvětleno v kapitole 3.5.3.

Hustota mapy označuje, jak detailní je vzniklá reprezentace prostředí. Řídké mapy se snaží velmi omezovat počet bodů, které zobrazují. Ušetří tím náročnost na úkor množství informace, které mapa obsahuje. Husté mapy naopak pracují v ohromným množstvím bodů, které má robota přiblížit k úrovni lidskému vnímání prostředí. Tento typ je výrazně detailnější, zabírá však podstatně větší paměť.

Globální optimalizace mapy nám říká, jestli daný systém využívá toto zpřesnění lokalizace, vysvětlené výše, či nikoliv. Uzavření smyčky pak říká, jestli daný systém zahrnuje i optimalizační krok uzavírání smyčky, který dále zpřesňuje lokalizaci i reprezentaci prostředí. I tento bod je vysvětlen výše.

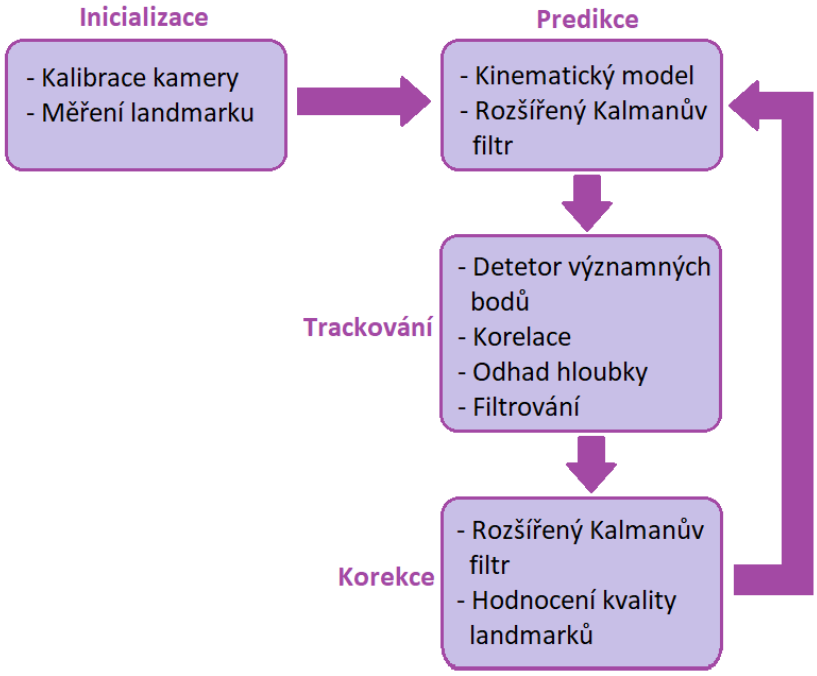
Jednotlivé druhy SLAM systému a principy jejich fungování jsou blíže popsány v kapitolách **3.5.1** až **3.5.6**. Jsou zde vybrány historicky nejvýznamnější systémy s větším důrazem na ty novější.

3.5.1 MonoSLAM

Hlavním charakteristickým rysem MonoSLAMu [33] je, že pro svou funkci využívá pouze jednu kameru. Nemá tedy mnoho informací o hloubce obrazu. Tento druh je sice už překonaný a zastaralý, ale přesto se stále ještě v nějakých aplikacích objevuje. Dal by se považovat za reprezentativní metodu vSLAMu založenou na filtru. Zmíněn je především z historického hlediska, protože se jedná o první ucelený systém monokulárního VSLAMu, který je založen na filtraci. Nevýhodou však je, že s rostoucím prostředím výrazně stoupá i náročnost.

Jako každý SLAM systém generuje MonoSLAM mapu. Prvním krokem je inicializace. Do tohoto bodu spadá kalibrace kamery a vytvoření počáteční mapy. To se provede za využití již známého objektu, jehož globální souřadnicový systém je předem definován. Provádí se jeho pozorování v obraze a pomocí tohoto pozorování je vytvořena první mapa.

Následují kroky predikce, trackování a korekce, které se v cyklu stále opakují. K odhadu pohybu a 3D struktury okolního prostředí využívá MonoSLAM rozšířený Kalmanův filtr. Do něj se během pohybu ukládají jako stavové vektory informace o stavu kamery, kam spadá poloha, rychlost, úhlová rychlost a orientace. Dále se ukládají údaje o poloze bodů pozorovaných v obraze. Přístup využívá pravděpodobnostní mapu, která sleduje jak odhady jednotlivých stavů kamery, tak jejich nejistotu a pozorování. Tato mapa se neustále vyvíjí, během pohybu se aktualizují odhady, nejistoty a pozorování objektů. Na Obrázku 24 je zobrazeno schéma principu MonoSLAMu.



Obrázek 24: Schéma funkce MonoSLAMu

3.5.2 PTAM

Paralelní Sledování a Mapování neboli Parallel Tracking and Mapping (PTAM) [34] je další z druhů SLAMu, který řeší některé nedostatky, jimiž trpí MonoSLAM. Hlavním pokrokem od předchozího typu je rozdělení sledování kamery a mapování okolního prostředí do dvou samostatných vláken vícejádrového procesoru. Jedno vlákno se využívá pro lokalizaci a druhé pro mapování, čímž se docílí toho, že jsou tyto úlohy prováděny současně. To má za následek výrazné zkrácení výpočetního času. Je tedy možné tyto úlohy provádět v reálném čase. Další výhodou vícevláknového přístupu je také vlastnost vláken běžet odlišnou rychlostí, čehož se využívá jak pro PTAM tak pro ORB-SLAM. PTAM byl prvním systémem, který představil pro řešení lokalizace a mapování multivláknový přístup. Od té doby je hojně využíván i v jiných typech SLAMů.

PTAM využívá takzvaný Bundle Adjustment, zmíněný v kapitole **3.2**. BA i přes svou výpočetní náročnost může být díky paralelnímu přístupu zahrnuto do úlohy mapování, čímž se zvyšuje jeho přesnost. Mapa se však i přes tato urychlení neaktualizuje po každém snímku. Algoritmus volí klíčové snímky, u kterých k tomuto přepočítání a aktualizaci dochází.

Prvním krokem algoritmu je vytvoření počáteční mapy za pomoci pětibodového algoritmu. Během trackování jsou pak body z mapy vloženy do obrazu a hledáním shod textury se naleznou korespondence mezi nimi. Díky tomu je možné odhadnout polohu kamery. Při mapování je v klíčových snímcích pomocí triangulace vypočtena 3D pozice nových významných bodů, která se upraví za použití BA. Na základě těchto bodů pak je aktualizována mapa.

Jaké se zvolí klíčové snímky záleží na konkrétní aplikaci. Když jde pouze o mobilní zařízení, kterému chybí výpočetní výkon na provedení náročného algoritmu, jako klíčové snímky se zvolí méně obrazů z kamery, aby se dosáhlo menšího množství výpočtů.

3.5.3 RGB-D SLAM

Tento vizuální SLAM [35] je založen na RGB-D kameře. Jeho výstupem je hustá mapa. Tyto kamery mají schopnost získávat i informace o hloubce. Tím rozšíří znalosti systému o okolním prostředí. Existují omezení, do jaké délky mohou hloubku snímat, takže se preferuje použití v uzavřených prostorách. SLAM systém pak může využívat informace z hloubkové mapy, díky čemuž jsou jeho výsledky přesnější.

Tento SLAM je nadstavbou nad hustou VO. Bylo však potřeba dořešit problém kumulované chyby a také získávat 3D mapu prostředí, čehož samotná VO není schopna.

Vstupem tohoto systému v každém časovém okamžiku je obraz intenzity a odpovídající mapa hloubky. Důležitý je předpoklad takzvané foto-konzistence, což je fakt, že ve dvou následujících obrazech má pixel zobrazující stejný kus reálného světa stejnou intenzitu. Tento předpoklad platí, nemění-li se mezi snímky jas, senzor nepřidává do vstupu šum a scéna je statická. Na základě informace ze vstupů se pak počítá odhad pozice kamery.

Model kamery zohledňuje parametry kamery a určuje souřadnice bodu na základě inverzní funkce projekce. Do výpočtu zahrnuje pixelové souřadnice bodu, jeho hloubku a parametry kamery jako jsou ohniskové vzdálenosti a souřadnice centra kamery.

Dále je třeba reprezentovat pohyb kamery, k čemuž se využívá například transformační matice T 4x4.

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix},$$

kde R je rotační matice o velikosti 3x3 a t je translační vektor. Dalším krokem je za využití souřadnic bodu x získaných z modelu kamery a transformační matice T pomocí deformační funkce τ vypočítat polohu bodu z prvního obrazu v obraze druhém.

$$x' = \tau(x, T)$$

Dále se definují fotometrická a hloubková chyba, r_I a r_Z, porovnáním s hodnotami v prvním obrázku a odhadnutými hodnotami z druhého snímku. Do výpočtu těchto chyb se zahrnuje deformační funkce, vzorce tedy vypadají následovně:

$$r_I = T_2(\tau(x, T)) - I_1(x)$$
$$r_Z = Z_2(\tau(x, T)) - [T\pi^{-1}(Z_1(x))]_Z,$$

kde $[.]_Z$ vypovídá o hloubce daného bodu určené z předchozího snímku.

Další výpočet se zakládá na pravděpodobnosti. Hledá odhad parametrů pohybu ξ^*, přičemž bere v úvahu vážené fotometrické a hloubkové chyby, ze kterých vytvoří dvourozměrnou náhodnou proměnnou $r = (r_I, r_Z)^T$. Odhad se pak vypočítá podle vzorce

$$\xi^* = \arg\min_{\xi} \sum_{i}^{n} \omega_i r_i^T \Sigma^{-1} r_i,$$

kde ω_i značí váhy. Jelikož ale ξ není lineární, rovnice se linearizuje prvním řádem Taylorova rozvoje kolem aktuálního pohybového odhadu ξ_k. Během každé iterace se upravují matice Σ a váhy w_i pomocí standardního EM (expectation maximilization) algoritmu.

Postup popsaný v předchozích odstavcích akumuluje chybu, protože pravděpodobnostní odhad není nikdy úplně přesný. K limitaci tohoto unášení chybou je využíván odhad transformace mezi aktuálním obrázkem a klíčovým snímkem. Klíčové snímky jsou vkládány do mapy a když se v průběhu algoritmu objeví dříve viděné místo, spojí se s

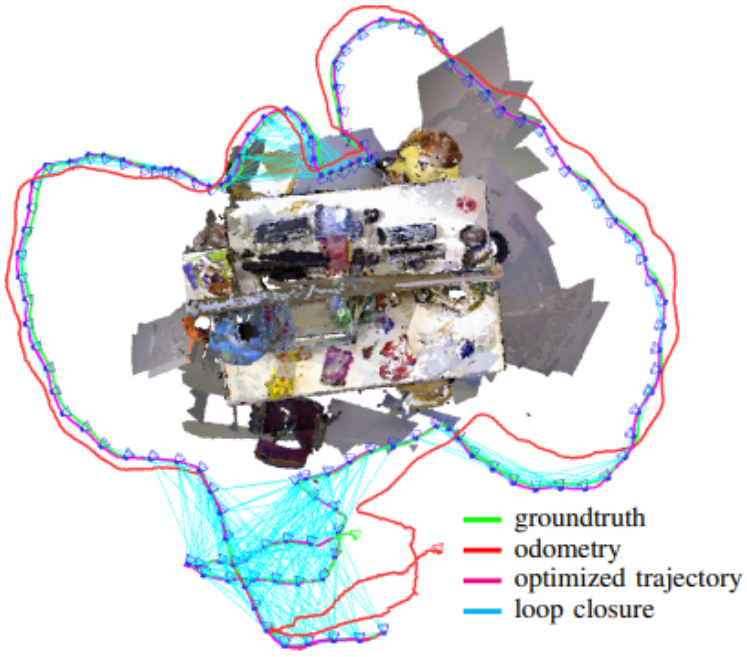
daným klíčovým bodem. Tímto se provádí uzavírání smyčky, které koriguje hromadící se chybu. SLAM tedy k VO přidává výběr klíčových snímků, detekci a uzavírání smyčky a optimalizaci mapy.

Výběr klíčových snímků probíhá tak, že když už se aktuální obrázek nedá přiřadit k poslednímu KS, vytvoří se nový. Existuje mnoho strategií, kdy vložit do mapy nový KS. Jako příklad se dá uvést vložení po daném počtu nových snímků, při určité vzdálenosti translace a rotace nebo při nižším počtu stejných významných bodů mezi novým snímkem a KS než je daný práh. Je také možné využít diferenciální entropii. Pro ni však nelze určit běžný práh, tak se používá poměr entropie mezi posledním aktuálním snímkem a posledním KS. Když se vzdálenost mezi těmito snímky zvětší, zvětší se i entropie a tento poměr se zmenší. Když poměr klesne pod danou hodnotu, vloží se do mapy nový KS.

Pro uzavírání smyčky není s postupem algoritmu možné procházet každý klíčový snímek kvůli rostoucí výpočetní náročnosti. Využívají se metody, které mají tuto náročnost snižovat. Je potřeba aktuální snímek porovnávat jen s nejpodobnějšími KS. K tomu se využívá například tvorby příznakových deskriptorů, které je snazší porovnávat. Dále se může používat například metrické nebo pravděpodobnostní metody hledání nejbližšího souseda.

Mapa se pak reprezentuje jako graf poloh kamery, kde je každý vrchol pozice KS. Hrany jsou pak relativními transformacemi mezi KS. Uzavření smyčky se stává další hranou v grafu. Náprava chyb je pak prováděna například řešením problému nelineární optimalizace nejmenších čtverců.

Velkou výhodou tohoto SLAMu je kombinace fotometrické ale i hloubkové chyby. Dává lepší výsledky než odometrie založená pouze na hloubce nebo RGB, protože postihuje obě tyto oblasti. Využití KS také výrazně snižuje kumulaci chyb v průběhu algoritmu. Na Obrázku 25 lze vidět ukázku z RGB-D SLAMu.



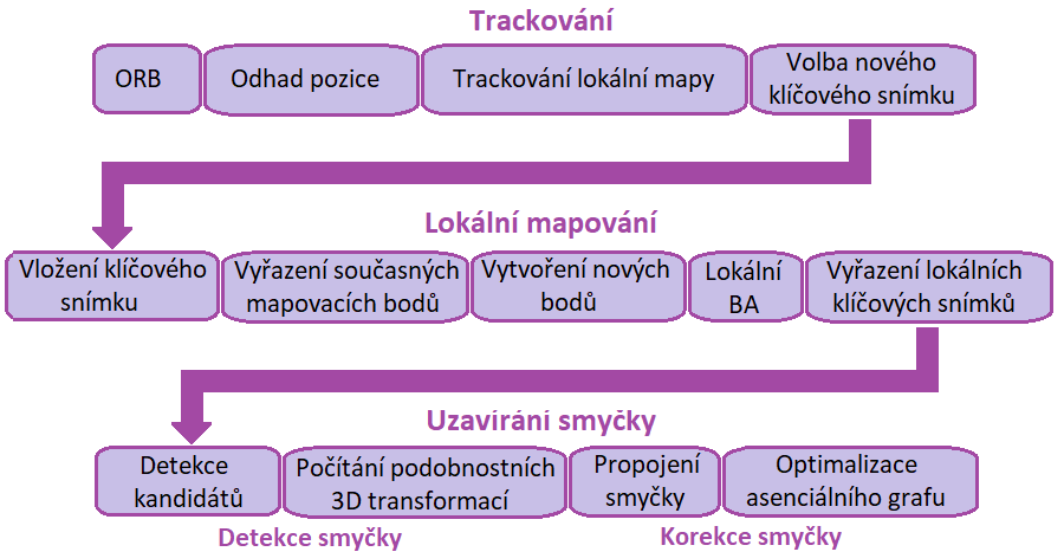
Obrázek 25: Příklad výsledků RGB-D SLAM

3.5.4 ORB-SLAM

Jak již název napovídá, ORB-SLAM [36] využívá jako deskriptor ORB (Oriented FAST and Rotated BRIEF). Jedná se o kombinaci FAST detektoru a BRIEF (Binary Robust Independent Elementary Features) deskriptoru. Tato kombinace je dále modifikována pro zlepšení výkonu.

ORB nejprve využívá FAST agloritmus pro nalezení klíčových bodů. FAST však nepočítá směr, proto se provedla úprava algoritmu, aby zahrnoval i orientační invarianci. ORB pro nalezené FAST body orientaci dopočítá. Poté vypočítá hodnotu R Harrisovým výpočtem. Z těchto bodů vybere několik nejlepších. Aby dosáhl větší robustnost vůči změně měřítka, využívá pyramidu. Pro popis příznaků je použit BRIEF deskriptor, se kterým ORB rotuje podle orientace klíčových bodů, čímž kompenzuje problémy BRIEFu s rotací.

Dalším významným rozdílem oproti předchozím systémům je využití třech vláken, což významně urychluje proces. Úloha je rozdělena do třech současně běžících úloh a to trackování, lokální mapování a uzavírání smyčky. Schéma tohoto systému je zobrazeno na Obrázku 26.



Obrázek 26: Schéma tří vláken ORB-SLAM

Trackování provádí lokalizaci kamery přes každý obraz a určuje, kdy se vkládá nový KS. Prvním krokem je ORB extrakce významných bodů. Aby bylo dosaženo homogenního rozdělení, jsou provedeny úpravy, aby byl v každém úseku nalezen alespoň předem daný počet bodů. U nich je pak vypočítaná orientace a je jim přidělen ORB deskriptor. Mezi obrazy se pak hledají shody pomocí trackování.

Počáteční odhad polohy je dělán dvěma způsoby, záleží na tom, jestli bylo trackování pro poslední snímek úspěšné nebo ne. Pokud ano, na základě modelu konstantní rychlosti kamery se vypočítá nový odhad polohy a vyhledají se v obrazu body z mapy. Poloha je pak optimalizována podle nalezených shod. Když je však trackování ztraceno, nastane úloha globální relokalizace. Ta je provedena převedením snímku do reprezentace takzvané bag of words a hledání kandidátů z klíčových snímků ke globální relokalizaci. Poté se ORB deskriptory příchozího snímku porovnají s klíčovými body KS a za použití RANSAC

algoritmu se odhadne pozice kamery. Když je nalezena takováto pozice s dostatečným počtem inlierů, optimalizuje se pozice a trackování může pokračovat.

Jakmile je k dispozici odhad pozice kamery a sada shod příznaků, provede se projekce mapy do snímku a naleznou se přesnější korespondující body. Mapa, která je k tomuto použita, je složena pouze z malého počtu KS, je tedy jen lokální. Body ze současného snímku se porovnají s mapou a vyřadí se všechny, které neodpovídají kritériím (například leží-li mimo hranice obrazu, mají mezi sebou příliš odlišný úhel atd.) a u zbylých se porovnají OBR deskriptory. Body se spojí s těmi, které s nimi mají nejlepší shodu, a na jejich základě se znovu optimalizuje pozice kamery.

Nový klíčový snímek se vloží do mapy v případě, že je splněno několik podmínek: je-li daný snímek alespoň dvacátý od poslední relokalizace a od posledního vkládání KS, trackuje-li alespoň padesát bodů a trackuje-li méně než 90% bodů referenčního KS.

Lokální mapování

V tomto bodu se s každým novým KS provádí několik kroků. Začíná se s aktualizací pozicového grafu, do kterého se přidá nový uzel reprezentující KS a hrany spojující jej s dalšími uzly. Poté se spočítá reprezentace Bag of Words nového KS.

Pro uchování nových bodů v mapě se provádí test, který má za úkol zajistit, aby se v mapě zbytečně neuchovávaly body, které byly získány špatnou triangulací a nejsou trackovatelné. Bod musí splňovat dvě podmínky a to: trackování ho musí nalézt alespoň ve čtvrtině snímků, ve kterých se dle odhadu má nacházet; když je bod nalezen v KS, musí se nacházet také v alespoň třech navazujících KS. Pokud bod tímto testem projde, je možné ho odstranit až za předpokladu, že by například BA odstranilo některý KS, na kterém je bod pozorován, čímž by přestal splňovat podmínku zobrazení na alespoň třech snímcích.

Vytváření nových bodů mapy se provádí triangulací ORBu z propojených KS v pozicovém grafu. Pro každý nespárovaný ORB se hledá shoda s jiným nespárovaným ORBem v dalším KS. Tyto páry jsou získány triangulací a pro schválení nových bodů je zohledňována i hloubka kamer, chyba reprojekce, konzistence měřítka atd.

Poté se přechází k lokálnímu BA. Ten nejen optimalizuje právě zpracovávaný KS, ale také všechny ostatní, které jsou k němu připojeny v pozicovém grafu, a také všechny body mapy, které jsou těmito snímky pozorovány. Pozorování označená jako outliery jsou v průběhu i na konci optimalizace vyřazovány.

Posledním bodem vlákna lokálního mapování je vyřazení lokálních klíčových snímků. Tento krok se snaží nalézt nadbytečné KS a vyřadit je. Menší počet KS usnadňuje průběh BA, a ve stejném prostředí zachovává přijatelný počet snímků, čímž prodlužuje možný běh algoritmu. Odstraňují se snímky, jejichž alespoň 90% bodů mapy je viděno v minimálně třech dalších KS.

Uzavírání smyčky

Poslední klíčový snímek K_t je zpracován vláknem uzavírání smyčky a zkouší detekovat a uzavřít smyčku. Nejprve se vypočítá podobnost mezi K_t a jeho sousedy, odkud se vezme nejnižší skóre. Poté se K_t začne porovnávat s KS, ze kterých se vyřadí ty s menší podobností, než je dříve získané skóre a také ty, které s K_t přímo souvisí. Když jsou tři po sobě jdoucí KS podobné současnému snímku, jsou označeny jako kandidáti. Poté se přechází k dalšímu bodu výpočtu transformace podobnosti.

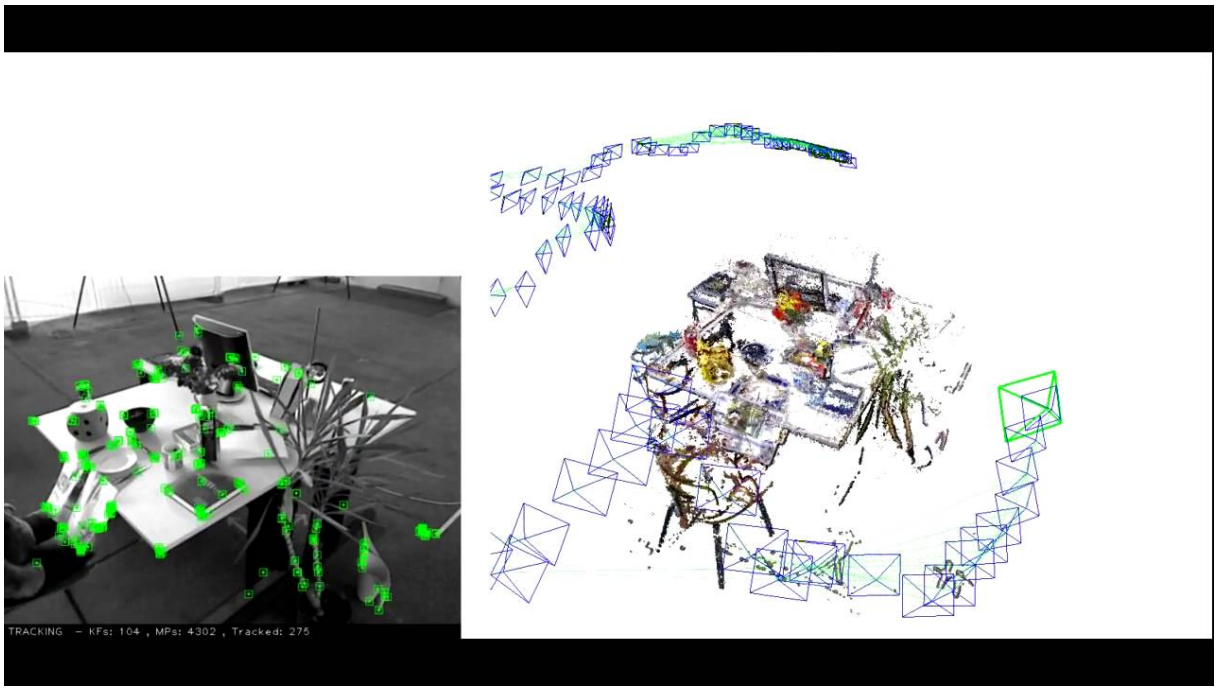
Je třeba zohlednit posun, rotaci a měřítko prostoru zobrazeného na jednotlivých snímcích. Proto je nutné spočítat transformaci podobnosti mezi KS potenciální smyčky a současným bodem, která vypovídá o nahromaděné chybě v průběhu algoritmu. Vypočítá podobnost mezi K_t a KS, které jsou možnými kandidáty k uzavření smyčky. Použije se

RANSAC algoritmus a pokud je nalezeno dostatečné množství inlierů, dojde k optimalizaci a provede se hledání pro více korespondencí. Po další optimalizaci se při dostatečné podpoře inlierů uzavře smyčka.

Při propojení smyčky je důležité spojení duplikovaných bodů mapy a přidání hran do pozicového grafu, kde uzly jsou jednotlivé KS a hrany značí propojení uzlů, které mají společné některé pozorované body. Nejprve se poopraví pozice současného KS za pomoci transformace podobnosti a tato změna je propagována i napříč jeho sousedy, čímž se umožní zarovnání obou stran smyčky. Všechny body mapy, které jsou na snímku smyčky a na jeho sousedech, jsou projektovány na K_t a jsou hledány shody. Všechny tyto shody označené jako inliery se propojí. Všechny KS podílející se na fúzi jsou aktualizovány a vznikají hrany v pozicovém grafu, které značí uzavřenou smyčku.

Po uzavření smyčky se provede optimalizace pozicového grafu přes takzvaný esenciální graf, což je graf, který stále uchovává všechny uzly, ale méně hran pro větší přehlednost. Tato optimalizace rozdělí chybu uzavření smyčky napříč grafem. Zohledňuje také transformaci podobnosti, aby opravila zkreslení měřítka. Každý bod mapy je pak transformován na základě korekce KS, na kterých se nachází.

Na Obrázku 27 je možné vidět výsledek aplikace ORB-SLAM algoritmu navrženého v práci [36], prezentovaného na stránkách ORB-SLAM Project Webpage[6].



Obrázek 27: Ukázka výsledků ORB-SLAM

[6]https://webdiis.unizar.es/∼ raulmur/orbslam/

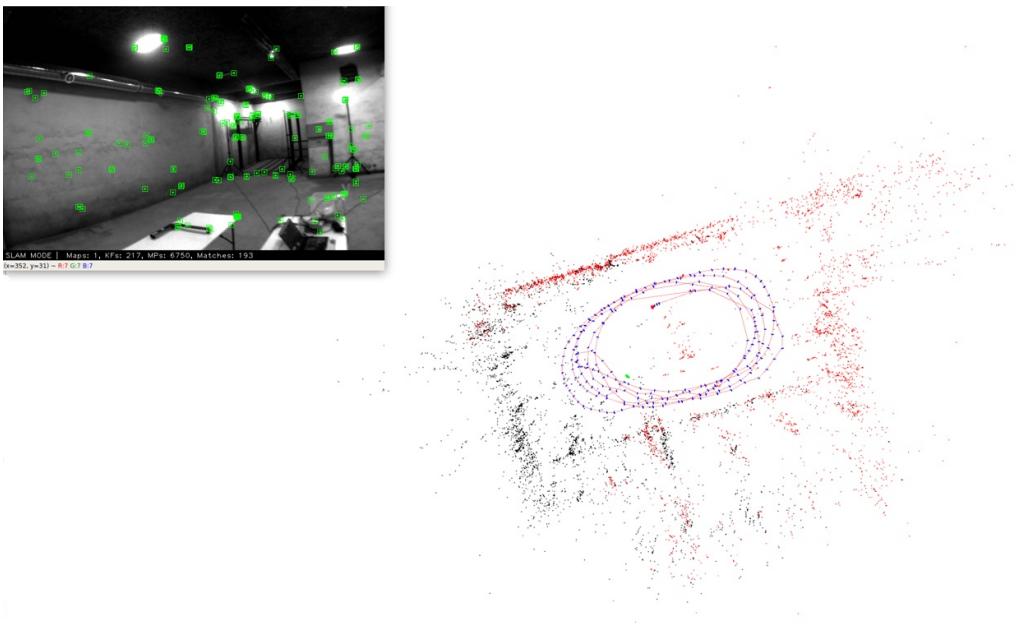
ORB-SLAM2

Tento systém je nadstavbou ORB-SLAMu. Je to open-source schopný pracovat s mono, stereo i RGB-D kamerami. Svého předchůdce ORB-SLAM2 [37] rozšiřuje o několik bodů a to:

- Je to první open-source SLAM pro více typů kamer, který umožňuje uzavírání smyčky, relokalizaci a znovupoužití mapy.
- Výsledky za použití RGB-D kamer popsané v práci [37] ukazují, že použitím Bundle Adjustment je dosaženo větší přesnosti než metodami založenými na iterativním nejbližším bodě nebo minimalizaci hloubkových a fotometrických chyb.
- Díky použití blízkých a vzdálených stereo bodů a monokulárního pozorování jsou stereo výsledky z výše zmíněné práce přesnější než u přímého stereo SLAMu.
- Režim odlehčené lokalizace, který dokáže efektivně znovu použít mapu, i když je samotné mapování deaktivované.

ORB-SLAM3

Jedná se o open-source systém schopný provádět vizuální, vizuálně-inerciální a multi-mapovací SLAM s jednou kamerou, stereo kamerami a dokonce i RGB-D. Tento systém je založen na maximálně aposteriorním odhadu pravděpodobnosti. Podle článku [38] se jedná o systém operující robustně v reálném čase v prostředí s různou rozlohou, venku i uvnitř, který je desetkrát přesnější než předchozí přístupy. Algoritmus pracuje na multi-mapovém principu, který mu umožňuje zvládat i dlouhé úseky se zhoršenou vizuální informací. Začne zkrátka vytvářet novou mapu, kterou pak propojí s původní, když rozezná místo, na kterém se již nacházel. Dalším velkým přínosem je sama ORB-SLAM3 knihovna. Na Obrázku 28 je příklad výstupu za použití ORB-SLAM3.



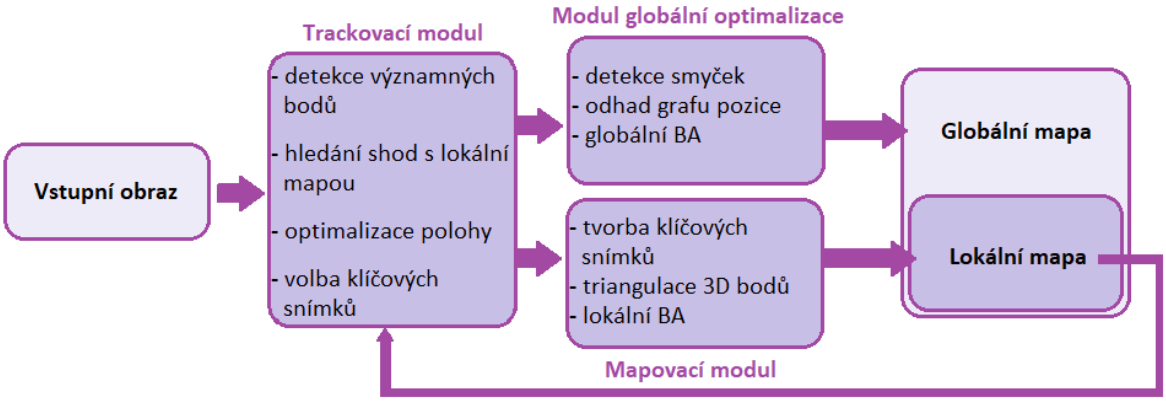
Obrázek 28: Příklad výstupu ORB-SLAM3

3.5.5 OpenVSLAM

OpenVSLAM je open-source software s vysokou použitelností a rozšiřitelností, který zahrnuje jiné, dobře známé SLAM systémy. Jsou zpracovány do samostatných komponent s rozhraním pro využití ke tvorbě různých aplikacích. Jako v předchozím případě je možné využití mono, stereo i RGB-D kamery. Dle článku [39] jsou hlavními výhodami

- OpenVSLAM je kompatibilní s různými modely kamer (fisheye, kamery mobilních telefonnů...) a dokonce může být optimalizovaný pro další modely.
- Mapy, které systém vytvoří, mohou být ukládány a načítány a díky tomu je možné lokalizovat nové vstupní obrázky v již vytvořených mapách.
- Je k dispozici multiplatformní vyhledávač běžící na webových prohlížečích.

OpenVSLAM je implementován převážně v C++. Jde o grafově založený algoritmus využívající nepřímé metody, které byly popsány v dřívějších kapitolách. Software tohoto SLAMu je rozdělen do tří hlavních modulů, kterými jsou trackování, mapování a globální optimalizace. Na Obrázku 29 je schéma systému.



Obrázek 29: Schéma hlavních modulů OpenVSLAM

V porovnání s ORB-SLAM2 na datasetech s předem známými správnými výsledky pro ověření dosáhl OpenVSLAM lepších výsledků. Ve většině datasetů dosáhl výrazně menších chyb.

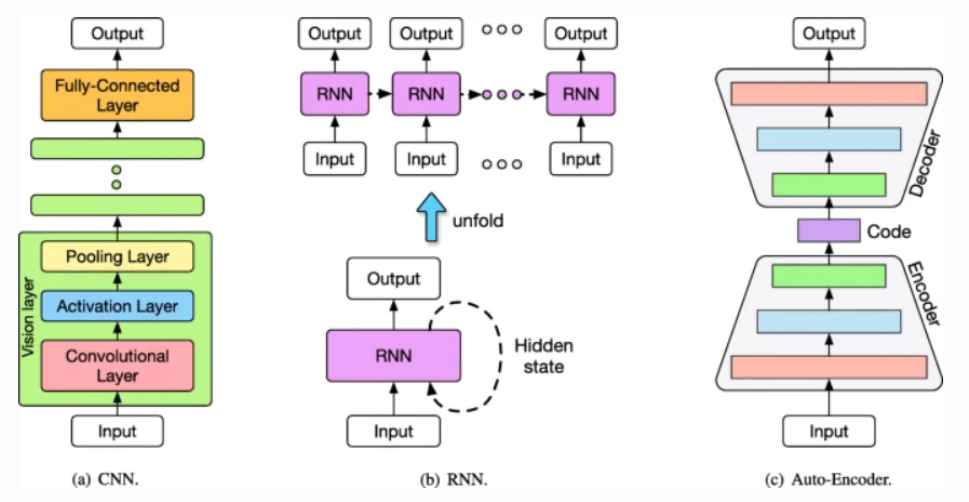
3.6 SLAMy a strojové učení

SLAM systémy založené na hlubokém učení [40] se dokáží automaticky naučit efektivní reprezentování vstupů z masivních datasetů a to způsobem end-to-end, který bude vysvětlen v následujících odstavcích. Tímto způsobem se systémy mohou naučit vytvářet výrazně robustnější a efektivnější příznaky přizpůsobené řešení konkrétní úlohy. Úspěšně prokázaly dobrou schopnost řešit některé náročné kognitivní a percepční úkoly, v konkrétním případě SLAMu například odhad hloubky scény z monokulárního obrazu, odhadování vizuální odometrie nebo generování sémantické mapy. Hlavní výhodou tohoto přístupu je schopnost adaptace, učení a práce s ohromnými soubory dat, které by příznakově založené přístupy nebyly schopny obsáhnout.

Tento přístup ke SLAMu se od modelově zaměřeného způsobu, kterému se věnovaly předchozí části kapitoly 3, výrazně liší. SLAM založený na hlubokém učení se skládá ze třech hlavních kroků a to konstrukce hlubokých neuronových sítí, návrh ztrátových funkcí a řešení flexibility odhadů.

3.6.1 Typy užívaných sítí

Prvním bodem je volba architektury sítě. V SLAM systémech jsou využívány tři typy sítí. Na Obrázku 30 je možné vidět architekturu konvoluční NN, rekurentní NN a nakonec auto-enkóderu.



Obrázek 30: Typy neuronových sítí

Tyto typy sítí se od sebe liší jak strukturou, tak funkčností. CNN mají jako hlavní komponenty konvoluční vrstvy. Výstup této vrstvy je konvoluce vstupu s lineárním filtrem, ke kterému je přičten zvolený práh b. Díky sdílení parametrů mezi vrstvami se jejich počet výrazně snižuje. Důležitá je také ztrátová funkce jejíž minimalizací dochází v rámci trénování k učení neuronové sítě. Volba této funkce záleží na typu úlohy. Například pro klasifikační problémy se využívá především křížová entropie a pro úlohu regrese pak Euklidovská ztrátová funkce.

RNN si na rozdíl od předchozího typu udržují v paměti své skryté stavy díky zpětným smyčkám a na základě toho jsou schopny hledat závislosti současného vstupu na předchozích stavech. Výstupem je pak vážená nelineární funkce zahrnující vážené hodnoty předchozích stavů, ke které se opět přičte práh. Protože ale tento typ sítí trpí ztrátou gradientu, využívá se takzvané dlouhé krátkodobé paměti (long short-term memory - LSTM),

která obsahuje brány rozhodující o tom, kdy se paměť zachová a kdy se vymaže. Pro využití ve SLAMu se většinou využívá propojení CNN a RNN jako tomu bylo u VO. Tímto propojením vzniká RCNN, která je schopná řešit problém odhadu pozice ze vstupních snímků prostředí.

Auto-enkóder je odvozen ze CNN a skládá se za dvou částí. Enkóder pomocí volené nelineární funkce mapuje vstup na skrytý kód a dekóder tuto reprezentaci převádí pomocí další nelineární funkce na rekonstrukci, která představuje stejné významné vlastnosti, které měl vstup.

3.6.2 Úlohy řešené pomocí NN a příklady existujících systémů

Při tvorbě SLAM systémů existuje řada problémů, které mohou být řešeny pomocí hlubokého učení - zapojením neuronových sítí. Příklady těchto úloh se stručným náhledem přístupu, který ve svých pracích využívali ti, jimiž byly tyto problémy řešeny, jsou vypsány níže. Kromě typů úloh je v následujících případech rozlišován i způsob řešení, jde-li o učení s učitelem, nebo o učení bez učitele.

Detekce uzavírání smyčky je ve SLAM systémech velmi důležitým úkolem. Gao [41] na řešení tohoto problému aplikoval neuronové sítě. Navržená metoda se dělí na dvě části: proces učení příznaků a algoritmus detekce smyček. První část metody je realizována pomocí složeného auto-enkóderu, což je typ neuronové sítě, která se učí bez učitele. Jeho úkolem je naučit správně reprezentovat příznaky v obraze, které jsou později užity pro hledání smyček. Auto-enkóder se skládá ze tří vrstev, vstupní, skryté a výstupní, jejichž jednotky počítají sigmoidální odezvu svého vstupu se zohledněním vah a prahů. Když se spojí víc enkóderů, vzniká složený auto-enkóder.

Trénovací proces je modelován jako optimalizační problém, počítající vzdálenost vstupu sítě x a jejího výstupu y, která je obvykle definována jako křížová entropie vstupu. Ta je pro dvě rozdělení p a q na dané množině určena jako: $H(p, q) = -E_p[log\ q]$. Vstupem sítě jsou vektory, které se úpravami získají ze vstupního snímku. Tréninkový proces se provádí aplikací stochastického gradientu sestupu (Stochastic Gradient Descent SGD), který rozděluje trénovací data do malých dávek a v každé aktualizuje W a b. Jako příznaky se označí odezvy skryté vrstvy v okamžiku, kdy optimalizace dosáhne lokálního minima. Algoritmus detekce smyček pak porovnává příznaky získané z auto-enkóderu. Jejich porovnáním vzniká rozdílová matice, která je prvním krokem k určení shody. Když je hodnota na souřadnicích reprezentujících dva obrazy velká, o shodu se zřejmě nejedná a obráceně. Poté se provádí krok redukce řádu matice odečtením velkých vlastních čísel, což zpřesní hledání smyček.

Odhad hloubky je důležitá úloha, bez které se přesný odhad polohy nedá uskutečnit. Existuje několik prací, které se zaměřily na řešení tohoto problému za pomoci NN. Příkladem přístupu **učení s učitelem** je práce D. Eigena a spol. [42], která využívá pro odhad hloubky prostředí pouze jediného snímku. Regrese odhadu hloubky je prováděna pomocí sítě se dvěma komponenty. Jeden nejprve odhaduje globální strukturu scény a druhý ji zpřesňuje pomocí místních informací. Síť je trénována pomocí ztrátové funkce, která kromě bodové chyby zohledňuje i hloubkové vztahy mezi umístěním pixelů.

První komponent, hrubá síť, se skládá z vrchních plně propojených vrstev, díky čemuž uchovává přehled o celkovém obraze. V dalších vrstvách se poté pomocí max-poolingu snižuje dimenze dat. Informace se tak kombinují, čímž je síť schopna integrovat porozumění celé scény, což vede k odhadu hloubky. Výstup této sítě je pak přiveden do druhého komponentu, kterým je jemná síť.

Jemná síť zjemňuje odhad. Jejím vstupem je jak původní obraz, tak výstup hrubé sítě. Tímto způsobem může jemná síť upravit globální předpověď hloubky tak, aby byly detaily popsány v jemnějším měřítku. Tato síť se skládá pouze z konvolučních vrstev a jedné, která provádí max-pooling. Síť se nevěnuje obrazu jako celku, ale pouze částem. Je aplikované napříč mapy příznaků. Při trénování se nejprve natrénuje hrubá síť, zafixují se její výsledky a až poté se trénuje druhá. Výstup je pak odhad hloubky.

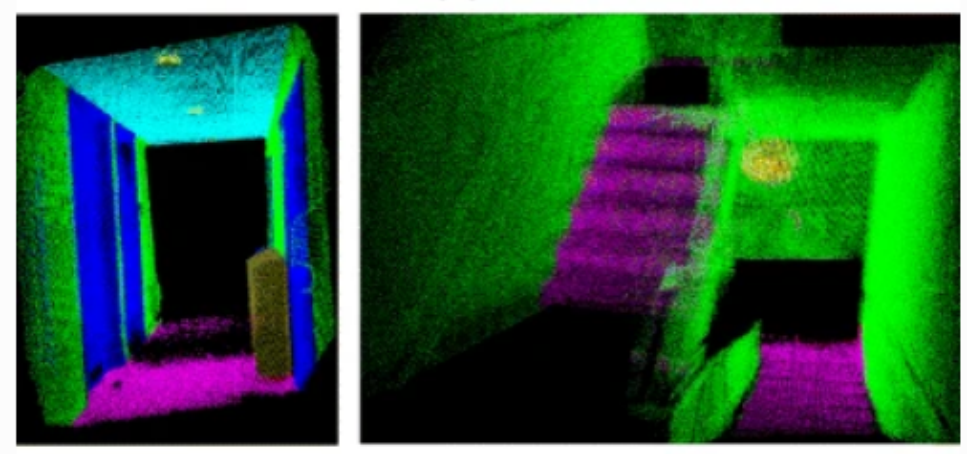
Hlavní myšlenou **učení bez dozoru** je využití schopnosti auto-enkóderu vytvořit reprezentaci obrazu. Enkóder odhaduje hloubkovou mapu pro levý vstupní obraz, a dekodér využívá transformační funkci, která vytváří rekonstruovaný levý obraz z pravého vstupního obrazu a odhadnuté hloubkové mapy. Pro trénování sítě se jako cena využívá chyba rekonstrukce.

Odhad pozice odhaduje pohyb kamery a pozici v prostoru přímo, bez nutnosti explicitního modelu. Systém se učí na základě datově orientovaného přístupu realizovaného hlubokou neuronovou sítí. Úlohy lokalizace, které se řeší pomocí hlubokého učení, jsou například relokalizace nebo odhad 3d pohybu kamery.

A. Kendall a spol uvedl ve své práci [43] poprvé využití CNN pro řešení úlohy regrese odhadu pozice. Pro trénování své sítě využil metod učení s učitelem. Jako základ použil existující GoogLeNet a byl vytvořen PoseNet. Ten byl dál zdokonalován, například R. Li a spol ve své práci [44] přidali možnost vstupů z RGB-D kamer. Hluboké učení užívané pro relokalizaci se zapojuje do řešení dalších úloh.

Sémantické mapování umožňuje robotu kromě získání informace o své orientaci a poloze v prostoru také do jisté míry porozumět svému okolí. Do mapy zanáší například popisky míst nebo objektů. Ty jsou pak často využívány při interakci robotu s člověkem nebo okolním prostředím. Při kombinaci sémantického mapování a SLAM sytému je možné vytvářet sémantické mapy a zároveň odhadovat polohu kamery. Tato úloha by byla velmi náročná bez použití NN. Existuje velké množství prací, které se zabývají sémantickou segregací, mapováním a také sémantickým SLAMem.

Příklad takové studie je práce R.Li a spol. [45], který propojil hlubokou neuronovou síť na základě CNN pro sémantickou segmentaci s 3D SLAM algoritmem za účelem sémantického mapování po pixelech. Použitá síť je složena z konvolučních NN, jejichž vstupy jsou ve dvou proudech. Prvním jsou snímky prostředí a druhým je časový proud s rozdíly mezi obrazy. Výstupem systému je 3D sémantická mapa, která barevně rozděluje různé druhy objektů. Na Obrázku 31 je ukázka takovéto mapy.



Obrázek 31: Příklad sémantické mapy

3.7 Shrnutí SLAM

Přístroj využívající vSLAM je schopen zároveň přijímat obrazy, vytvářet mapu a určovat vlastní polohu v prostředí. Tyto schopnosti se využívají v mnoha situacích ulehčujících práci nebo jiné úlohy.

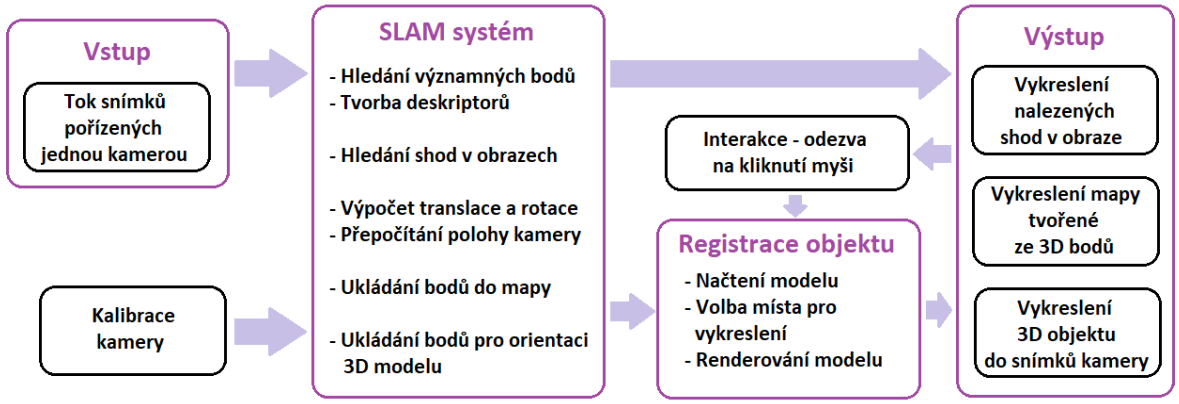
Snímat polohu kamery a okolí v okolí bez znalosti předem daných bodů je neuvěřitelně obtížné. Systémy vSLAM se však při řešení této výzvy ukázaly jako velmi efektivní a objevují se jako jedna z nejdokonalejších dostupných technologií vestavěného vidění. Proto jsou také vhodné pro použití v AR. Kromě faktu, že pracují v reálném čase, jsou velmi dobře schopné odhadovat polohu kamery, což je pro AR systémy důležitým požadavkem, jak je zmíněno v kapitolách výše. AR je pak schopno v reálném čase mísit reálné obrazy s virtuálními objekty a ty přesně umisťovat na své místo.

V této práci bylo popsáno jen několik vybraných vizuálních SLAM systémů, což ovšem plně neshrnulo všechny dostupné možnosti a vývoj těchto technologií. Kapitola se zaměřovala především na popis různých odlišně fungujících systému pro ilustraci možných přístupů k problému.

4 Praktická část

4.1 Návrh AR systému

Cílem této práce je návrh systému bezznačkové rozšířené reality, jehož součástí je vizuální SLAM a zobrazení virtuálního předmětu do skutečného prostředí. Při jejím návrhu jsem vycházela z metod popsaných v předchozích kapitolách. Všechny kódy pro Python použité v této práci jsou uloženy v GitHub repozitáři[7].



Obrázek 32: Schéma systému

Schéma na Obrázku 32 zobrazuje jednu iteraci programu. Kalibrace kamery je krok, který probíhá zvlášť před zbytkem programu, ten není třeba pokaždé opakovat. Z ní do zbytku algoritmu vstupují pouze parametry kamery, které se ručně nastavují v souboru cam_params.py. Jednotlivé kroky dále probíhají v cyklu. Do bloku SLAMu vstupují snímky z kamery. Na základě informací, které jsou dostány výpočty, do výstupu putují informace o 3D mapě složené z jednotlivých bodů a také pozice v pixelech pro zobrazení klíčových bodů v konkrétních snímcích. Interakce ovlivňuje výběr samotného modelu, na zbytek systému nemá vliv. Z bloku registrace objektu pak vystupuje 3D objekt vykreslen do téhož snímku. Jednotlivé kroky jsou blíže popsány v následujících kapitolách.

[7] https://github.com/AnVarackova/DiplomovaPraceAR

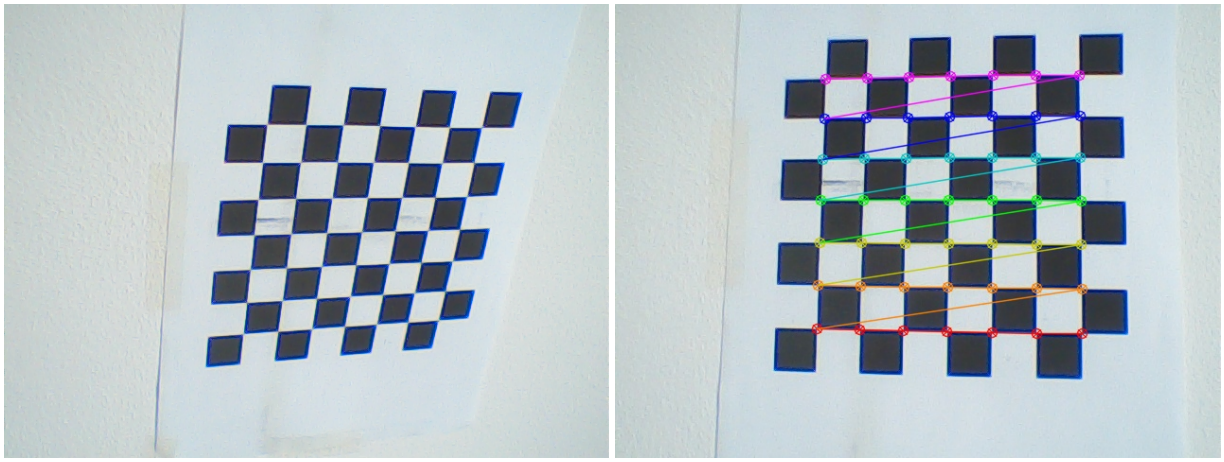
4.2 Kalibrace kamery

Jako u většiny úloh počítačového vidění se nejprve musí provést kalibrace kamery, jejímž úkolem je získat potřebné parametry pro kompenzaci zakřivení čočky. Výstupem kalibrace jsou parametry kamery, které se používají pro další výpočty. Tento proces zajišťuje, že se další úkony mohou provádět s potřebnou přesností.

Parametry zahrnují matici kamery, jež obsahuje ohniskovou vzdálenost f_x a f_y a optická centra c_x a c_y, a parametry zakřivení, kterých je celkem pět. Pro získání těchto podstatných informací jsem využila výše zmíněný kód ze stránek OpenCV. Ve své práci k pořizování snímku využívám kameru Canyon CNR-WCAM43, jejíž kalibrační matice vypadá následovně:

$$C_M = \begin{bmatrix} 872.33317543 & 0 & 297.0579428 \\ 0 & 872.97305203 & 248.74854805 \\ 0 & 0 & 1 \end{bmatrix}$$

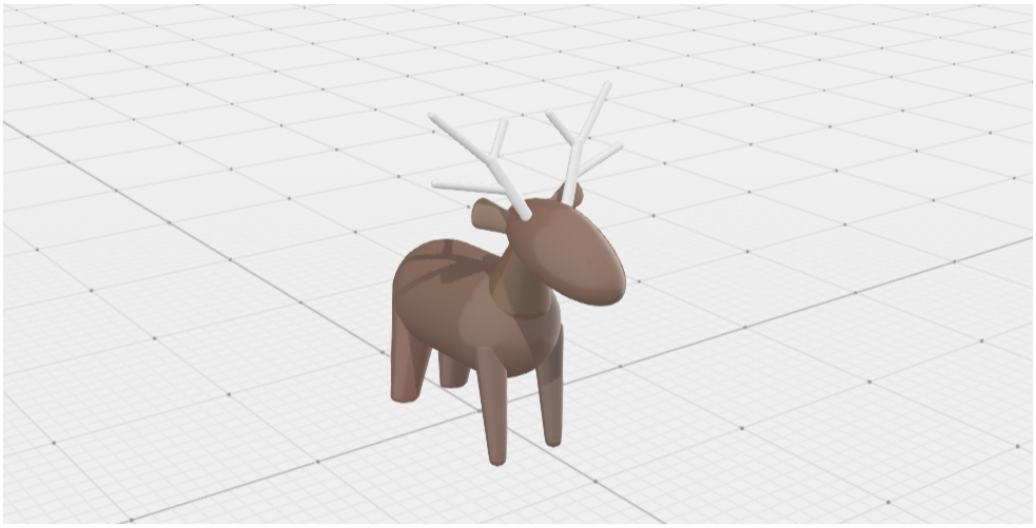
Algoritmus počítá parametry z fotek šachovnice o předem daném počtu políček, viz Obrázek 33. Na druhém obrázku jsou zvýrazněné body, podle kterých se algoritmus orientuje. Kalibrace samotná se provádí mimo hlavní program. Získaná data se nemění. S těmito parametry se tedy můžeme přesunout k dalšímu bodu.



Obrázek 33: Snímek pořízený kamerou a vizualizace nalezených bodů

4.3 Návrh 3D objektu

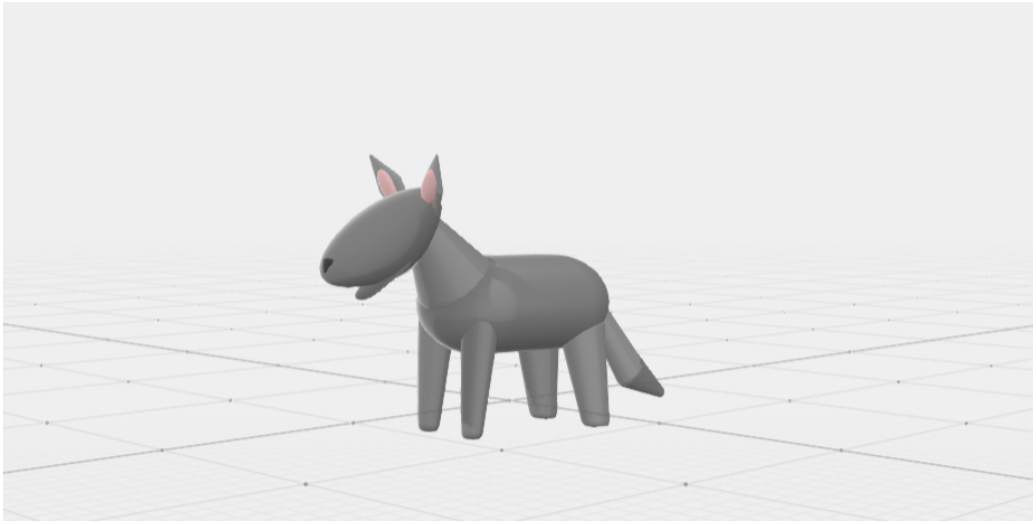
Pro systém prezentovaný v této práci příliš nezáleží na zvoleném objektu, není totiž třeba pro žádnou konkrétní úlohu. Musí být 3D, ale jeho podoba není podstatná. Zvolila jsem si tedy dva velmi jednoduché modely zvířat. Vytvořila jsem je na stránkách vectary.com, jak je výše zmíněno. Na Obrázku 34 je vidět finální podoba prvního 3D modelu, který se v práci objeví jako první.



Obrázek 34: 3D model jelena

Jak lze poznat z obrázku, výsledný model se skládá z různých jednoduchých tvarů. Exportuje se z vectary.com ve formátu zip s několika soubory. Mezi ně spadají obrázky s texturami a vzory, které jsou na model použity, a dále obj a mtl soubory, obsahující informace o modelu. Ty jsou později načítány v pythonu.

Ve svém případě jsem vytvořila dva modely, které jsou střídány, jakmile proběhne interakce uživatele se systémem. Tento proces je blíže popsán v kapitole 4.5. Na Obrázku 35 je možné vidět druhý model, objevující se po proběhnutí interakce.

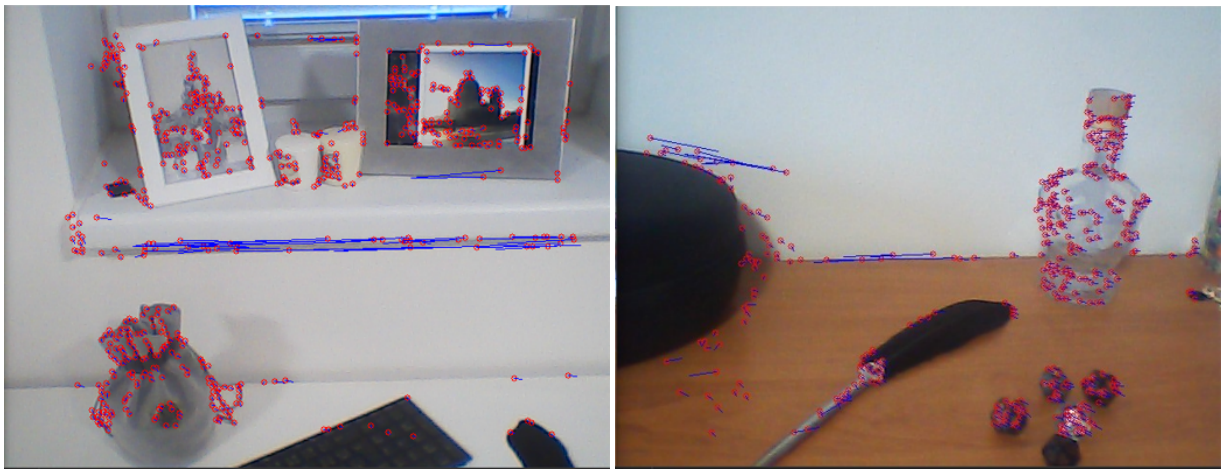


Obrázek 35: 3D model vlka

4.4 SLAM

Systém je založen na algoritmu vizuálního SLAMu využívajícího ORB deskriptor. V této práci pro praktickou část využívám pouze jednu kameru bez informace o hloubce. Tu je nutné vypočítat z po sobě jdoucích snímků. Princip vizuálního SLAM systému je blíže popsán v Kapitole 3.5.4.

Jako základ pro svůj AR systém jsem využila kód[8] z GitHubu, jehož autorem je Yitao Yu, založen na ORB-SLAMu za využití jedné kamery. Tento kód psaný v Pythonu jsem upravila a použila jako zdroj informací pro potřeby kroku popsaného v Kapitole 4.5., kterým je samotné vkládání 3D objektů. Systém za použití ORB deskriptoru hledá významné body, jejichž ukázky je možné vidět na Obrázku 36.



Obrázek 36: Snímaný obraz s vyhledanými významnými body

Systém hojně využívá OpenCV knihovnu. Ta výrazně usnadňuje práci s vizuálními daty. Využívá také knihovnu Open3D, která umožňuje vykreslování 3D mapy. Úhel pohledu na ni se mění stejně s polohou kamery. Na Obrázku 37 je možné vidět ukázky této mapy pořízené ve stejné okamžiky jako snímky z Obrázku 36.

Obrázek 37: Ukázky mapy složené z nalezených 3D bodů

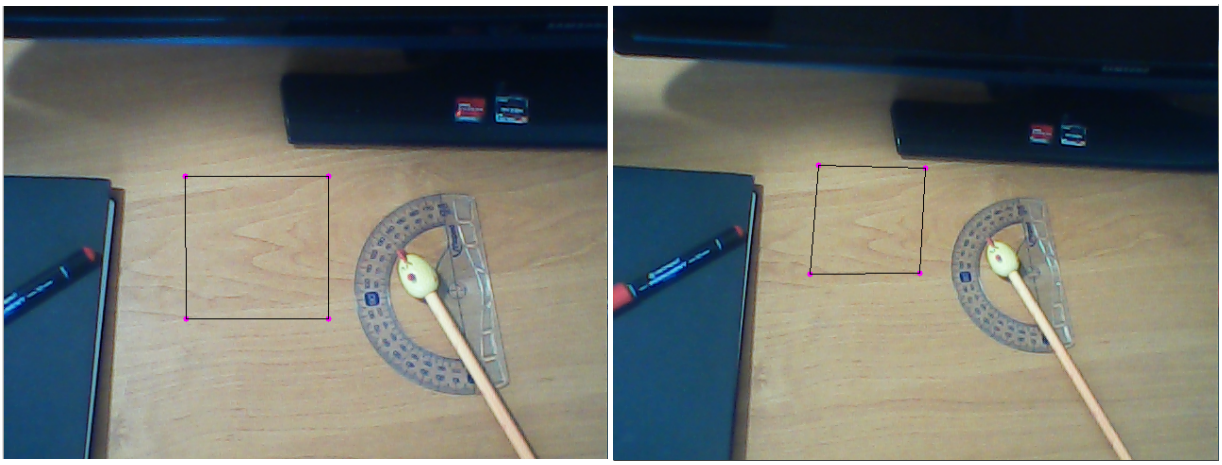
[8] https://github.com/yitao-yu/PythonORBSlAM

Informace ze SLAMu v podobě homografie, jejíž výpočet jsem do kódu přidala, jsou dále využívány pro potřebu úlohy popsané v následující kapitole. Tato matice je nutná k získávání perspektivy a přepočítávání polohy bodů v obraze.

4.5 Vizualizace a interakce

Pro účel vizualizace je nejprve nutné vybrat místo, kam bude objekt umístěn. Jako základ jsem využila kód[9] z repozitáře na GitHubu, od autora Juan Gallostra. On ve své práci vytvořil vizualizaci 3D objektu, který je umístěn na předem známý plochý objekt (například obrázek, sešit, atd.). Kód bylo nutné rozšířit tak, aby bylo možné vkládat 3D objekt do neznámého prostředí.

Pro začátek jsem zvolila čtyři body v obraze. Ty jsem si uložila a později během průběhu programu na základě homogenity získané ze SLAMu přepočítávám jejich aktuální polohu. Na Obrázku 38 je možné vidět vložení těchto bodů, které jsem pro větší přehlednost propojila čarami, ze dvou úhlů.

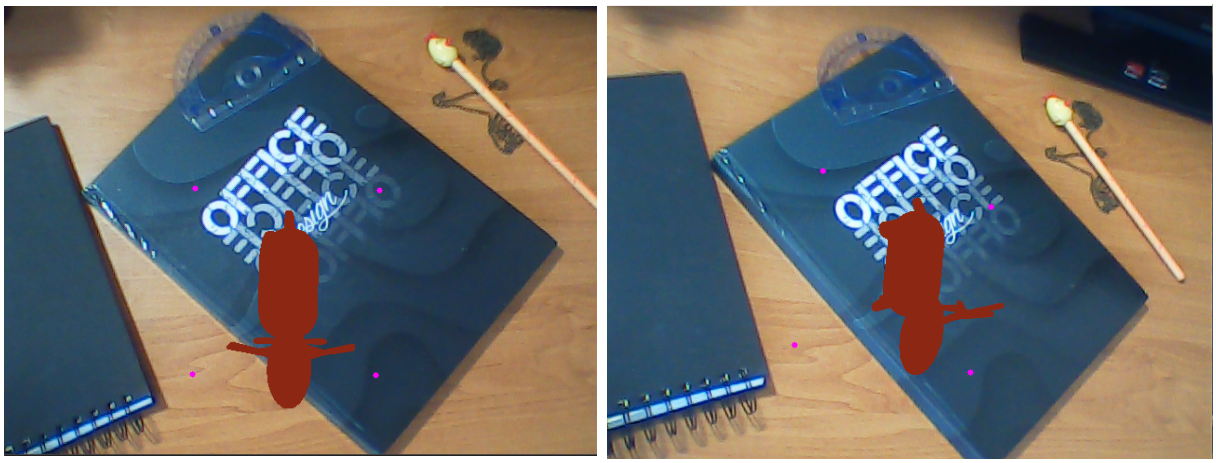


Obrázek 38: Vstupní obraz z vyznačenými zvolenými body

Tento způsob implementace AR funguje, má však několik nedostatků. Vzhledem k tomu, že neřeší uzavírání smyčky a relokalizaci, ale pracuje pouze s homografií získanou ze SLAMu, dochází ke kumulaci driftu, který může při delší sekvenci vést k selhání systému. Je také třeba kamerou pohybovat pomalu, aby nedošlo k deformaci bodů.

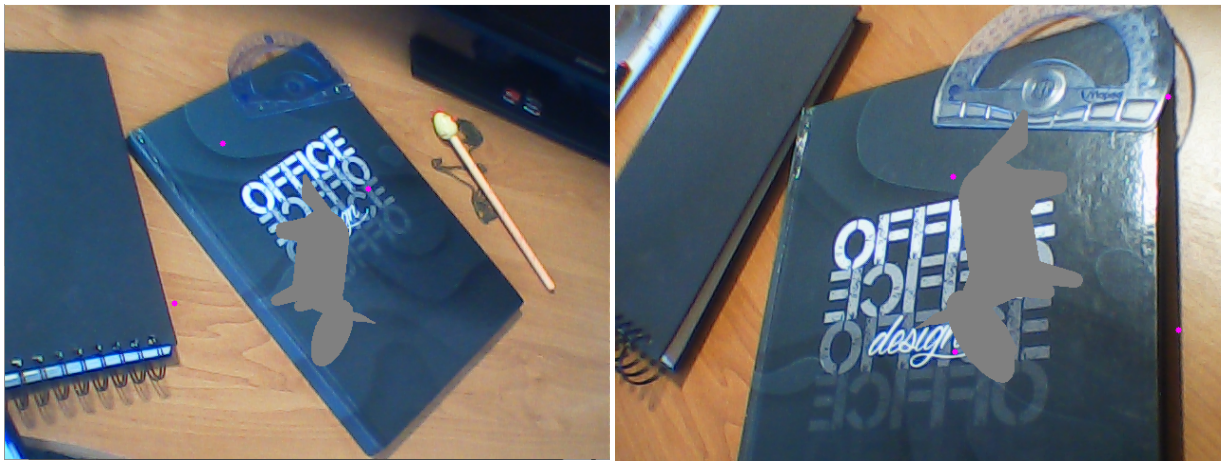
Aplikace vykresluje geometrii vloženého objektu na čtveřici bodů a upravuje jeho vizualizaci na základě dat o pozici kamery. Při implementaci jsem narazila na problém se zobrazením textury, proto jsou v současné verzi vizualizované objekty pouze jednobarevné, bez textury či osvětlení. Na funkčnosti systému však tento fakt neubírá. Ukázky jsou zobrazeny na obrázcích Obrázcích 39 a 40.

[9] https://github.com/juangallostra/augmented-reality



Obrázek 39: Vizualizace 3D modelu jelena na zvoleném místě

Posledním krokem je pak interakce. Je provedena velmi jednoduše. Levé a pravé kliknutí myši (v místě okna s modelem) přenastavují globální proměnnou, která se přepíná mezi hodnotami 1 a 0, a na základě ní se pak vybírá model, který je vykreslován. Systém začíná s vizualizací modelu jelena v hnědé barvě, ke kterému se vrací, když je stisknuto levé tlačítko myši. Pokud uživatel stiskne pravé, modely se vymění a zobrazí se model šedého vlka, který je možné vidět na Obrázku 40.

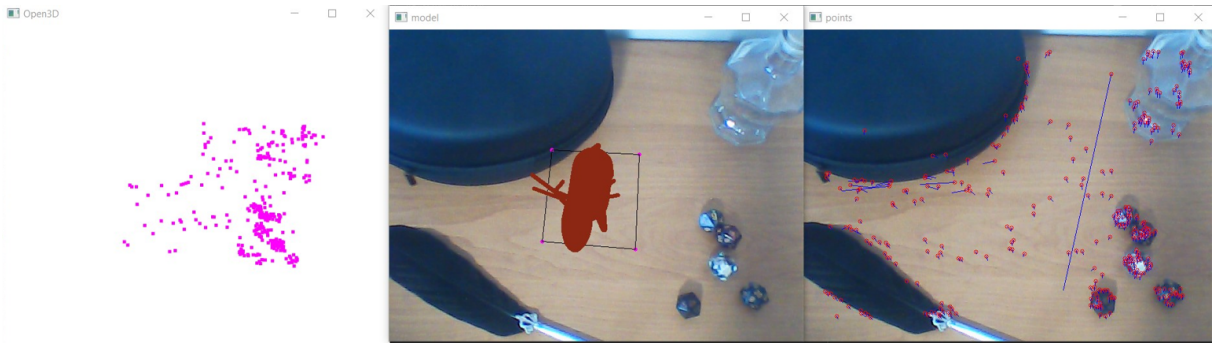


Obrázek 40: Vizualizace 3D modelu vlka

4.6 Poznámky k implementaci

Jak bylo výše zmíněno, všechny kódy i oba modely jsou k nalezení na GitHubu v repozitáři https://github.com/AnVarackova/DiplomovaPraceAR. Model jelena, obj i mlt soubor, se nachází ve složce Deer a model vlka pak ve složce Wolf. Prvním kódem je kalibrace kamery camera_calib.py. Dále se tam nachází soubor cam_params.py, do kterého je nutné získané informace o kameře vložit. ORB_slam_class.py počítá SLAM algoritmus a do třídy Frame, která je využívána v hlavním souboru, ukládá všechny potřebné informace. Objekty ve formátu obj se načítají v souboru load_obj.py, který je založen na skriptu ze

stránek knihovny PyGame[10]. Hlavní soubor AR_SLAM_main.py pak všechno zpracovává a jeho celkovým výstupem jsou tři okna zobrazující 3D mapu, proud snímků s vykresleným 3D objektem a nakonec tok snímků s vyznačenými klíčovými body, viz Obrázek 41.



Obrázek 41: Výstup programu

[10]https://www.pygame.org/wiki/OBJFileLoader

5 Závěr

Ve své práci jsem se zabývala sepsáním rešerše o bezznačkové rozšířené realitě a SLAM systému a jejich možných aplikací. Stručně jsem vysvětlila pojem rozšířená realita. Detailněji jsem se věnovala především bezznačkové AR a popsala její využití spolu s principy, na kterých funguje. V práci se zmiňuji i o několika konkrétních již existujících systémech využívajících právě tento typ rozšířené reality.

V další části jsem popsala, co to je SLAM a vizuální odomertie, jak se liší a základní princip vizuálního SLAMu. Dále jsem se zabývala jeho různými typy a jejich využitím. Zmínila jsem i užívání neuronových sítí pro jeho funkci.

V praktické části jsem navrhla systém AR za využití V-SLAMu. Nejprve jsem provedla kalibraci kamery pro získání jejích parametrů a poté si vytvořila dva 3D modely pro použití v samotném AR systému. Modely jsou ve formátu obj, je tedy možné je nahradit jinými ve stejném formátu.

Na základě již existujícího kódu OBR-SLAMu jsem vytvořila nadstavbu, která z něj bere potřebné informace jako například homografii, a pomocí nich pak přepočítává souřadnice čtyř předem zvolených bodů, které jsou vkládány do prostoru. Na ně poté staví samotný 3D objekt.

Dalším rozšířením je i možnost interakce. Systém reaguje na kliknutí do okna zobrazujícího 3D objekt zasazený v reálném prostředí. Na základě toho, které tlačítko bylo stisknuto, se 3D modely střídají. Všechny kódy k praktické části jsou uloženy na internetu v repozitáři na GitHubu, který je volně přístupný.

Jak je popsáno v předchozích odstavcích, práce se věnuje všem bodům zadání. Napsala jsem rešerše daných témat, zmínila ukázky již fungujících systémů a vytvořila implementaci AR.

Do budoucna je mým hlavním cílem optimalizovat program pro větší stabilitu. Dále bych ráda vylepšila zobrazování - detailnější 3D mapu ze SLAMu a přidání textury, osvětlení či více barev na 3D modely.

Reference

[1] R. T. Azuma, "A survey of augmented reality," *Presence: Teleoperators & Virtual Environments*, vol. 6, no. 4, pp. 355–385, 1997.

[2] J. Paine. 10 real use cases for augmented reality. [Online]. Available: https://www.inc.com/james-paine/10-real-use-cases-for-augmented-reality.html

[3] N. Dey, P. Nandi, N. Barman, D. Das, and S. Chakraborty, "A comparative study between moravec and harris corner detection of noisy images using adaptive wavelet thresholding technique," *arXiv preprint arXiv:1209.1558*, 2012.

[4] M. Bansal, M. Kumar, M. Kumar, and K. Kumar, "An efficient technique for object recognition using shi-tomasi corner detection algorithm," *Soft Computing*, vol. 25, no. 6, pp. 4423–4432, 2021.

[5] M. Fularz, M. Kraft, A. Schmidt, and A. Kasiński, "A high-performance fpga-based image feature detector and matcher based on the fast and brief algorithms," *International Journal of Advanced Robotic Systems*, vol. 12, no. 10, p. 141, 2015.

[6] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and vision computing*, vol. 22, no. 10, pp. 761–767, 2004.

[7] X. Hu, Y. Tang, and Z. Zhang, "Video object matching based on sift algorithm," in *2008 International Conference on Neural Networks and Signal Processing*. IEEE, 2008, pp. 412–415.

[8] E. Oyallon and J. Rabin, "An analysis of the surf method," *Image Processing On Line*, vol. 5, pp. 176–218, 2015.

[9] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[10] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "Kaze features," in *European conference on computer vision*. Springer, 2012, pp. 214–227.

[11] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[12] O. Chum, J. Matas, and J. Kittler, "Locally optimized ransac," in *Joint Pattern Recognition Symposium*. Springer, 2003, pp. 236–243.

[13] O. Chum and J. Matas, "Matching with prosac-progressive sample consensus," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 220–226.

[14] D. Van Krevelen and R. Poelman, "A survey of augmented reality technologies, applications and limitations," *International journal of virtual reality*, vol. 9, no. 2, pp. 1–20, 2010.

[15] B.-K. Seo, J. Choi, J.-H. Han, H. Park, and J.-I. Park, "One-handed interaction with augmented virtual objects on mobile devices," in *Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, 2008, pp. 1–6.

[16] Y. Wang, S. Zhang, S. Yang, W. He, and X. Bai, "Mechanical assembly assistance using marker-less augmented reality system," *Assembly Automation*, 2018.

[17] H. Alvarez, I. Aguinaga, and D. Borro, "Providing guidance for maintenance operations using automatic markerless augmented reality system," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 2011, pp. 181–190.

[18] J.-M. Frahm, K. Koeser, D. Grest, and R. Koch, "Markerless augmented reality with light source estimation for direct illumination," in *Conference on Visual Media Production CVMP, London*. IET Stevenage, 2005, pp. 211–220.

[19] Y. Sato, T. Fukuda, N. Yabuki, T. Michikawa, and A. Motamedi, "A marker-less augmented reality system using image processing techniques for architecture and urban environment," 2016.

[20] H. Kolivand, A. El Rhalibi, S. Abdulazeez, and P. Praiwattana, "Cultural heritage in marker-less augmented reality: A survey," *IntechOpen*, 2018.

[21] T. Kilgus, E. Heim, S. Haase, S. Prüfer, M. Müller, A. Seitel, M. Fangerau, T. Wiebe, J. Iszatt, H.-P. Schlemmer *et al.*, "Mobile markerless augmented reality and its application in forensic medicine," *International journal of computer assisted radiology and surgery*, vol. 10, no. 5, pp. 573–586, 2015.

[22] F. Huang, Y. Zhou, Y. Yu, Z. Wang, and S. Du, "Piano ar: A markerless augmented reality based piano teaching system," in *2011 Third international conference on intelligent human-machine systems and cybernetics*, vol. 2. IEEE, 2011, pp. 47–52.

[23] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[24] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: a survey," *Artificial intelligence review*, vol. 43, no. 1, pp. 55–81, 2015.

[25] H. Lategahn, A. Geiger, and B. Kitt, "Visual slam for autonomous ground vehicles," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1732–1737.

[26] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE robotics & automation magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[27] Y. Tian and M. Compere, "A case study on visual-inertial odometry using supervised, semi-supervised and unsupervised learning methods," in *2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*. IEEE, 2019, pp. 203–2034.

[28] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2043–2050.

[29] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: a survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, pp. 1–11, 2017.

[30] H. Taheri and Z. C. Xia, "Slam; definition and evolution," *Engineering Applications of Artificial Intelligence*, vol. 97, p. 104032, 2021.

[31] H. Strasdat, J. M. Montiel, and A. J. Davison, "Visual slam: why filter?" *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012.

[32] G. Silveira, E. Malis, and P. Rives, "An efficient direct approach to visual slam," *IEEE transactions on robotics*, vol. 24, no. 5, pp. 969–979, 2008.

[33] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[34] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera, and J. J. Berlles, "S-ptam: Stereo parallel tracking and mapping," *Robotics and Autonomous Systems*, vol. 93, pp. 27–42, 2017.

[35] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 2100–2106.

[36] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[37] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[38] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. Montiel, and J. D. Tardos, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, p. 1874–1890, Dec 2021. [Online]. Available: http://dx.doi.org/10.1109/TRO.2021.3075644

[39] S. Sumikura, M. Shibuya, and K. Sakurada, "Openvslam," *Proceedings of the 27th ACM International Conference on Multimedia*, Oct 2019. [Online]. Available: http://dx.doi.org/10.1145/3343031.3350539

[40] R. Li, S. Wang, and D. Gu, "Ongoing evolution of visual slam from geometry to deep learning: Challenges and opportunities," *Cognitive Computation*, vol. 10, no. 6, pp. 875–889, 2018.

[41] X. Gao and T. Zhang, "Loop closure detection for visual slam systems using deep neural networks," in *2015 34th Chinese Control Conference (CCC)*. IEEE, 2015, pp. 5851–5856.

[42] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *Advances in neural information processing systems*, vol. 27, 2014.

[43] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946.

[44] R. Li, Q. Liu, J. Gui, D. Gu, and H. Hu, "Indoor relocalization in challenging environments with dual-stream convolutional neural networks," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 651–662, 2017.

[45] R. Li, D. Gu, Q. Liu, Z. Long, and H. Hu, "Semantic scene mapping with spatio-temporal deep neural network for robotic applications," *Cognitive Computation*, vol. 10, no. 2, pp. 260–271, 2018.

Zdroje obrázků

Obrázek 1 https://www.oneyoungworld.com/blog/future-augmented-reality-advanced-medicine

Obrázek 2 https://www.researchgate.net/figure/Several-examples-of-AR-applications_fig3_326216037

Obrázek 3 https://mhdsupplychain.com.au/2019/05/03/warehouse-future-augmented-reality/

Obrázek 4 https://www.telekom.com/en/company/details/this-is-augmented-reality-614136

Obrázek 5 https://mobilemarketingmagazine.com/lego-is-launching-its-first-augmented-reality-enhanced-building-set-and-app

Obrázek 7 https://medium.com/data-breach/introduction-to-feature-detection-and-matching-65e27179885d

Obrázek 9 Inspirováno z: https://www.researchgate.net/figure/Figure-26-Image-generation-for-augmented-reality-display-Bimber-Raskar-2005_fig4_262379913

Obrázek 10 https://dl.acm.org/doi/fullHtml/10.1145/3290605.3300849

Obrázek 11 Mechanical assembly assistance using marker-less augmented reality system [16]

Obrázek 12 Providing Guidance for Maintenance Operations Using Automatic Markerless Augmented Reality System [17]

Obrázek 13 Markerless Augmented Reality with Light Source Estimation for Direct Illumination [18]

Obrázek 14 A Marker-less Augmented Reality System Using Image Processing Techniques for Architecture and Urban Evironment [19]

Obrázek 15 Cultural Heritage in Marker-Less Augmented Reality: A Survey [20]

Obrázek 16 Mobile markerless augmented reality and its application in forensic medicine [21]

Obrázek 17 Piano AR: A Markerless Augmented Reality Based Piano Teaching System [22]

Obrázek 18 Inspirováno z: https://www.researchgate.net/figure/Our-Edge-SLAM-system-overview-showing-major-blocks-Contributed-blocks-are-highlighted_fig2_322648936

Obrázek 19 https://www.semanticscholar.org/paper/Original-Loop-closure-Detection-Algorithm-for-vSLAM-Bokovoy-Yakovlev/4d9ccb944f6a3d0c19783a641a15012e5616fe2b/figure/2

Obrázek 20 https://medium.com/@dcasadoherraez/introduction-to-visual-slam-chapter-1-introduction-to-slam-a0211654bf0e

Obrázek 22 https://i.ytimg.com/vi/tP1GFapGalQ/maxresdefault.jpg

Obrázek 23 https://arxiv.org/pdf/2006.12567.pdf

Obrázek 25 https://www.semanticscholar.org/paper/Dense-visual-SLAM-for-RGB-D-cameras-Kerl-Sturm/4b838aa7b06b202242916ecaf1387d92e52a40df

Obrázek 26 Inspirováno z: https://www.researchgate.net/figure/ORB-SLAM-system-overview-showing-all-the-steps-performed-by-the-tracking-local-mapping_fig1_271823237

Obrázek 27 https://i.ytimg.com/vi/HlBmq70LKrQ/maxresdefault.jpg

Obrázek 28 https://github.com/UZ-SLAMLab/ORB_SLAM3/issues/28

Obrázek 29 Inspirováno z: https://www.researchgate.net/figure/Main-modules-of-OpenVSLAM-The-figure-is-from-SSS19_fig12_350088488

Obrázek 30 https://link.springer.com/article/10.1007/s12559-018-9591-8

Obrázek 31 https://link.springer.com/article/10.1007/s12559-017-9526-9