VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

OPTICKÝ RADAR S VYUŽITÍM DVOUOSÉHO KAMEROVÉHO MANIPULÁTORU

OPTICAL LOCALIZATION SYSTEM WITH A PAN/TILT CAMERA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. DANIEL SENČUCH**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VÍTĚZSLAV BERAN, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání diplomové práce/19418/2017/xsencu00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Senčuch Daniel, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Optický radar s využitím dvouosého kamerového manipulátoru**
Optical Localization System with a Pan/Tilt Camera
Kategorie: Zpracování obrazu

Pokyny:

1. Prostudujte problematiku obrazového modelování pozadí a detekci popředí při pohyblivé kameře bez proměnlivé ohniskové vzdálenosti. Uvažujte pohybová omezení daná umístěním kamery na dvouosém manipulátoru.
2. Navrhněte systém pro automatickou detekci nových objektů s využitím dvouosého kamerového manipulátoru.
3. Systém implementujte s využitím vhodných existujících knihoven. Systém dále optimalizujte pro dosažení co nejvyšší rychlosti skenování prostředí.
4. Demonstrujte a vyhodnoťte funkčnost vytvořeného řešení na reálných datech. Testujte robustnost systému vůči krátkodobým i dlouhodobým jasovým změnám, periodickým pohybům atd.
5. Zhodnoťte dosažené výsledky a navrhněte možnosti dalšího pokračování práce.
6. Vytvořte plakát nebo krátké video pro prezentování projektu.

Literatura:

- Background Modeling and Foreground Detection for Video Surveillance. Chapman & Hall, 2014. ISBN: 9781482205374.
- An Optimised Background Modelling for Efficient Foreground Extraction. International Journal of High Performance Computing and Networking. Ženeva: Inderscience Publishers, 2017, 10(1-2), 44-53. ISSN 1740-0562.
- Dále dle pokynu vedoucího.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Beran Vítězslav, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Efektivní sledování velkých kritických oblastí je důležité pro zajištění jejich bezpečnosti a soukromí. Řešení pro automatizaci této činnosti nejsou volně dostupná. Tato práce si klade za cíl vytvořit aplikaci pro sestavu dvouosého kamerového manipulátoru a kamery snímající viditelné spektrum. Aplikace na základě polohy manipulátoru a obrazu z kamery zjišťuje sémanticky významné změny v okolí, které je aktuálně snímáno, a vyznačuje oblasti zájmu, ve kterých k těmto změnám došlo.

Abstract

The effective surveillance of large critical areas is crucial for their security and privacy. There is no publicly available and acceptable solution of automating this task. This thesis aims to create an application utilizing a combination of a pan-tilt robotic manipulator and a visible-spectrum camera. Based on the pan-tilt unit's position and camera's images, the application searches for semantically significant changes in the captured environment and marks these regions of interest.

Klíčová slova

Optický radar, pan-tilt, robotický manipulátor, kamera, počítačové vidění, odečítání pozadí, detekce popředí, model pozadí, sledování oblasti, panorama, Robotický operační systém, ROS, C++.

Keywords

Optical radar, pan-tilt, robotic manipulator, camera, computer vision, background subtraction, foreground detection, background model, area surveillance, panorama, Robotic Operating System, ROS, C++.

Citace

SENČUCH, Daniel. *Optický radar s využitím dvouosého kamerového manipulátoru.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vítězslav Beran, Ph.D.

Optický radar s využitím dvouosého kamerového manipulátoru

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Vítězslava Berana, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Daniel Senčuch
23. května 2018

Poděkování

Vedoucí mé práce, Ing. Vítězslav Beran, Ph.D., věnoval velké množství času a úsilí, aby mi při práci poskytl cenné rady a účinně dokázal najít vždy ten správný směr, kterým se mám vydat. Tímto bych mu chtěl poděkovat.

Dále bych rád poděkoval mému zadavateli, Ing. Davidu Hermanovi, a Ing. Jiřímu Župkovi za jejich zpětnou vazbu, mnohé přínosné diskuze o mé práci a za poskytnutí a pomoc s hardwarem, na kterém jsem mohl výslednou aplikaci testovat.

Obsah

Kapitola 1

Úvod

Efektivní sledování velkých kritických oblastí je důležité pro zajištění bezpečnosti a soukromí letišť, územních hranic, vojenských zařízení, citlivých průmyslových zařízení apod. Cílem takového sledování je detekovat, klasifikovat a případně identifikovat a sledovat příchozí cíle jako např. lidi, vozidla, malá zvířata nebo dálkově ovládané drony. Taková úloha musí být vykonávána v rozmanitých prostředích a za různého počasí a světelných podmínek. Automatizace této úlohy má potenciál zvýšit spolehlivost a snížit náklady jejího vykonávání.

K inteligentnímu sledování perimetru se často využívají mikrovlnné radary, někdy ve spojení s RGB kamerami[1]. Jiné metody využívají obraz ve stupních šedi z termokamery. Každý z těchto senzorů má své výhody a nedostatky, které budou dále prodiskutovány. Senzory bývají umístěny na pan-tilt (PT) jednotce, tedy robotickém manipulátoru schopném rotovat kolem svislé osy a naklánět se kolem vodorovné osy. Takové umístění senzorům umožňuje zachovat užší zorné pole a zároveň potenciál sledovat široké okolí, někdy dokonce ve vodorovném rozsahu 360°. Příklad sestavy PT jednotky a kamery je uveden na obr. 1.1.

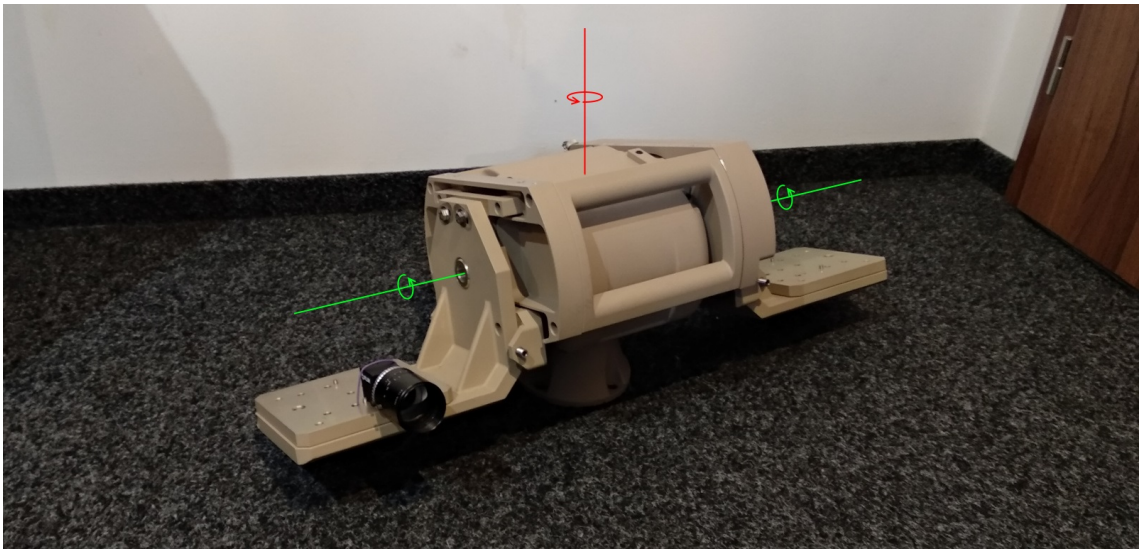
Tato práce si klade za cíl vytvořit aplikaci pro sestavu PT jednotky a kamery bez proměnlivé ohniskové vzdálenosti snímající viditelné spektrum. Aplikace bude na základě polohy manipulátoru a obrazu z kamery zjišťovat sémanticky významné změny ve scéně, která je aktuálně snímána, a vyznačovat oblasti zájmu, ve kterých k těmto změnám došlo. Výsledný detektor musí pracovat v reálném čase a za co nejrychlejšího pohybu manipulátoru, to vše v rozsahu 360° horizontálně a alespoň 84° vertikálně. Přípravná fáze, justáž a učení se prostředí před samotným začátkem detekce je přípustná.

Dále je v této práci aplikace experimentálně vyhodnocena. Byly zjištěny limity použitých algoritmů v rámci maximální rychlosti manipulátoru, povahy prostředí, nutného výpočetního výkonu a přesnosti samotné detekce. Některé z těchto limit jsou testovány na mnou vytvořené datové sadě pořízené ze simulátoru robotického manipulátoru a kamery.

Výstup této aplikace (oblast zájmu, ve které došlo ke změně) by dále mohl být využit jako vstup pro klasifikátor, který by rozpoznal, co změnu v prostředí způsobilo, nebo pro sledovací aplikaci, která se na cíl zaměří a převezme kontrolu nad manipulátorem. Při použití dvou a více sestav manipulátoru a kamery by bylo možné odhadnout polohu cíle ve světě.

Dle mých znalostí všechny dosavadní práce a aplikace na téma kombinace pan tilt jednotky a RGB kamery jsou velmi stručné, není veřejně dostupný jejich zdrojový kód, nebo jsou v nich použity radary a drahé senzory.

[1] RGB kamery jsou kamery snímající viditelné spektrum



Obrázek 1.1: Příklad sestavy dvouosého robotického manipulátoru a kamery. V obrázku jsou vyznačeny osy otáčení manipulátoru.

Kapitola 2 této práce představuje různé typy manipulátorů a senzorů, které lze na úlohu sledování významných změn v okolí použít. Také rozebírá algoritmy používané za tímto účelem. Výběru metody odečítání pozadí a návrhu jejího rozšíření tak, aby dokázala pracovat s pohyblivou kamerou, se věnuji v kapitole 3. Kapitola 4 popisuje implementaci celého systému, jehož funkcionalita byla vyhodnocena v kapitole 5. Na závěr v kapitole 6 shrnuji výsledky celé práce a navrhuji její rozšíření do budoucna.

Kapitola 2

Teorie

Sekce 2.1 této kapitoly představuje různé typy manipulátorů a senzorů, které lze na úlohu sledování významných změn v okolí použít. Problém hledání těchto změn by aplikace pracující se stacionární kamerou řešily použitím některého z mnoha algoritmů pro odečítání pozadí. Tuto problematiku rozebírá sekce 2.2.

2.1 Hardware pro snímání okolí

Existuje široké spektrum hardwarových zařízení, které by mohly ke sledování oblastí sloužit. Tato sekce obsahuje jejich stručný přehled a vysvětlí přínos použití právě pan tilt jednotky s RGB kamerou.

Pan tilt jednotky

PT jednotky jsou robotické manipulátory schopné rotovat kolem svislé osy (změna azimutu) a naklánět se kolem vodorovné osy (změna elevace), čímž mění prostor zabíraný případnými senzory, které jsou umístěné na jejich nosné ploše. Tato schopnost dokáže kompenzovat menší zorný úhel senzorů, které ale na druhou stranu dokážou zabrat scénu detailněji.

Profesionální PT jednotky jsou schopné pracovat venku za kteréhokoliv počasí, které je pro ČR typické. Jejich pohyblivé části mívají konektory (viditelné na obr. 2.1), do kterých lze připojit výstupy senzorů. Napojení pokračuje vnitřní částí manipulátoru. V místě, kde je uvnitř pohyblivá část manipulátoru mechanicky napojena na nepohyblivou základnu, bývají např. rtuťové kruhy, které umožní přenos elektrických signálů i tam. Vše je potom vyvedeno konektorem ze základny manipulátoru a může být napojeno do zařízení, které může manipulátor napájet, řídit nebo zpracovávat data ze senzorů. Kabely mívají typicky na jednom konci proprietární podobu uzpůsobenou pro napojení do manipulátoru, na druhém konci to mohou být klasické koncovky jako UTP nebo pro napájení stejnosměrným proudem. Často vede z jednoho kabelu několik různých koncovek najednou. S manipulátory se tedy typicky dá komunikovat pomocí ethernetového rozhraní zasíláním a přijímáním jednoduchých řetězců textu.

Důležité parametry manipulátorů pro mou aplikaci budou následující: Operační rozsah, maximální úhlová rychlost otáčení a přesnost udávané polohy. Tyto parametry se mohou lišit pro oba směry otáčení. S přesností udávané polohy souvisí také mrtvé úhly, tedy mrtvé operační oblasti v úhlové rychlosti a zrychlení manipulátoru.

Další parametry manipulátoru zahrnují např.: Hmotnost, nosnost, rozměry, podporovaná rozhraní senzorů, provozní teplota, druh napájení atd. Manipulátory také mohou pod-

Obrázek 2.1: Příklady různých pan tilt jednotek[2]. Lze pozorovat konektory na jejich pohyblivých částech a na jejich nepohyblivé základně. Největší manipulátor (vpravo nahoře) má nosnost 80 kg a rozměry cca $45 \times 40 \times 37$ cm.

porovat různé módy řízení, např. zadáním rychlosti otáčení, absolutní nebo relativní pozice atd. Dále mohou podporovat omezení operačního rozsahu nebo maximální rychlosti, některé dokonce stabilizaci pomocí gyroskopu apod. Většinou dokážou s periodou v řádech desítek milisekund hlásit svou aktuální polohu a rychlost v azimutu a elevaci, stejně jako např. napětí nebo stav vstupů a výstupů.

Všechny tyto schopnosti umožňují libovolně složité vzory pohybu kamery, identifikaci její polohy a případně hbité sledování cíle za všech povětrnostních podmínek.

Senzory

Každý typ senzoru má své slabé a silné stránky, ovšem při sledování velkých otevřených prostor i před nevítanými létajícími drony má na základě dřívějších prací [17] [4] [41] smysl uvažovat pouze o třech druzích senzorů: FMCW[3] radary, termokamery a RGB kamery.

Dle [38] se CW radary od kamer liší především tím, že jsou to aktivní prvky – vysílají buď frekvenčně modulované, nebo nemodulované elektromagnetické vlny. Pomocí nemodulovaných vln se vzdálenost k bodu ve scéně měří na základě Dopplerova jevu – pokud se část scény pohybuje, způsobí změnu frekvence vlny, která se od ní odrazí zpět k senzoru. Na základě velikosti změny frekvence lze spočítat vzdálenost k této části scény. FMCW Radary periodicky mění frekvenci vln, které vysílají, a měří dobu letu vln ke každé části scény a zpět k senzoru. Podle frekvence vln, které se vrátí zpět k senzoru, dokáže radar určit, jaká doba uplynula od jejich vyslání.

Caris aj. [4] dokázali pomocí radaru s milimetrovou vlnovou délkou detekovat malá bezpilotní letadla „na stovky metrů“. Pojednávají o výhodách použití takových radarů: Jejich funkčnost není znatelně ovlivněna deštěm, mlhou, kouřem ani nedostatkem okolních zářičů. Milimetrové vlny proniknou skrz oblečení a kamufláž a jsou například silně odráženy většinou výbušnin. Na druhou stranu očividná nevýhoda vysílání nemodulovaných vln je, že se sledovaný objekt musí pohybovat dostatečnou rychlostí, jinak nebude detekován.

[2] Převzato z http://www.2bsecurity.com/wp-content/uploads/2015/12/pan-tilt-series1.jpg

[3] Z angl. Frequency Modulated Continuous Waves – frekvenčně modulované spojité vlny

Obrázek 2.2: Příklad výstupu z termokamery[5]. Pixely zabírající teplejší oblasti jako lidská kůže mají větší intenzitu.

Modulované vlny tuto nevýhodu nemají, ale stále mají nízké rozlišení a přesnost měření vzdálenosti v řádech desítek centimetrů. Kvůli tomu, že jsou to vysílače, jsou také velmi snadno detekovatelné.

Termokamery pasivně snímají tepelné záření, tedy elektromagnetické vlny v různém rozsahu vlnové délky 700 nm až 14 μm. Jejich výstupem je typicky šedotónový obraz, kde světlost jednotlivých pixelů odpovídá intenzitě vln, které z odpovídající oblasti ve scéně přišly. Příklad takového výstupu je vidět na obr. 2.2. Výhodou termokamer je nezávislost na okolním osvětlení. Na druhou stranu jsou extrémně drahé a mívají nízké rozlišení, takže hrozí, že na velkou vzdálenost nezaznamenají s dostatečnou intenzitou vetřelce jako například kvadrokoptéru. To, že výstupní obrázky jsou šedotónové, velmi omezuje další možnosti a spolehlivost zpracování obrazu. Možnosti termokamer při detekci změn ve scéně dále rozebírá [41].

RGB kamery snímají viditelné světlo o vlnové délce cca 390 až 700 nm s tím, že toto spektrum ještě dělí na 3 barevné složky: červenou, zelenou a modrou. To umožňuje spolehlivější detekci změn ve scéně stejně jako případnou klasifikaci změny nebo sledování cíle. Takové kamery jsou velmi rozšířené, levné, mají vysoké rozlišení a je pro ně dostupná široká škála objektivů, které mohou přizpůsobit jejich charakter vnímání okolí. Nicméně jsou závislé na okolním osvětlení a viditelné světlo je zkreslováno např. deštěm nebo turbulencemi.

RGB kamery jsou tedy lehce dostupná řešení s potenciálem za vhodných podmínek spolehlivě sledovat i velmi vzdálené (respektive malé) objekty. Ze všech zmíněných důvodů jsem se rozhodl blíže prozkoumat jejich možnosti a limity při detekci změn ve scéně. Pokud jejich výkon nebude dostatečný, budu uvažovat i o možnosti použití jiných senzorů společně s RGB kamerou a o fúzi jejich dat. Hägelen aj. [17] úspěšně použili radar společně s kamerou pro detekci vetřelců, ovšem pouze na vzdálenost 30 m.

[5]Převzato z http://www.thermoteknix.com/wp-content/uploads/2013/03/men-car-thermal-image.jpg

2.2 Odečítání pozadí

Odečítání pozadí (dále BGS[6]) je algoritmický proces, který segmentuje oblast zájmu obrazu (popředí) od pozadí. Je to jeden z kroků předzpracování obrazu v mnoha úlohách počítačového vidění, jako sledování nebo rozpoznání objektů, bezpečnostní sledování, rozpoznání chování atd. Výběr a použití algoritmu pro odečítání pozadí bude pro mou úlohu nezbytné.

V následujících podsekcích představím výzvy a problémy, se kterými se aktuální metody BGS potýkají, a datové sady určené k jejich porovnávání a testování. Poté budou probrány obecné principy fungování algoritmů BGS, jejich klasifikace, významní zástupci jednotlivých tříd a jejich schopnost řešit obtížné úlohy. Po analýze současných metod BGS v sekci 3.2 jsem vybral metodu ViBE, která je podrobně popsána na konci této sekce.

Výzvy pro metody odečítání pozadí, datové sady

Ačkoliv je princip odečítání pozadí poměrně jednoduchý, kvůli komplikacím reálného prostředí je třeba použít důmyslné algoritmy, které se s nimi dokáží vyrovnat. Dle [40] [3] [23] [41] k těmto komplikacím typicky patří:

- Změny osvětlení – ve venkovních prostředích typicky dochází k postupným změnám intenzity osvětlení, avšak ty mohou být i náhlé, jako například momentální zakrytí Slunce mrakem, nebo dokonce vypnutí světla v místnosti. Ideální algoritmus by se měl vyrovnat se všemi těmito změnami.
- Dynamické pozadí – v pozadí mohou být i pohyblivé předměty, které si nepřejeme klasifikovat jako popředí, jako např. větve stromů, vlny na vodní hladině, semafory nebo blikající světelné tabule. Ideální algoritmus by měl identifikovat periodické i nepravidelné pohyby objektů v pozadí.
- Stíny – ideální algoritmus by měl modelovat pozadí nezávisle na aktuálním zastínění scény a neměl by zastíněná místa klasifikovat jako popředí. Tento názor je ovšem kontroverzní a někteří akademici, např. [8], argumentují pro klasifikaci stínů jako popředí, ať už z principu povahy stínů, nebo kvůli zjednodušení algoritmu klasifikace.
- Šum v obraze – snímky vstupující do algoritmu BGS jsou nevyhnutelně postiženy šumem, ať už kvůli třesu kamery, senzorovému šumu nebo kompresním artefaktům. Ideální algoritmus by měl zohledňovat všechny tyto možné vady.

Pro účel této práce je důležité, aby se vybraný algoritmus dokázal vypořádat se všemi zmíněnými problémy kromě velmi prudkých změn osvětlení. Kromě těchto problémů existují další, které se algoritmy BGS snaží řešit, avšak tato práce na ně příliš velký důraz klást nebude: Kamufláž, špatné počasí, velmi pomalé nebo velmi rychlé pohyby popředí, nízká frekvence snímání scény, třes kamery atd.

Právě taková obtížná data jsou obsažena v datových sadách. Tou zdaleka nejpoužívanější je CDnet 2014 [11] – rozšířená verze původního CDnet 2012 [12]. Obsahuje 53 videosekvencí reprezentujících 11 kategorií včetně vnitřních i vnějších prostor s auty, chodci a dalšími objekty. Většina snímků je anotovaná pro získání ground truth popředí, pozadí a stínů. Součástí této datové sady je i návod a sada programů pro standardizované testování kterékoliv metody. Výsledky testů můžou přispěvatelé poslat autorům sady, kteří je zveřejní na

[6] Angl. background subtraction

svých webových stránkách[7]. Díky tomuto se CDnet 2014 stal benchmarkem, podle kterého se objektivně srovnává velké množství metod již v článcích, ve kterých jsou představeny.

Do jeho vzniku byly však metody testovány na CDnet 2012, který obsahoval 30 videí rozdělených do následujících kategorií:

- Baseline – 4 videa, celkem 6049 snímků. Nejjednodušší případy pro detekci popředí. Kamera i pozadí jsou statické a v průběhu videa se ve scéně předvídatelně pohybují výrazné objekty nebo lidé.
- Thermal – 5 videí, celkem 21100 snímků. Videa pořízená termokamerou.
- Shadow – 6 videí, celkem 16949 snímků. Videa s velkým množstvím tvrdých a měkkých stínů, včetně stínů vrhaných pohybujícími se objekty a lidmi.
- Intermittent Object Motion – 6 videí, celkem 18650 snímků. Videa obsahují objekty v pozadí, které se najednou začaly hýbat, opuštěné objekty a objekty, které se na chvíli zastaví a poté pokračují v pohybu.
- Camera Jitter – 4 videa, celkem 6420 snímků. Videa jsou nahrána kamerou, která se třese nebo pohybuje trhaným způsobem.
- Dynamic Background – 6 videí, celkem 18871 snímků. Pozadí je velmi dynamické, snímky jsou plné vlnící se vodní hladiny nebo třesoucích se větví stromů.

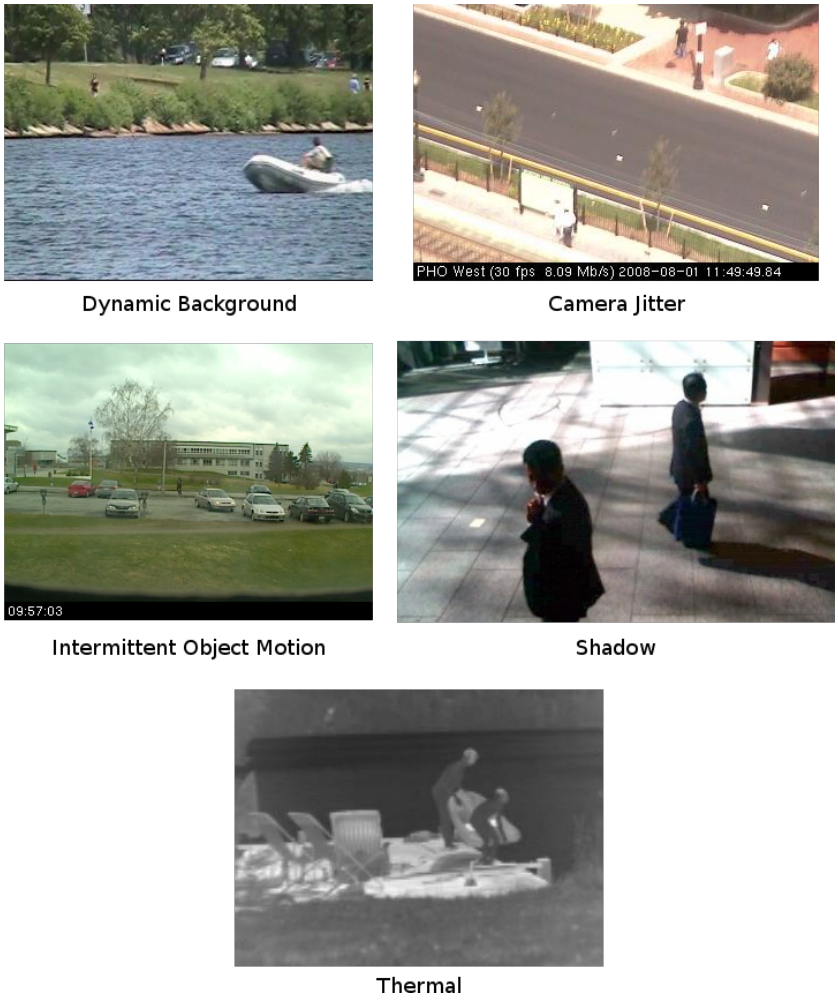
Snímky datové sady mají rozlišení cca 320×240 px, vzácně i 720×480 px. Pokud nebylo uvedeno jinak, videa byla nahrána statickou RGB kamerou s frekvencí 24 snímků za sekundu. Příklady jednotlivých kategorií jsou vidět na obr. 2.3, kde ovšem byla vynechána kategorie Baseline, protože kvůli své jednoduchosti není pro mé účely relevantní.

CDnet2014 přidává k videím svého předchůdce pět kategorií videí:

- Bad Weather – 4 videa, celkem 20900 snímků. Videa pořízená za silného sněžení.
- Low Framerate – 4 videa, celkem 9400 snímků. Videa pořízená s frekvencí 1, 3, 5, 5 a 17 FPS.
- Night Videos – 6 videí, celkem 16609 snímků. Videa natočená v noci, jedno z nich infrakamerou.
- PTZ – 4 videa, celkem 8630 snímků. Kamera byla umístěná na pan tilt jednotce a v některých případech je schopná přiblížení (zoom). Někdy se PT jednotka pohybuje souvisle, někdy velmi náhle. V žádném případě však nejsou k dispozici informace o pozici PT jednotky.
- Turbulence – 4 videa, celkem 15700 snímků. Videa jsou černobílá a pořízena na velkou vzdálenost v prostředí, které mělo ve vzduchu velkou míru turbulencí. Z tohoto důvodu se obraz značně vlní.

Celkem tedy tato sada obsahuje 53 videí a 159 278 snímků. Příklady nově přidaných kategorií jsou vidět na obr. 2.4, byly však vynechány kategorie Low Framerate a PTZ, neboť nejsou pro mé potřeby vůbec relevantní.

[7] Dostupné na http://wordpress-jodoin.dmi.usherb.ca/results2014/



Obrázek 2.3: Snímky z videí z jednotlivých kategorií datové sady CDnet 2012. Kategorie Baseline byla vynechána.

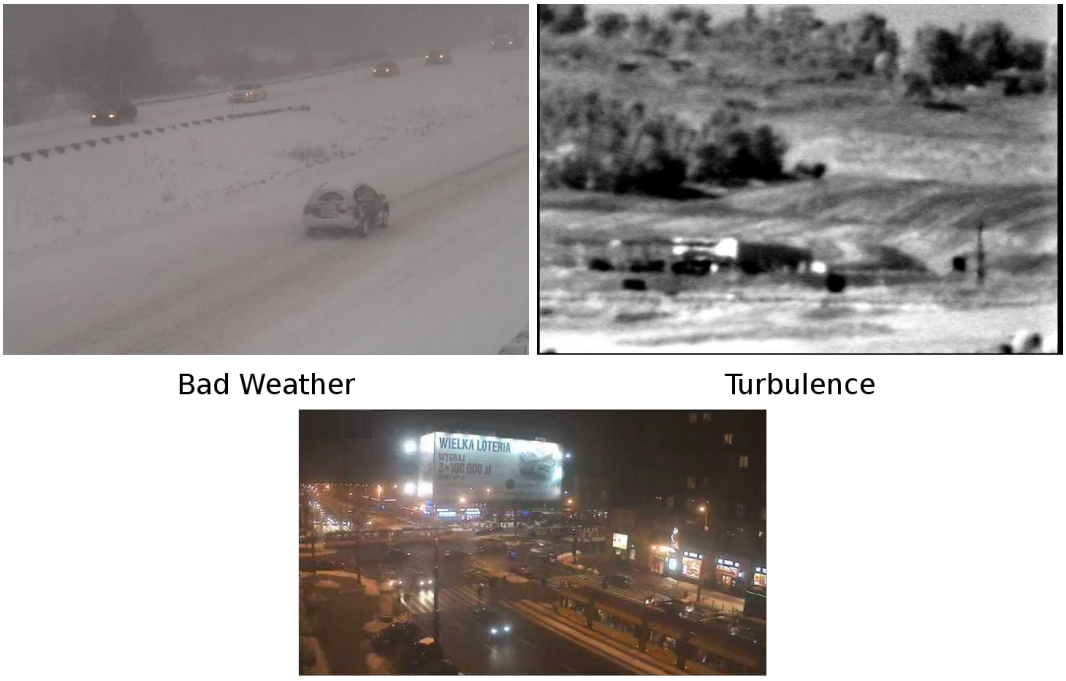
Principy odečítání pozadí

BGS má smysl provádět pouze na videu nebo sadě více snímků. Typický postup je [41]:

1. Vybudovat model pozadí pomocí několika počátečních nebo předchozích snímků.
2. Srovnat aktuální snímek s modelem pozadí a vytvořit segmentační mapu.
3. Aktualizovat model pozadí dle aktuálního snímku a případně segmentační mapy.
4. Načíst nový snímek a vrátit se na bod 2.

Segmentační mapa či maska je obrázek se dvěmi možnými hodnotami pixelů – jedna značí, že pixel byl klasifikován jako popředí, druhá značí, že jako pozadí. Příklad je vidět na obr. 2.5. Obecný algoritmus a tok dat v něm je naznačen na obr. 2.6.

Téma odečítání pozadí bylo ve vědeckém poli velmi bádané a vzniklo přes 700 vědeckých článků, které se jím zabývají. Metody BGS lze proto kategorizovat mnoha způsoby, nicméně převažují především následující [8] [40]:

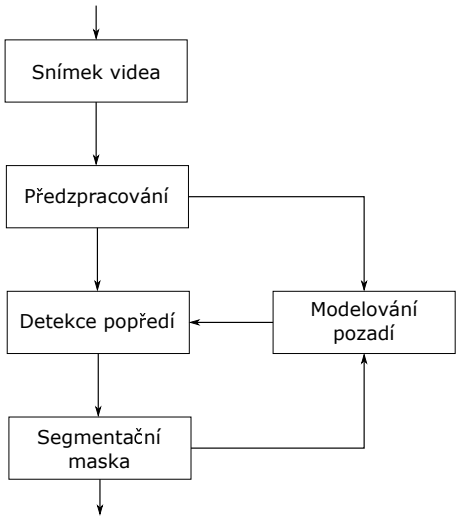


Obrázek 2.4: Snímky z videí z jednotlivých kategorií datové sady CDnet 2014, které jsou oproti CDnet 2012 nové. Kategorie Low Framerate a PTZ byly vynechány.

- Dle povahy modelu pozadí
  - Statistické modely – pozadí je modelováno funkcemi rozložení pravděpodobnosti (dále FRP) nebo alespoň statistickými parametry jako střední hodnota a rozptyl. Ty se s každým novým snímkem patřičně adaptují. Typicky se používají GMM[8]. Zástupci tohoto přístupu byli jedny z prvních obstojně fungujících metod BGS, neboť se dokázaly do jisté míry vypořádat s dynamickým pozadím, šumem i změnami pozadí či osvětlení. Dnes je však typicky předčí metody z jiných kategorií. Zástupci jsou např.: Jedna Gaussova funkce [39], GMM [28], Adaptivní GMM [42], která dokáže měnit počet modelujících Gaussových funkcí dle potřeb každého pixelu, Bayesovská metoda [21] a KDE [9].
  - Pozadí je modelováno sadou vzorků pixelů z minulosti – do jisté míry se v daném barevném prostoru nedá vyhnout nutnosti modelovat FRP hodnot každého pixelu. Autoři metod, které přímo modelují FRP nebo její statistické parametry, však málokdy diskutují statistický význam těchto funkcí. Ve skutečnosti není nutné přímo modelovat FRP, pokud je splněn cíl segmentace pozadí a popředí. Alternativou k tomuto přístupu je postupné zvyšování statistického významu modelu a jedním ze způsobů jak toho dosáhnout je stavět model s opravdovými pozorovanými hodnotami pixelů. Tento způsob vychází z předpokladu, že již pozorované hodnoty by měly mít větší pravděpodobnost, že budou pozorovány znovu, než hodnoty, které dosud pozorovány nebyly. Tudíž tento přístup oproti přímému modelování FRP dává větší smysl ze stochastického hlediska – odchylky

[8] Z angl. Gaussian Mixture Model – směsice Gaussových funkcí rozložení pravděpodobnosti

Obrázek 2.5: Příklad segmentační mapy (vpravo). Vlevo je původní obrázek. Bílé pixely v mapě značí popředí – v tomto případě projíždějící auto.



Obrázek 2.6: Tok dat v obecném algoritmu odečítání pozadí.

od předpokládané FRP jsou totiž všudypřítomné.
Hodnoty vzorků pixelů z minulosti nemusí z časového hlediska nutně následovat za sebou a kromě samotných hodnot pixelů se mohou uchovávat i jiné příznaky. Významní zástupci např. modelují stavy pixelů pozadí pomocí kódové knihy [18] či hledají konsensus mezi vzorky pixelů z minulosti, např. SACON [36] a ViBE [1]. SOBS [19] využívá samoorganizující se neuronové sítě. PBAS [15] staví na metodách SACON a ViBE a dělá je robustnějšími za cenu výpočetní náročnosti, SubSENSE [26] v tomto trendu pokračuje dále.

- Dle zkoumaných jednotek
  - Metody založené na pixelech – zkoumanou jednotkou jsou jednotlivé pixely a jejich příznaky, např. barva nebo směsice Gaussových funkcí. Patří do ní všechny výše zmíněné metody kromě KDE.
  - Metody založené na regionech – zkoumají oblasti pixelů reprezentované příznaky, např. histogramem rozložení hodnot intenzit obsažených pixelů nebo LBP[9]. Tyto

[9] Z angl. Local Binary Patterns – lokální binární vzory (příznaky)

metody jsou odolnější vůči šumu, ale dokáží získat pouze přibližný tvar objektu. Významným zástupcem je KDE [9], která modeluje pravděpodobnosti intenzit pixelů v pozadí v rámci okenní funkce s pomocí několika vzorků z minulosti. Tato metoda není příliš spolehlivá, ale dala základ mnohým zajímavým vylepšením [5] [31] [33] [32] [30] [22]. Vyhledávání LBP [14] bylo také účinné.

  - Hybridní metody – kombinují informace o pixelech i o regionech. Dokáží velmi efektivně segmentovat popředí, ale na druhou stranu jsou náročné na výpočetní prostředky. Významní zástupci: Wallflower systém [34], Huang aj. [16], Tsai aj. [35] či LOBSTER [25], což je ViBE rozšířený o příznaky LBSP[10].

- Dle nutnosti ladění parametrů modelu

  - Parametrické – při klasifikaci pixelů hrají roli i parametry, které se ladí za běhu algoritmu. Tato vlastnost je typicky považována za nevýhodu, neboť proces ladění je citlivý na konkrétní průběh videa, zejména jeho začátku. Do této kategorie spadá většina statistických metod včetně těch, které používají GMM.
  - Neparametrické – všechny parametry jsou dané před spuštěním algoritmu a výsledek klasifikace závisí pouze na obsahu minulých snímků. Tento přístup je flexibilnější, ale náchylnější na špatná data (šum, klepání kamery atd.). Spadá sem většina metod pracujících s historií hodnot pixelů včetně KDE, což je první statistická metoda, která není parametrická.

- Dle politiky aktualizace modelu pozadí

  - Konzervativní – pixely klasifikované jako popředí nikdy nejsou zahrnuty do aktualizace modelu pozadí. Tento přístup zajišťuje konzistenci modelu pozadí, ale hrozí při něm, že objekty vložené nebo odebrané z pozadí nikdy s pozadím nesplynou. Metody využívající tento přístup většinou mají jiné mechanismy, jak tyto případy rozpoznat a ošetřit. Například autoři algoritmu W^4 [13] průběžně tvoří a aktualizují „detection support map“, která počítá kolikrát za sebou byl který pixel klasifikován jako popředí. Pokud tento počet překročí určitý práh, je hodnota pixelu zakomponována do modelu pozadí. Jiné varianty sledují skupiny propojených pixelů popředí, které byly dlouhou dobu nehybné [6]. Autoři W^4 a SACON [36] [37] používají kombinaci aktualizace na pixelové i objektové úrovni.
  - Nekonzervativní/slepá – všechny pixely aktuálního snímku jsou použity pro aktualizaci pozadí. Nicméně třeba metody založené na oknech (KDE apod.) zjemňují výskyt nových hodnot v pozadí tím, že jim zpočátku dají nízkou váhu. Tento přístup není citlivý na duchy, např. předměty odebrané z pozadí, má však problém rozpoznat pomalu pohybující se objekty. Řešením by dle [37] mohlo být rozšíření paměti pro vzorky z minulosti a zvýšení výpočetní náročnosti, to ale nemusí být vždy dosažitelné, zejména pokud kamera snímá scénu s vysokou frekvencí. [9] a [10] navrhují zahrnout do výpočtů dva temporální podmodely, které se starají o rychlé i pomalé změny pozadí. Tento přístup byl efektivní, avšak s ním se zvyšuje problém parametrizace – je nutné ladit více parametrů, aby implementace mohla být praktická.

[10] Z angl. Local Binary Similarity Pattern.

Ačkoliv většina vědeckých článků představujících svou metodu BGS do jisté míry zmiňuje předchozí významné metody, existuje i mnoho průzkumných článků, které se zaměřují výhradně na shrnutí dosavadních metod a datových sad. Xu aj. [40] vybírají nejvýznamnější metody do roku 2016 a objektivně je porovnávají. Stejně tak Yao aj. [41] – ti se zaměřují spíše na zpracování šedotónových obrázků, avšak mnoho principů platí i pro RGB obrázky. Sorbal a Bouwmans vytvořili Background Subtraction Library [23], která poskytuje pro veřejnost implementace mnoha z metod BGS. Tyto metody Sorbal s Vacavantem vyhodnocují ve svém článku [24].

ViBE

Visual Background Extractor je metoda BGS, kterou jsem vybral pro použití v mé aplikaci. (Podrobnosti jsou uvedeny v sekci 3.2.) Byla poprvé stručně představena Barnichem a Drogenbroeckem v [1]. Později byla velmi podrobně probrána a vysvětlena v [2]. Tato podsekce je věnována důkladnému popisu její funkce a rozhodnutí, která autory vedla k tomu dát jí její dnešní podobu.

Pokud jde o charakteristiku metod BGS, autoři ViBE přikládají vysokou důležitost třem otázkám: 1) Jak vypadá model pozadí a jak se chová? 2) Jak je model inicializován? 3) Jak je model aktualizován v čase? Odpovědi na tyto otázky budou uvedeny v následujících odstavcích.

Model pozadí a proces klasifikace

Autoři se chtějí vyhnout přímému statistickému přístupu k modelování pozadí z důvodů uvedených v předchozí podsekci. V modelu pozadí ViBE má každý pixel svou sadu hodnot (vzorků), kterých nabýval někdy v minulosti. Pokud má být aktuální hodnota považována za pozadí, měla by mít dostatečně malou euklidovskou vzdálenost v barevném prostoru od několika vzorků. Tím se ViBE liší od metod založených na konsenzu vzorků, kde je potřeba dostatečně nízká vzdálenost k **většině** uložených vzorků. Pokud má ale stačit pouze několik vzorků, musí být tyto vzorky vybírány pečlivě. Kolem této myšlenky je navržen proces aktualizace pozadí, který je popsaný později.

Formálně tedy nechť $v(x)$ značí hodnotu v daném euklidovském prostoru barev, kterou má pixel v obrázku na pozici x, a v_i je hodnota vzorku pozadí s indexem i. Každý pixel pozadí x je modelovaný množinou N vzorků pozadí

$$\mathcal{M}(x) = \{v_1, v_2, \ldots, v_N\} \tag{2.1}$$

získaných z předchozích snímků. Ty nemusí časově následovat za sebou, jak bude vysvětleno dále.

Aby byla hodnota $v(x)$ klasifikována dle modelu $\mathcal{M}(x)$, je porovnána s nejbližšími hodnotami v množině vzorků tak, že je definována koule $S_R(v(x))$ o poloměru R se středem na $v(x)$. Hodnota $v(x)$ je pak klasifikována jako pozadí, pokud kardinalita, značená jako $\sharp$, průniku množiny vzorků v této kouli a množiny vzorků v modelu $\mathcal{M}(x)$ je větší nebo rovna danému prahu $\sharp_{min}$. Formálně je tedy $\sharp_{min}$ porovnávána s

$$\sharp\{S_R(v(x)) \cap \{v_1, v_2, \ldots, v_N\}\} \tag{2.2}$$

Z rovnice 2.2 vyplývá, že klasifikace hodnoty $v(x)$ v nejhorším případě obnáší výpočet N vzdáleností mezi $v(x)$ a vzorky modelu a N porovnání s prahovou vzdáleností R. Přesnost

modelu je tedy dána dvěma parametry: poloměrem koule R a minimální kardinalitou $\sharp_{min}$. Podobně citlivost modelu je dána poměrem

$$\frac{\sharp_{min}}{N} \tag{2.3}$$

Inicializace modelu pozadí

Vysvětlení tohoto procesu bývá často autory jiných metod opomíjeno. Typicky je potřeba několik desítek snímků, než je model pozadí jiných metod inicializován. Autoři ViBE však představují způsob, jak inicializovat model již z prvního snímku sekvence. Jelikož v prvním snímku není dostupná temporální informace o hodnotách pixelu, ViBE v tomto případě využívá prostorových informací – předpokládá, že sousedící pixely sdílí podobné temporální rozložení. Ve fázi inicializace je tedy model každého pixelu naplněn hodnotami náhodných sousedů v jeho osmiokolí. Formálně, pokud $t = 0$ indexuje první snímek a $N_G(x)$ je prostorové okolí pixelu x, pak

$$\mathcal{M}^0(x) = \{v^0(y \mid y \in N_G(x))\} \tag{2.4}$$

kde polohy y jsou zvoleny náhodně pomocí rovnoměrného rozložení pravděpodobnosti. Je samozřejmě možné, že daná hodnota $v^0(y)$ bude zvolena víckrát, nebo vůbec, ale to není problém, protože všechny pixely v okolí jsou vhodnými kandidáty.

Jednou z výhod tohoto způsobu inicializace je rychlé zotavení se z náhlé změny osvětlení scény. Pokud je změna osvětlení detekována jiným dodatečným algoritmem, stačí pouze znovu inicializovat model z nového snímku a detekce může pokračovat. Nevýhodou je možnost vzniku duchů při inicializaci. Duch je množina propojených bodů, které jsou detekovány jako by byly v pohybu, ale přitom nepřísluší žádnému reálnému pohybujícímu se objektu. V tomto případě pokud se v inicializačním snímku nachází pohybující se objekt, je považován za pozadí. Když se v následných snímcích pohne z původního místa a odkryje skutečné pozadí, je toto pozadí považováno za popředí, dokud se model postupně nenaučí správné hodnoty pomocí mechanizmu aktualizace pozadí.

Aktualizace modelu pozadí v čase

Klasifikační krok ViBE porovnává hodnoty pixelu $v^t(x)$ přímo se vzorky modelu pozadí náležícího předchozímu snímku $\mathcal{M}^{t-1}(x)$ v čase $t-1$. Je proto důležité rozhodnout, které vzorky mají být uloženy a na jak dlouho. Typický přístup je zahodit a přepsat staré vzorky po určitém čase nebo počtu snímků novými hodnotami. Tento přístup však nemusí být validní – není důvod přepsat naprosto relevantní hodnotu, která stále souhlasí s pozadím. ViBE místo toho určuje exponenciální monotónní snižování pravděpodobnosti, že vzorek při aktualizaci zůstane v modelu, tím, že pokud má být aktuální hodnota pixelu zahrnuta do modelu pozadí, vzorek, který má tato hodnota nahradit, je vybrán náhodně s rovnoměrným rozdělením pravděpodobnosti. ViBE navíc používá konzervativní strategii, takže pixely popředí nejsou pro aktualizaci nikdy použity. To má za následek náhodné podvzorkování vývoje hodnot pixelů pozadí v čase.

Samotné rozhodnutí, zda má být daný pixel v modelu pozadí aktualizován, závisí také na náhodě. Autoři zavádí parametr ϕ, tzv. update factor, který znamená to, že hodnota každého pixelu klasifikovaného jako pozadí bude s pravděpodobností $\frac{1}{\phi}$ použita pro aktualizaci modelu pozadí. Tím prakticky ovlivňuje celkovou životnost vzorků v modelu.

Při aktualizaci ViBE počítá také s předpokladem, na který spoléhá i při inicializaci – že hodnoty sousedících pixelů sdílí podobné temporální rozložení. Při aktualizaci vzorků pixelu je zvolen jeden soused, jehož náhodně zvolený vzorek je také nahrazen. Formálně pokud máme osmiokolí $N_G(x)$ pixelu x a bylo rozhodnuto aktualizovat množinu vzorků $\mathcal{M}(x)$ vložením $v(x)$, pak je také použita pro aktualizaci množiny vzorků $\mathcal{M}(y \in N_G(x))$ jednoho z pixelů v okolí, který je zvolen náhodně s rovnoměrným rozdělením pravděpodobnosti. Díky tomuto procesu je ViBE schopný zakomponovat nové nebo odebrané objekty pozadí postupnou difuzí od jejich hranic směrem ke středu. Tím se liší od jiných metod, kde je takové rozhodnutí většinou náhlé a binární. Přitom ViBE stále zůstává striktně konzervativním algoritmem. Pokud jsou tímto způsobem náhodou do modelu vloženy irelevantní informace, nevadí to, protože model obsahuje mnoho vzorků. Ačkoliv politika výběru vzorků ani jejich propagace není deterministická, v experimentech autorů metody se projevila jako velmi mocná.

Shrnutí

Následuje stručná rekapitulace práce algoritmu:

1. Přijetí nového snímku.
2. Inicializační fáze – ViBE sestaví paměť tak, aby si pro každý pixel uchoval sbírku N vzorků pozadí z předchozích snímků. Celý model je inicializován pouze z jednoho snímku – místo vzorků hodnot pixelů z minulosti dosadí do jejich historie hodnoty náhodně vybrané z jejich osmiokolí (včetně hodnoty daného pixelu).
3. Přijetí nového snímku.
4. Segmentační fáze – pro každý pixel aktuálního snímku je zjištěna euklidovská vzdálenost v barevném prostoru od příslušných vzorků z historie. Pokud vzdálenost alespoň $\sharp_{min}$ vzorků je menší než prahová, pixel je klasifikován jako pozadí, jinak jako popředí. Tím je vytvořena segmentační maska a předána k dalšímu zpracování.
5. Aktualizační fáze – s pixely klasifikovanými jako pozadí je provedeno následující: S pravděpodobností $\frac{1}{\phi}$ je náhodně vybráno, jestli aktuální pixel bude vybrán pro aktualizaci. Pokud ano, jeho hodnota přepíše náhodnou hodnotu v historii. Dále opět s pravděpodobností $\frac{1}{\phi}$ je vybráno, zda pixel přepíše náhodnou hodnotu v historii některého náhodně zvoleného souseda. Algoritmus pokračuje krokem 3.

Ohledně nastavení parametrů autoři doporučují $N = 20$, $\sharp_{min} = 2$ a $\phi = 16$. Pro šedotónové obrázky doporučují prahovou vzdálenost vzorků $R = 20$. V experimentech s nimi autoři měli nejlepší výsledky. Pokaždé by však tyto parametry, zejména ϕ, měly být přizpůsobeny konkrétní aplikaci.

ViBE vyniká svou rychlostí a svou účinností navzdory jednoduchosti. Mnoho dalších metod z něj vychází např. tím, že přidávají k vyhodnocení některé další příznaky. Mezi tyto metody se řadí např. LOBSTER [25], PBAS [15] nebo SubSENSE [26]. Drogenbroeck a Paquot navrhují vylepšení zvané ViBE+ [8], které zjednodušuje prahování při segmentaci, přidá do modelu pozadí 6 bajtů na pixel navíc, a navrhuje filtrovací funkce do fáze dodatečného zpracování segmentační masky. To vše umožňuje bezstarostné snížení ϕ na 5 a výrazně zlepšuje spolehlivost detekce.

Kapitola 3

Návrh detektoru změn v okolí 360°

3.1 Analýza a dekompozice problému

Cílem této práce je vytvořit aplikaci pro sestavu PT jednotky a kamery bez proměnlivé ohniskové vzdálenosti snímající viditelné spektrum. Aplikace má na základě polohy manipulátoru a obrazu z kamery zjišťovat sémanticky významné změny ve scéně, která je aktuálně snímána, a vyznačovat oblasti zájmu, ve kterých k těmto změnám došlo. Výsledný detektor musí pracovat v reálném čase a za co nejrychlejšího pohybu manipulátoru, to vše v rozsahu 360° horizontálně a alespoň 84° vertikálně. Přípravná fáze, justáž a učení se prostředí před samotným začátkem detekce je přípustná.

Problém hledání těchto změn by aplikace pracující se stacionární kamerou řešily použitím některého z mnoha algoritmů pro BGS, které byly představeny v sekci 2.2. Navrhuji tedy zvolit BGS algoritmus, který nejlépe vyhovuje mému účelu, a implementovat jeho rozšíření a rozhraní tak, aby dokázal přijímat data z kamery a PT jednotky a aby modeloval pozadí celého okolí HW sestavy. Proces výběru metody BGS je popsán v sekci 3.2.

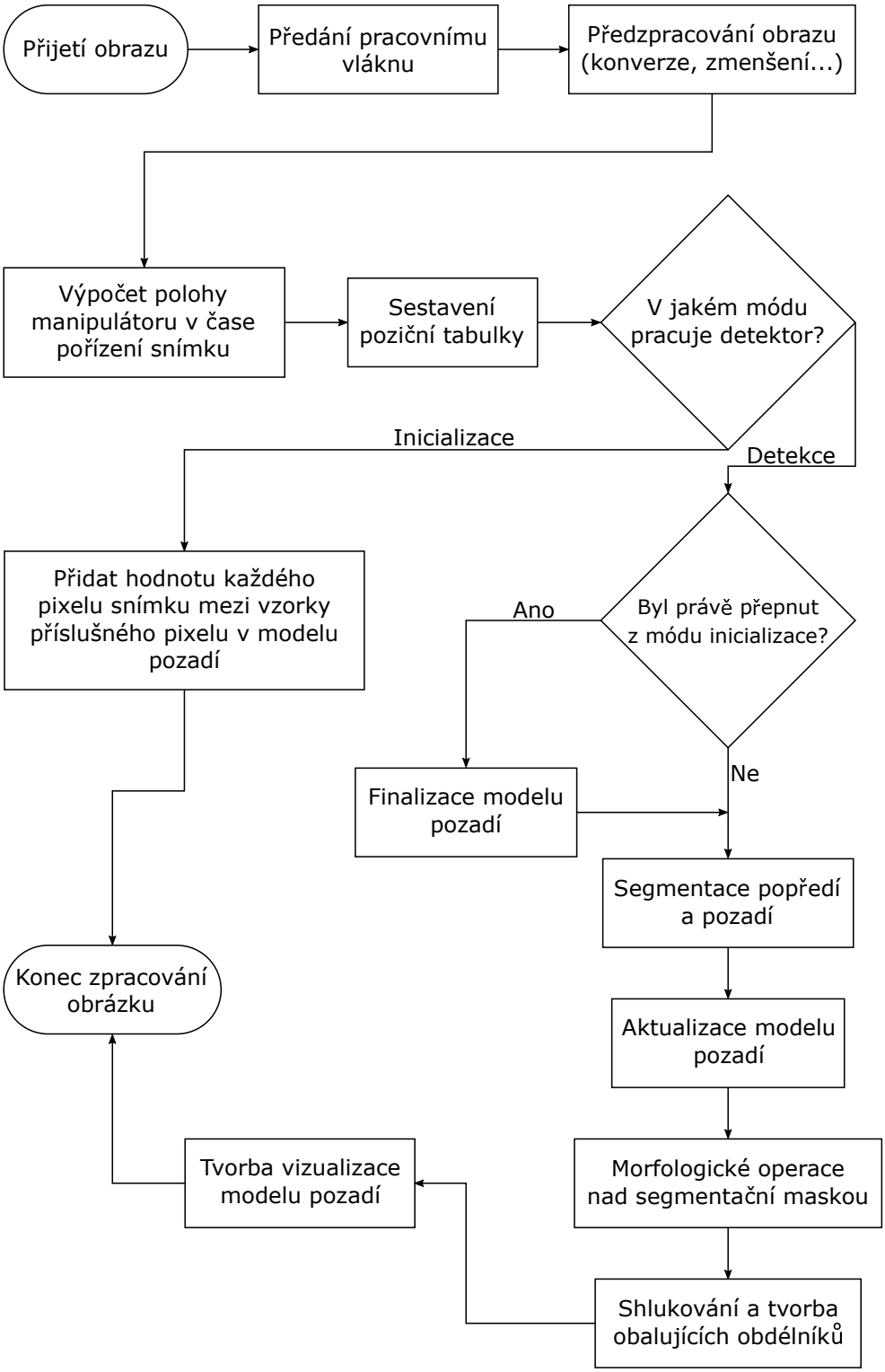
Pokud je modelováno pozadí pro všechny pixely v okolí, je nutné navrhnout mechanismus, který zprostředkuje operace BGS mezi pixely pozadí a pixely aktuálního snímku, zejména mapování mezi těmito dvojicemi pixelů. Vzhledem k paměťové náročnosti celého modelu pozadí je také nutné navrhnout mechanismus pro správu paměti, ve které je tento model uložen. Tyto záležitosti jsou probrány v sekci 3.3.

Jelikož kamera i PT jednotka budou dvě na sobě operačně nezávislá zařízení, nebudou data, která z nich budou přicházet, navzájem nijak časově synchronizována. Lze se spolehnout pouze na to, že každé zařízení bude zasílat informace s nějakou pevnou periodou, avšak při určitém nastavení ani to nemusí pro kameru platit. Vzhledem k tomu, že aplikace potřebuje znát co nejpřesnější polohu manipulátoru v době expozice snímku, je třeba navrhnout algoritmus pro dopočítání této polohy. Tento návrh je v sekci 3.4.

Návrh všech podčástí řešení je zakomponován do jednoho funkčního celku v sekci 3.5. Obsahuje popis modulů a běhu algoritmu. Ten je také ilustrován na obr. 3.1 – doporučuji se na něj obrátit pro získání celkového přehledu o podproblémech cílové aplikace.

3.2 Výběr vhodné metody odečítání pozadí

V této části vyhodnotím některé významné metody BGS podle jejich úspěšnosti v článcích, které je testují, náročnosti na výpočetní zdroje a jejich principiální vhodnosti pro cílovou aplikaci. Ne ke všem metodám je veřejně dostupný zdrojový kód, avšak tvůrci, kteří jsou



Obrázek 3.1: Vývojový diagram zpracování příchozího snímku.

	DB	CJ	IOM	Shadow	Therm.	Průměr
LOBSTER [25]	0,5679	0,7423	0,5770	0,8728	0,8248	0,717
PBAS [15]	0,6829	0,7220	0,5745	0,8143	0,7556	0,7099
ViBE+ [8]	0,7197	0,7538	0,5093	0,8302	0,6646	0,6955
ViBE [1]	0,5652	0,5995	0,5074	0,8342	0,6647	0,6342

Tabulka 3.1: Hodnoty F-Measure pro vybrané metody BGS a kategorie videosekvencí z datové sady CDnet 2012. Význam zkratek sloupců: DB – Dynamic Background, CJ – Camera Jitter, IOM – Intermittent Object Motion, Therm. – Thermal.

ochotni svůj kód zveřejnit, ho často zařazují do knihovny BGSLibrary [23]. Výběr v ní je široký a její metody mají v různých vyhodnoceních velmi dobré výsledky, proto omezím svůj výběr jen na některé z nich. Bohužel v ní chybí metody založené na neuronových sítích, které se začaly objevovat v posledních letech. Při vyhodnocování je pro mě důležitá spolehlivost metody, její náročnost na výpočetní zdroje, možnosti případné paralelizace a jednoduchost případných modifikací. Těmto kritériím se věnuji v následujících podsekcích.

Spolehlivost

Vyhodnocení metod z hlediska korektnosti se provádí statistickými výpočty nad rozdílem jejich výstupu (segmentační masky) a referenčních snímků (ground truth) datových sad. Tyto výpočty zahrnují např. podíl falešných pozitiv a falešných negativ, nicméně tou nejvíce souhrnnou je F-Measure počítaná jako

$$Pr = \frac{TP}{TP + FP} \tag{3.1}$$

$$Re = \frac{TP}{TP + FN} \tag{3.2}$$

$$\text{F-Measure} = 2\frac{Pr \cdot Re}{Pr + Re} \tag{3.3}$$

kde Pr je Precision, Re je Recall, TP je počet pravých pozitiv, FP je počet falešných pozitiv a FN je počet falešných negativ.

F-Measure tedy nabývá hodnot mezi 0 a 1, kdy vyšší číslo značí vyšší spolehlivost nebo úspěšnost. Hodnoty F-Measure vybraných metod na datových sadách CDnet jsou vidět v tabulkách 3.1 a 3.2. Z tabulek byly vypuštěny výsledky videosekvencí, které nejsou pro můj účel relevantní, jako PTZ (pohyb kamery na PT jednotce bez údajů o jejím pohybu). Tyto tabulky jsou výsledkem rešerše v příslušných vědeckých článcích autorů nebo souhrnných článcích porovnávajících různé metody BGS.

Při výběru vhodné metody bylo přihlédnuto i k dalším metrikám jako false negative rate (FNR), false positive rate (FPR), specificity (Sp) a percent wrong classification (PWC). Jejich význam je vysvětlen v rovnicích 3.4-3.7, které jsou k nalezení např. v [40]. Pro stručnost je zde však tyto výsledky nebudu uvádět.

	BW	NV	Turb.	DB	CJ	IOM	Th.	Pr.
SubSENSE [26]	0,8619	0,5599	0,7792	0,8177	0,8152	0,6569	0,8171	0,76
PAWCS [27]	0,8152	0,4152	0,6450	0,8938	0,8137	0,7764	0,8324	0,74
SOBS [19]	0,6370	0,4482	0,4702	0,6519	0,7150	0,5810	0,7140	0,6
KDE [9]	0,7571	0,4365	0,4478	0,5961	0,5720	0,4088	0,7423	0,57
AGMM [42]	0,7406	0,3960	0,4169	0,6328	0,5670	0,5325	0,6548	0,56
Codebook [18]	0,4	-	0,225	0,59	0,6	0,49	0,28	0,43

Tabulka 3.2: Hodnoty F-Measure pro vybrané metody BGS a kategorie videosekvencí z datové sady CDnet 2014. Význam zkratek sloupců: BW – Bad Weather, NV – Night Vision, Turb. – Turbulence, DB – Dynamic Background, CJ – Camera Jitter, IOM – Intermittent Object Motion, Th. – Thermal, Pr. – Průměr.

$$Sp = \frac{TN}{TN + FP} \tag{3.4}$$

$$FPR = \frac{FP}{FP + TN} \tag{3.5}$$

$$FNR = \frac{FN}{TP + FN} \tag{3.6}$$

$$PWC = 100 \cdot \frac{FN + FP}{TP + FP + TN + FN} \tag{3.7}$$

Náročnost na výpočetní zdroje

Narozdíl od vyhodnocení korektnosti metod, náročnost na výpočetní zdroje je v článcích zmíněna velmi vzácně. Rychlost výpočtu jsem posoudil empiricky pomocí GUI aplikace, která je součástí BGSLibrary. Ta ukazuje počet snímků za sekundu, které dokáže daná metoda zpracovat na CPU. Žádná z uvažovaných metod kromě ViBE nedokázala za sekundu zpracovat více než 25 snímků. Pro vyhodnocení možností paralelizace algoritmů bylo nutné pochopit jejich princip nebo si přečíst doporučení od jejich tvůrců.

Ukázalo se, že paměťová náročnost může být výrazný problém. Moje cílová aplikace vyžaduje vytvoření modelu pozadí (panoramatu) pro celé okolí v úhlu 360° horizontálně a 84° vertikálně. Pokud budu uvažovat kameru s rozlišením 2448×2048 px a zorným úhlem $9,4° \times 7,9°$, pak by bylo třeba modelovat 1,5 miliardy pixelů (přesný výpočet je uveden v sekci 3.3). Vzhledem k nutnosti ukládat si pro každý pixel data pro model pozadí, může celý model jednoduše překročit možnosti současných běžných pamětí RAM. Řešením by mohlo být držet v paměti pouze model pozadí pro snímek, který náleží aktuální pozici kamery v panoramatu, a jeho okolí a zbytek mít uložen na SSD. Výsledky výpočtů paměťové náročnosti jsou uvedeny v tabulce 3.3 na konci této sekce. Veličiny v ní uvedené jsou vypočítány pomocí rovnic 3.8-3.16:

$$R_{Fr} = w_{Fr} \cdot h_{Fr} \tag{3.8}$$

$$M_{Fr} = R_{Fr} \cdot M_{Px} \tag{3.9}$$

	M_{Px}: Počet bajtů na pixel	N	M_{Fr}: Pam. náročnost pro snímek 2480 × 2048 [MB]	M_{Pan}: Pam. n. pro celé panorama [GB]
Samotný obraz	3	1	14	4
ViBE	60	20	287	86
ViBE+	66	20	316	94
LOBSTER	72	35	344	103
PBAS	113	35	540	161
AGMM	120	—	574	171
SubSENSE	178	50	851	254
PAWCS	1050	50	5020	1500

	P: Jaká část panoramatu by se vešla do RAM [%]	α_H: Úhel záběru panoramatu v RAM – hor. [°]	α_V: Úhel záběru – vert. [°]
Samotný obraz	700 %	1372	1148
ViBE	35 %	69	57
ViBE+	32 %	62	52
LOBSTER	29 %	57	48
PBAS	19 %	36	31
AGMM	17 %	34	29
SubSENSE	12 %	23	19
PAWCS	2 %	4	3

	ω_H: Max. úhlová rychlost – horizontální [°/s]	ω_V: Max. úhlová rychlost – vertikální [°/s]	ω_B: Max. úhlová rychlost – oba směry [°/s]
Samotný obraz	1966	1652	898
ViBE	98	83	45
ViBE+	89	75	41
LOBSTER	82	69	37
PBAS	52	44	24
AGMM	49	41	22
SubSENSE	33	28	15
PAWCS	6	5	3

Tabulka 3.3: Výpočet paměťových nároků dle rovnic 3.8-3.16. Sloupec N značí počet minulých snímků, které si metoda potřebuje uchovat. Toto číslo je pouze ilustrační a do výpočtů se nepromítá. Výsledky výpočtů předpokládají rozlišení kamery ($w_{Fr} \times h_{Fr}$) 2480 × 2048, zorné pole ($H_{Fr} \times V_{Fr}$) $9,4° \times 7,9°$, počet pixelů v panoramatu (R_{Pan}) 1 534 398 240 (pro výpočet vizte sekci 3.3), velikost RAM (M_{RAM}) 32 GB a rychlost současného čtení a zápisu SSD (S) 3 GB/s.

kde w_{Fr} a h_{Fr} jsou po řadě šířka a výška obrazu z kamery v pixelech, R_{Fr} je rozlišení (počet pixelů) snímku, M_{Px} je paměťová náročnost jednoho pixelu modelu pozadí a M_{Fr} je paměťová náročnost modelu pozadí pro jeden snímek, dále:

$$P = 100 \cdot \frac{M_{RAM}}{M_{Pan}} \tag{3.10}$$

$$\alpha_H = \frac{w_{Fr}}{w_{Fr} + h_{Fr}} \cdot \frac{M_{RAM}}{M_{Pan}} \cdot 360 \tag{3.11}$$

$$\alpha_V = \frac{h_{Fr}}{w_{Fr} + h_{Fr}} \cdot \frac{M_{RAM}}{M_{Pan}} \cdot 360 \tag{3.12}$$

$$\tag{3.13}$$

kde P je procentuální část panoramatu, která by se do RAM vešla, M_{Pan} je velikost panoramatu v paměti (výpočet v sekci 3.3), M_{RAM} je velikost paměti RAM, α_H je úhel [°] v horizontálním směru, který by zabrala část panoramatu, která by se do paměti vešla, a přitom by byl zachován stejný poměr stran, jaký má obraz kamery a α_V je analogická veličina ve vertikálním směru, a nakonec:

$$\omega_H = \frac{S}{h_{Fr} \cdot \frac{w_{Fr}}{H_{Fr}} \cdot M_{Px}} \tag{3.14}$$

$$\omega_V = \frac{S}{w_{Fr} \cdot \frac{h_{Fr}}{V_{Fr}} \cdot M_{Px}} \tag{3.15}$$

$$\omega_B = \frac{S}{\left(h_{Fr} \cdot \frac{w_{Fr}}{H_{Fr}} + w_{Fr} \cdot \frac{h_{Fr}-1}{V_{Fr}}\right) \cdot M_{Px}} \tag{3.16}$$

kde S je rychlost SSD při současném čtení i zápisu dat v [B/s], ω_H je maximální úhlová rychlost, kterou by se manipulátor mohl pohybovat v horizontálním směru tak, aby aplikace stíhala zapisovat a číst části panoramatu z SSD, ω_V je analogická veličina pro pohyb v čistě vertikálním směru a ω_B je analogická veličina pro pohyb v obou směrech zároveň (dokonale šikmo), tedy nejhorší případ z hlediska množství načítaných dat. H_{Fr} a V_{Fr} značí horizontální a vertikální zorný úhel kamery.

Rovnice 3.11-3.16 počítají pouze přibližný výsledek, protože předpokládají, že projekce promítací plochy do paměti bude mít obdélníkový charakter, pro účely ilustrace však postačují. Tyto rovnice jsou použity pro výpočet výsledků v tabulce 3.3.

Po přezkoumání metod jsem pro další postup zvolil ViBE. Má ze všech metod nejnižší paměťové nároky, je principiálně velmi jednoduchá (mnoho ze zmíněných metod jsou ve skutečnosti jejími rozšířeními) a má uspokojivé výsledky jak dle objektivních kritérií, tak dle subjektivního pozorování v GUI aplikaci BGSLibrary. Tyto výsledky lze dále přizpůsobit pomocí vhodného nastavení parametrů metody pro konkrétní účel. Její implementace je volně dostupná a má dobrý potenciál pro paralelizaci. Princip této metody byl popsán v sekci 2.2.

3.3 Panorama

Panorama navržené v této sekci bude modelem pozadí pro metodu ViBE – místo pixelů v aktuálním snímku bude modelovat pixely v celém okolí manipulátoru. Pro tento účel je

třeba navrhnout mapování pixelů z promítací plochy kamery na kouli, a následně z této koule do paměti počítače.

Projekce na kouli

Cílem je promítnout souřadnice na promítací ploše (F_x, F_y) na sférické souřadnice (θ, ϕ) . Promítnutí souřadnic z promítací plochy na jednotkovou kouli lze provést vypočtením 3D souřadnic bodu P^S dle rovnice 3.17 (K je vnitřní matice kamery), převedením do homogenních souřadnic jako P^H a rotací kolem ohniska (středu jednotkové koule) dle rovnice 3.23. Rotace spočítaná dle rovnice 3.22 vychází přímo z aktuálního azimutu a a elevace e manipulátoru. Pokud je kamera otočená kolem normály promítací plochy, lze provést korekci pomocí z_{corr}, jinak se tato hodnota rovná 0.

$$P^S = [F_x, h_{Fr} - F_y, 1] \cdot K^{-1} \tag{3.17}$$

$$P^H = [P_1^S, P_2^S, P_3^S, 1] \tag{3.18}$$

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(e) & -sin(e) & 0 \\ 0 & sin(e) & cos(e) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.19}$$

$$R_y = \begin{bmatrix} cos(a) & 0 & sin(a) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(a) & 0 & cos(a) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.20}$$

$$R_z = \begin{bmatrix} cos(z_{corr}) & sin(z_{corr}) & 0 & 0 \\ -sin(z_{corr}) & cos(z_{corr}) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.21}$$

$$R = R_y \cdot R_x \cdot R_z \tag{3.22}$$

$$P^R = R \cdot P^H \tag{3.23}$$

Sestavení rotačních matic předpokládá pravoruký souřadný systém s osou z kolmou k promítací ploše a orientovanou směrem ke snímané scéně. Nakonec z bodu P^R lze spočítat sférické souřadnice dle rovnic 3.24 a 3.25. Tento proces je ilustrován na obr. 3.2.

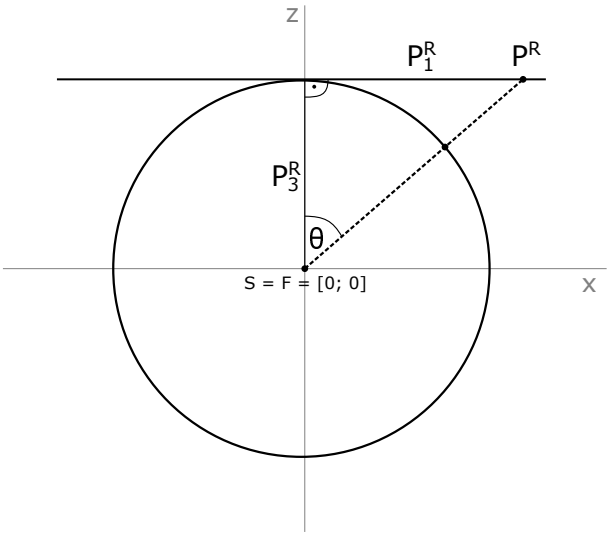
$$\theta = \arctan\left(\frac{P_1^R}{P_3^R}\right) \tag{3.24}$$

$$\phi = \arctan\left(\frac{P_2^R}{\sqrt{(P_1^R)^2 + (P_3^R)^2}}\right) \tag{3.25}$$

Rovnice 3.24-3.25 a další panoramatické projekce popisují Szelinski a Shum ve své práci [29].

Projekce do paměti

Dalším problémem je mapování sférických souřadnic (θ, ϕ) do paměti počítače a volba struktury této paměti. V oboru kartografie existuje mnoho mapovacích funkcí s různými

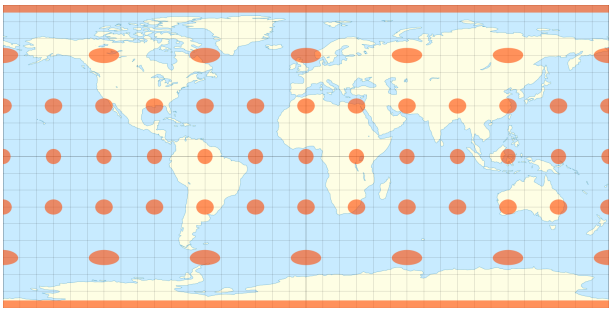


Obrázek 3.2: Ilustrace výpočtu sférické souřadnice (úhlu θ) z bodu na orotované promítací ploše.

druhy zkreslení, tato zkreslení by však byla pro můj účel nevítaná. Příklad je uveden na obr. 3.3, kde je nutné horizontálně roztahovat oblasti blízké pólům, aby výsledná mapa byla obdélníková. Kvůli diskrétní povaze promítací plochy kamery a paměti počítače by však došlo k tomu, že sousední pixely v promítací ploše by se nepromítly na sousední pixely v paměti – při statické poloze kamery by v paměti vznikaly neinicializované „díry“ (čím blíže k pólům, tím větší a častější by byly). U jiných projekcí by se naopak sousední pixely v promítací ploše promítly do stejného pixelu v paměti. Válcová projekce (na obdélník), která je charakteru počítačové paměti nejpřirozenější, by byla právě kvůli těmto „děrám“ velmi neúsporná.

Navrhuji tedy paměťový model uspořádaný do různě dlouhých řádků pixelů. Každý řádek obaluje promítací kouli kolem jejího obvodu, ale každý v jiné výšce. Řádky tedy budou rozvržené do rovnoběžek této projekční koule. Ilustrace paměťového uspořádání je na obr. 3.4. Počet pixelů v řádku kolem rovníku (w_{Pan}) závisí na rozlišení a zorném úhlu kamery:

[2] J. Kunimune, CC BY-SA 4.0, dostupné z https://commons.wikimedia.org/w/index.php?curid=66467577.



Obrázek 3.3: Ekvidistantní válcová projekce zemského povrchu s Tissotovými rozptyly zkreslení[2], které zobrazují zkreslení kružnic s nekonečně malým poloměrem promítnutých z kulového modelu Země na mapu. Čím jsou kruhové oblasti blíže pólům, tím více jsou na mapě zkreslené.

$$l_{Proj} = 2 \cdot z \ \cdot tan\left(\frac{H_{Fr}}{2}\right) \tag{3.26}$$

$$l_{Px} = \frac{l_{Proj}}{w_{Fr}} \tag{3.27}$$

$$\alpha_E = \frac{H_{Fr}}{2} - \arctan\left(\frac{\frac{l_{Proj}}{2} - l_{Px}}{z}\right) \tag{3.28}$$

$$\alpha_M = \arctan\left(\frac{l_{Px}}{z}\right) \tag{3.29}$$

$$w_{Pan} = 2\pi \div \frac{w_E \cdot \alpha_E + w_M \cdot \alpha_M}{w_E + w_M} \tag{3.30}$$

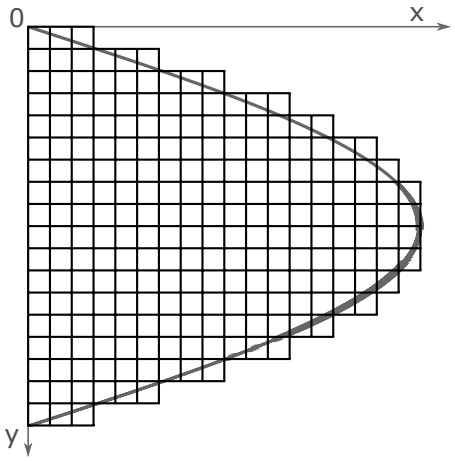
kde l_{Proj} značí délku promítací plochy, l_{Px} značí délku pixelu na této ploše a α_E značí úhel zabraný krajním pixelem na promítací ploše, α_M úhel zabraný prostředním pixelem promítací plochy a w_E a w_M jsou vhodně zvolené váhy. Hodnota z může být libovolná nenulová, protože se v rovnicích vykrátí. Analogicky lze získat i počet řádků modelu. Výpočet indexu řádku p_y a pixelu v tomto řádku p_x ze sférických souřadnic je poté jednoduše:

$$p_y = \left(-\phi + \frac{V_{Pan}}{2}\right) \cdot \frac{h_{Pan}}{V_{Pan}} \tag{3.31}$$

$$p_x = \theta \cdot \frac{w_{Pan}}{H_{Pan}} \cdot cos(\phi) \tag{3.32}$$

Rovnice 3.30 prakticky počítá vážený průměr hodnot α_E a α_M. Vzhledem k nelineární povaze funkce arkus tangens v rovnici 3.28 a kosinus v rovnici 3.32 nelze obecně najít takové w_E a w_M, aby se sousední pixely na promítací ploše vždy promítly na sousední pixely v paměti. Lze ovšem najít takové hodnoty, které zajistí alespoň to, že se žádné dva pixely z promítací plochy nepromítnou do stejného pixelu v paměti – pouze občas jeden pixel přeskočí. Problém těchto děr řeším v sekci 3.5 dvěma způsoby: rozdělením běhu algoritmu na inicializační a detekční fázi a interpolací hodnot pixelů vkládaných do děr.

V sekci 3.2 byl pro použití v mé aplikaci vybrán ViBE jakožto metoda BGS. Aktualizace modelu pozadí ViBE však vyžaduje propagaci hodnot aktuálního pixelu do vzorků jeho



Obrázek 3.4: Uspořádání pixelů modelu pozadí v paměti počítače naznačené černou mřížkou. Osami x a y je naznačen způsob indexace. Délka každého řádku má kosinovou závislost na délce nejdelšího řádku, jak naznačuje šedá křivka v pozadí.

sousedů. Hledání souřadnic souseda není ale s mnou navrženým modelem pozadí triviální. Máme-li absolutní souřadnice výchozího pixelu v panoramatu p_x a p_y a relativní souřadnice cílového pixelu p_x^R a p_y^R, pak lze spočítat absolutní souřadnice cíle p'_x a p'_y jako:

$$p'_y = p_y + p_y^R \tag{3.33}$$

$$\phi = -\left(p_y \cdot \frac{V_{Pan}}{h_{Pan}}\right) + \frac{V_{Pan}}{2} \tag{3.34}$$

$$w_R = \lceil \cos(\phi) * w_{Pan} \rceil \tag{3.35}$$

$$\phi' = -\left(p'_y \cdot \frac{V_{Pan}}{h_{Pan}}\right) + \frac{V_{Pan}}{2} \tag{3.36}$$

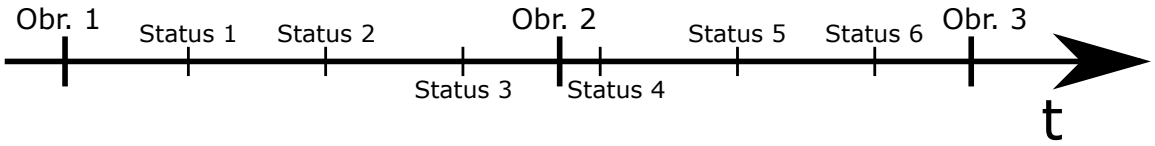
$$w'_R = \lceil \cos(\phi') * w_{Pan} \rceil \tag{3.37}$$

$$p'_x = \left\lfloor (p_x + p_x^R) \cdot \frac{w_R}{w'_R} \right\rfloor \tag{3.38}$$

kde ϕ a w_R jsou po řadě sférická výška a pixelová délka řádku výchozího pixelu a ϕ' a w'_R jsou sférická výška a délka řádku cílového pixelu.

Dlaždice

Jak bylo demonstrováno tabulkou 3.3, model pozadí může zabírat přes 80 GB, což by znamenalo, že jako souvislý obraz by se nemusel vejít do běžné paměti RAM. Proto navrhuji seskupit pixely v paměti do buněk pravidelné mřížky – dlaždic, kdy každá dlaždice bude souvislá obdélníková část panoramatu. Pak postačí, když v paměti RAM budou pouze dlaždice v okolí aktuální polohy manipulátoru, zbytek může být uložen na SSD. Při pohybu manipulátoru budou dlaždice v jeho trajektorii preemptivně načítány dříve, než je jejich obsah potřeba, a analogicky budou nepotřebné dlaždice ukládány.



Obrázek 3.5: Ilustrace různých případů pořadí příchodu obrázků a statusů. Pro výpočet polohy manipulátoru při příchodu obr. 1 je nutné provést extrapolaci od statusu 1 směrem do minulosti. Pro obr. 2 je to interpolace mezi statusy 3 a 4. Pro obr. 3 je to extrapolace od statusu 6 směrem do budoucnosti.

Vzhledem k navrženému paměťovému modelu dává největší smysl, aby dlaždice měly výšku 1 pixel, čímž se z nich stanou jednoduše řádky hodnot. Výpočet souřadnic v dané dlaždici je poté

$$t_x = \lfloor p_x \div w_T \rfloor \tag{3.39}$$
$$t_y = p_y \tag{3.40}$$
$$o_x = p_x \ \% \ w_T \tag{3.41}$$

Symboly t_x a t_y značí indexy dlaždic v horizontálním a vertikálním směru, w_T značí šířku dlaždice v pixelech, a o_x je offset v rámci dlaždice, tedy potřebný index pro adresování pixelu v této dlaždici. Operátor % značí operaci modulo.

3.4 Časová nesynchronicita pořízení snímku a hlášení o poloze pan tilt jednotky

Z PT jednotky lze očekávat pravidelná hlášení o její aktuální poloze a rychlosti (statusy). Nezávisle na těchto statusech může aplikace kdykoliv přijmout obrázek z kamery a potřebuje dopočítat, v jaké poloze byl manipulátor, když byl tento obrázek pořízen. Navrhuji vytvořit dočasnou paměť, ve které bude uložených několik posledních statusů. Z hlediska doby příchodu snímku můžou nastat tři možné situace: Snímek je starší než nejstarší uložený status, je novější než nejnovější status, nebo byl pořízen v době mezi nejstarším a nejnovějším. Tyto situace jsou ilustrovány na obr. 3.5.

Nejmenší perioda statusů pro manipulátor, který jsem měl k dispozici při testování, byla 20 ms. Pro tento interval postačí lineární interpolace polohy, tedy pokud φ_0 je úhel hlášený statusem v čase t_0, φ_1 je úhel hlášený statusem v čase t_1, pak lze spočítat úhel φ_{Fr} v čase t_{Fr}, kdy $t_0 \leq t_{Fr} \leq t_1$, jako:

$$\varphi_{Fr} = \varphi_0 + (\varphi_1 - \varphi_0) \cdot \frac{t_{Fr} - t_0}{t_1 - t_0} \tag{3.42}$$

Pokud je nutná extrapolace, pak extrapoluji zrychlení, ze kterého dopočítám změnu rychlosti a z ní úhel uražený od času posledního statusu. Tedy pokud ω_0 je úhlová rychlost v čase t_0 a ω_1 je úhlová rychlost v čase t_1 a ε je úhlové zrychlení, pak lze spočítat úhel φ_{Fr} v čase t_{Fr}, kdy $t_0 < t_1 \leq t_{Fr}$, jako:

$$\varepsilon = \frac{\omega_1 - \omega_0}{t_1 - t_0} \tag{3.43}$$

$$\omega_{Fr} = \omega_0 + \varepsilon \cdot (t_{Fr} - t_1) \tag{3.44}$$

$$\varphi_{Fr} = \varphi_1 + (\omega_1 \cdot (t_{Fr} - t_1)) + \frac{(\omega_{Fr} - \omega_1) \cdot (t_{Fr} - t_1)}{2} \tag{3.45}$$

Rovnice 3.42-3.45 platí jak pro výpočet azimutu, tak elevace.

3.5 Architektura a běh systému

V sekci 3.3 bylo vysvětleno, že někdy nastane případ, že sousední pixely ve snímku z kamery nejsou namapovány na pixely v paměti, které spolu sousedí, ale bývají mezi nimi skoky, čímž vznikají celé oblasti neinicializovaných pixelů. Hodnoty pixelů v těchto oblastech by se daly interpolovat z hraničních pixelů (pokud nějaké existují), avšak pokud by se tento postup prováděl pro každý pixel, byl by poměrně složitý a výpočetně náročný. Vzhledem k tomu, že zadání úlohy připouští přípravnou fázi aplikace, rozhodl jsem se rozdělit její vykonávání do dvou módů:

- Inicializace – probíhá plnění dlaždic daty z aktuálního snímku.
- Detekce – probíhá odečítání pozadí a aktuálního snímku. Model pozadí se aktualizuje jen tam, kde byl předem inicializován.

Mezi těmito módy přepíná uživatel nebo jiná aplikace. Tímto rozdělením módů ztrácí aplikace výhodu inicializace celého pozadí z jednoho snímku, kterou má ViBE, ale ušetří se tak výkon při samotné detekční fázi, která je kritická.

Z důvodu tohoto rozdělení běhu obsahují dlaždice dvě matice:

- Pozadí – pixely reprezentující část řádku modelu pozadí včetně N vzorků ze své historie.
- Inicializační matice – matice čísel inicializovaných na hodnotu 0, kdy každému pixelu (nikoliv vzorku) pozadí náleží jedno číslo. To značí, kolik vzorků již bylo pro tento pixel vloženo a zároveň index pro vložení dalšího vzorku.

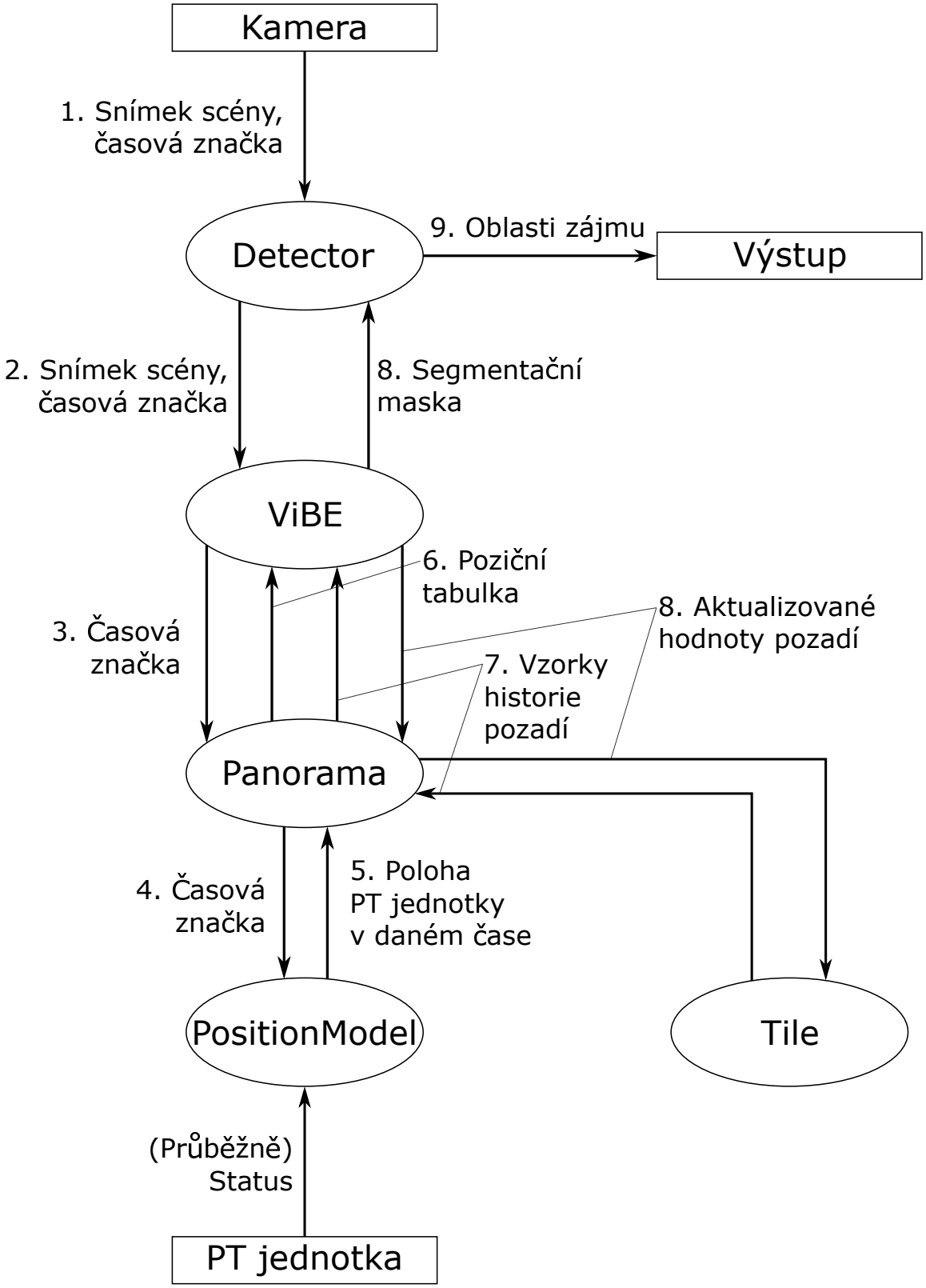
Při inicializačním módu detektoru jsou postupně pro každý pixel přidávány vzorky do jeho historie z hodnot v aktuálním obrázku. Index vzorku je určen hodnotou příslušné buňky v inicializační matici, poté je tento index inkrementován. Může se stát, že bude přepnut mód detektoru z inicializace na detekci dříve, než budou všechny vzorky pro některý pixel zaplněné. Proto existuje přechodná finalizační fáze. V ní je do všech neinicializovaných vzorků zkopírována hodnota některého z inicializovaných.

Pro urychlení výsledné aplikace je vhodné při příchodu nového snímku sestavit **poziční tabulku**. Ta mapuje souřadnice pixelů v aktuálním snímku na konkrétní dlaždice a pixely v nich. Jsou v ní tedy předpočítané hodnoty pomocí rovnic 3.17-3.25, 3.31-3.32 a 3.39-3.41. Tuto tabulku je třeba přepočítat při příchodu nového snímku, ale pouze pokud se od pořízení předchozího snímku pozice PT jednotky změnila.

Nyní by návrh běhu programu na obr. 3.1 měl být plně srozumitelný. Navrhuji tedy rozdělit program na následující význačné moduly:

- Detector – postará se o přijetí obrázku z kamery, jeho případné předzpracování (konverze, zmenšení), shlukování v segmentační masce získané od ViBE a její publikování jiným aplikacím. Také v něm bude probíhat případná vizualizace procesu BGS.
- ViBE – bude implementovat funkcionalitu algoritmu ViBE – segmentaci a aktualizaci pozadí.
- Panorama – spravuje model pozadí a polohy aktuálního snímku v něm (funkcionalita je popsána v sekci 3.3). Komunikuje s moduly PositionModel a Tile.
- PositionModel – dopočítává polohu manipulátoru v době pořízení snímku, jak je navrženo v sekci 3.4.
- Tile – představuje jednotlivé dlaždice, jak byly popsány v sekci 3.3.

Tok dat mezi moduly je znázorněn na obr. 3.6. Diagram tříd praktické implementace je vidět dále na obr. 4.1.



Obrázek 3.6: Diagram toku dat mezi moduly programu.

Kapitola 4

Optický radar

Tato kapitola popisuje programovou realizaci návrhu v kapitole 3, stejně jako podpůrných programů a nástrojů pro testování její funkcionality. V sekci 4.1 popisuji nástroje a knihovny použité při implementaci výsledných aplikací. Dále v sekci 4.2 uvádím přehled všech ROS balíků, které jsem vytvořil nebo využil pro běh aplikace a testování. Jádro celé této práce, ROS uzel pro segmentaci popředí a pozadí v okolí kamery, je podrobně popsáno v sekci 4.3.

4.1 Použité nástroje a knihovny

Existující implementace ViBE je dostupná např. jako součást Background Subtraction Library [23] v jazyce C/C++. Tuto implementaci jsem převzal a pro své účely přetvořil na C++ třídu. Pro zprostředkování komunikace s manipulátorem, kamerou a jakoukoliv jinou aplikací, která by mohla výstup optického radaru využít, je velmi vhodný C++ framework Robot Operating System (ROS), který umožňuje právě zasílání zpráv (libovolných dat) mezi aplikacemi a mnoho dalších užitečných funkcí a možností. ROS je představen v mé bakalářské práci [7], na oficiálních stránkách[1] a v tutoriálech[2]. Pro implementaci aplikací jsem tedy použil ROS Kinetic pro Linux Ubuntu 16.04 (a příbuzné systémy) a pro zpracování obrazu knihovnu OpenCV verze 3.3.2[3].

Výraznou součástí ROS je 3D grafický a fyzikální simulátor Gazebo. Gazebo dokáže simulovat svět specifikovaný souborem ve formátu SDF, který je odvozený od XML. Tento soubor zpravidla obsahuje popis pozičních, vizuálních, mechanických a kolizních vlastností objektů ve světě, typicky analyticky nebo inkluzí 3D modelů, textur atd. Umožňuje také specifikovat senzory a kinematické řetězy – fyzické články spojené různými druhy kloubů. Tyto klouby lze ovládat posíláním ROS zpráv na příslušné topicy, zatímco senzory mohou pravidelně publikovat vygenerovaná data na své topicy. Tím Gazebo umožňuje kompletní simulaci širokého spektra robotů bez nutnosti jakkoliv upravovat aplikace, které data ze senzorů zpracovávají nebo naopak robota ovládají. Další informace lze nalézt online[4].

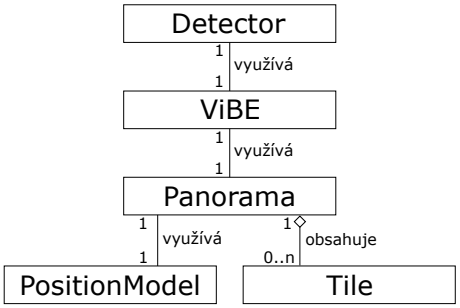
[1] Dostupné na http://www.ros.org/about-ros/
[2] Dostupné na http://wiki.ros.org/ROS/Tutorials
[3] Dostupné na https://opencv.org/
[4] Na adrese http://gazebosim.org/

4.2 Přehled ROS balíků a uzlů

Součástí této práce je 24 ROS balíků, které obsahují 7 spustitelných uzlů. V této sekci je uveden jejich stručný přehled. Klíčové uzly budou podrobněji popsány v následujících kapitolách. Pokud není explicitně řečeno jinak, byly balíky vytvořeny mnou. Zdrojové kódy některých uzlů nebudou v této práci zveřejněny a aplikace tak budou dostupné pouze v binární formě. Balíky v této práci jsou následující:

- `detector` – jádro celé práce. Přijímá ROS zprávy z kamery a z manipulátoru a provádí segmentaci pozadí a popředí. Práce uzlu `detector` je popsána v sekci 4.3.
- `detector_eval` – uzel pro vyhodnocení detektoru, jak je popsáno v kapitole 5. Přijímá segmentační masky a statusy manipulátoru z ROS topiců a provádí na nich příslušné výpočty.
- `display_img` – obsahuje jednoduchý uzel, který přijímá obraz z kamery a zobrazuje ho.
- `env_simulation` – balík se SDF soubory (přípona `.world`), které popisují statické prvky třech různých simulačních světů. Dynamické prvky (dron, manipulátor) jsou popsány v jiných balících a vkládány do těchto světů za běhu. Obsahuje také 3D modely a textury statických prvků ve světech.
- `hector_quadrotor_flyover` – výsledný uzel dokáže vygenerovat předem naprogramovanou cestu a vysílat potřebné zprávy pro kvadrokoptéru Hector.
- Balíky s předponou `hector_` – komponenty potřebné pro realistickou simulaci kvadrokoptéry Hector v Gazebu. Dostupné online[5] a popsané v článku autorů [20].
- `keyboard` – výsledný uzel vytvoří prázdné SDL okno, čte stisky kláves klávesnice, pokud je okno ve fokusu, a publikuje je jako ROS zprávy. Dostupný online[6].
- `human_interfaces` – jednoduchý uzel napsaný v jazyce Python, který čte zprávy o stisku kláves klávesnice a překládá je na povely pro manipulátor.
- Balíky s předponou `image_` – poskytují rozhraní pro pohodlné posílání a přijímání obrázků jako ROS zpráv. Dostupné online[7].
- `mso2a_description` – obsahuje 3D modely a xacro popis robotického manipulátoru MSO-2/A od společnosti EVPÚ Defence a kamery v něm umístěné. Xacro je skriptovací mechanismus pro ulehčení tvorby URDF popisu robota[8]. URDF (Unified Robot Description Format) je XML formát pro reprezentaci robotů, podobný SDF[9].
- `ols` – obsahuje `.launch` soubory, které agregují spuštění několika uzlů pro dosažení určitého výsledku, jako třeba spuštění Gazeba s přednastaveným světem, funkčním manipulátorem a kvadrokoptérou.

[5] Na adrese http://ros.org/wiki/hector_quadrotor
[6] Na adrese http://wiki.ros.org/keyboard
[7] Na adrese http://wiki.ros.org/image_transport
[8] Více informací na http://wiki.ros.org/xacro
[9] Více informací na http://wiki.ros.org/urdf



Obrázek 4.1: Zjednodušený diagram tříd uzlu `detector`.

- `ols_messages` – definice všech vlastních ROS zpráv a service požadavků (soubory `.srv`) použitých v této práci.
- `pt_gazebo` – obsahuje dílčí `.launch` soubory pro spuštění simulace manipulátoru nebo pro jeho vložení do existující simulace.
- `pt_driver` – uzel z tohoto balíku se stará o ovládání reálného či simulovaného manipulátoru MSO-2/A. Přijímá zprávy s instrukcemi pro manipulátor a pravidelně vysílá údaje o jeho aktuální poloze. Zdrojový kód není v této práci zveřejněn.
- `pt_controller` – uzel spravuje oprávnění pro ovládání manipulátoru. Přijímá instrukce pro manipulátor a předává mu pouze ty, které pochází od zařízení, které má na ovládání právo. Zdrojový kód není zveřejněn.

4.3 Detektor

Jakožto jádro této práce, detektor pravidelně přijímá dva druhy ROS zpráv – status, tedy polohu a rychlost manipulátoru v azimutu a elevaci, a obrázek z kamery – a vysílá zprávy s polem obdélníků, které ve snímku značí oblast zájmu, ve které došlo k sémanticky významné změně. Pro účely demonstrace je aplikace také schopná vizualizovat několik kroků svého vykonávání. V této sekci popíšu funkcionalitu jeho jednotlivých stavebních bloků (C++ tříd). Na obr. 4.1 je vyobrazen jednoduchý diagram tříd této aplikace a v tabulce 4.1 je přehled topiců, kterými tento uzel komunikuje s ostatními. Na obr. 3.1 je vývojový diagram algoritmu zpracování obrázku, který bude popsán dále.

Hlavní funkce

Mechanismus ROSu, který zajišťuje, že uzel pravidelně přijímá zprávy a zpracovává je v příslušné callback funkci, se nazývá spinning[10]. Spinner je nekonečná smyčka, která čte obsah fronty příchozích zpráv a reaguje na ně. Zprávy jsou zpracovávány sekvenčně dle času jejich příchodu, ale vzhledem k náročnosti zpracovávání obrázků by bylo blokováno zpracování mnohem jednodušších statusů manipulátoru a časových značek pořízení snímků, což není vyhovující. Proto jsou při spuštění aplikace ve funkci `main()` vytvořeny dva asynchronní spinnery, kdy každý provádí spinning ve vlastním vlákně. Jeden z nich vyzvedává snímky, které jsou uloženy ve zvlášť vyhrazené frontě, druhý zpracovává všechny ostatní zprávy, které přicházejí do globální (implicitní) fronty.

[10] Více informací na http://wiki.ros.org/roscpp/Overview/Callbacks%20and%20Spinning.

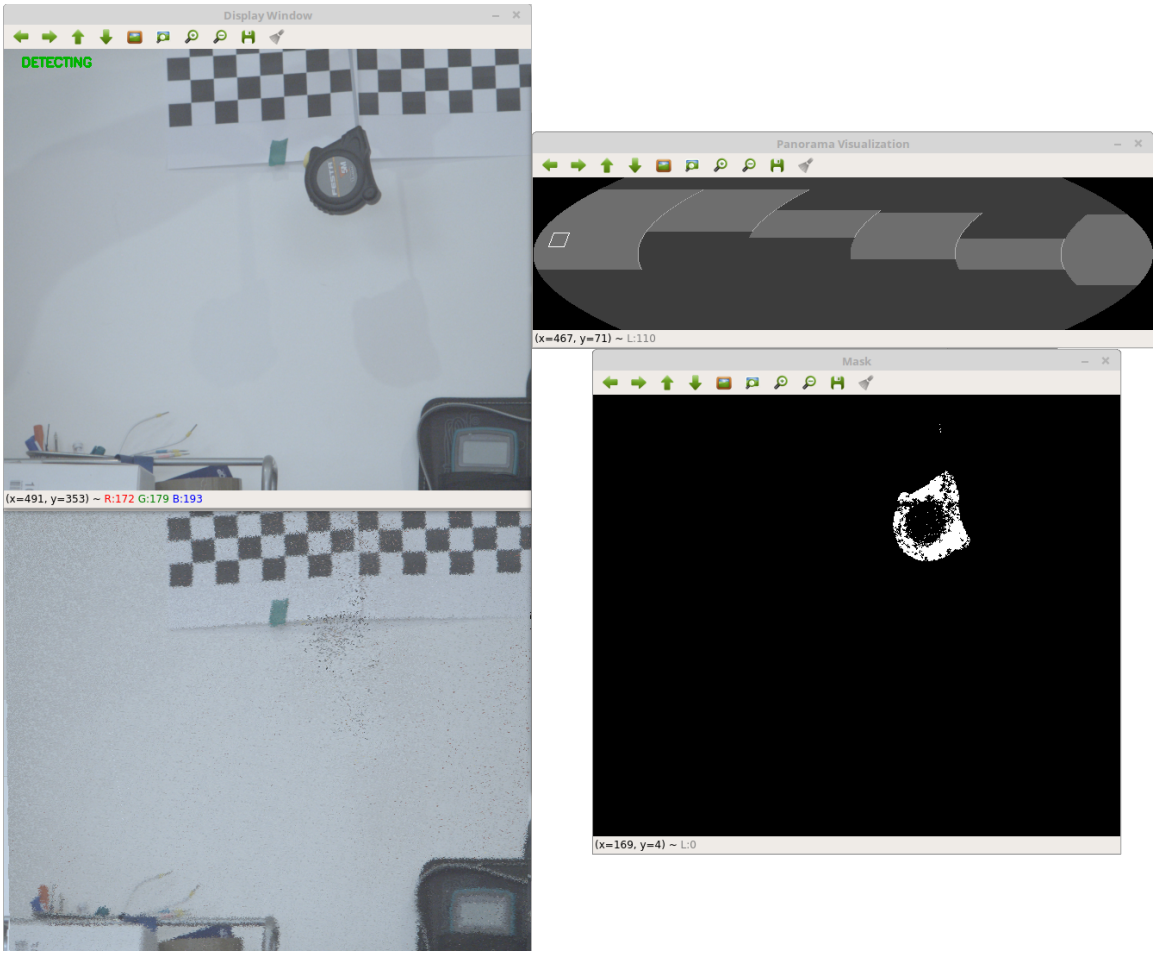
Název topicu	Definice zprávy	Význam
`/pt0/camera/image`	sensor_msgs/Image	Obraz z kamery ve standardním formátu ROSu.
`/pt0/camera/frame_timestamp`	ols_messages/OLSFrameTimestamp	Skutečný čas pořízení obrazu kamery. Volitelné. Pokud není informace dostupná, bere se jako čas pořízení časová značka zprávy se samotným obrázkem.
`/pt0/driver1/status`	ols_messages/OLSPTDriverStatus	Status manipulátoru (rychlost a poloha v azimutu a elevaci) posílaný každých 20 ms.
`/detector/mode`	ols_messages/OLSDetectorMode	Přepínání módu detektoru mezi inicializací a detekcí.
`/detector/foreground`	ols_messages/OLSForeground	Pole obdélníků ohraničujících oblasti, kde bylo detekováno popředí.

Tabulka 4.1: Seznam významných topiců uzlu `detector`. Názvy topiců předpokládají jmenný prostor manipulátoru `/pt0`, jmenný prostor kamery `/pt0/camera` a jmenný prostor detektoru `/detector`.

Kromě inicializace běží v hlavní funkci vizualizační smyčka. Ta iteruje nad několika obrázky třídy `ProtectedImage`, což jsou běžné OpenCV obrázky obohacené o mutex (obrázky jsou aktualizovány v jiných vláknech) a příznak, zda byly aktualizovány. Pokud některý od poslední iterace aktualizován byl, je vizualizován v příslušném okně pomocí `cv::imshow()`. Příklad takové vizualizace je vidět na obr. 4.2. Na ní levé horní okno ukazuje aktuální snímek, kde je zachycen pohybující se svinovací metr, levé spodní okno ukazuje první vzorek každého pixelu modelu pozadí v prostoru aktuálního snímku. Pravé spodní okno ukazuje segmentační masku. Dle černých pixelů v zobrazení modelu pozadí a segmentační masce tam, kde je metr, je vidět, jak se metr postupně prolíná do modelu pozadí. Pravé horní okno ukazuje stav panoramatu. V tomto okně černá barva je pouze výplň, tmavě šedá představuje neinicializované dlaždice, světle šedá barva dlaždice s inicializovanými pixely, bílý lichoběžník ukazuje aktuální polohu snímku v rámci celého panoramatu a velmi světle šedé křivky zvýrazňují horizontální hranice mezi dlaždicemi.

Třída Detector

Hlavní třída `Detector` zpracovává v callback funkci `modeCb()` zprávy, které přepínají mód detektoru. Ve funkci `tsCb()` zpracovává zprávy OLSFrameTimestamp. Obě tyto informace se uloží pro pozdější zpracování. Veškerá práce probíhá ve funkci `imageCb()` při přijetí nového snímku z kamery. Ten se předá pracovnímu vláknu, které vykoná zbytek práce ve funkci `processImg()`. Tam se po předzpracování obrazu (konverze, zmenšení) algoritmus větví podle aktuálně nastaveného módu. Při módu inicializace se volá funkce `initializeBG()` třídy `ViBE`, při detekci její funkce `detect()`. Výsledkem detekce je segmentační maska, se kterou jsou provedeny morfologické operace eroze a následné dilatace. Nyní může být provedeno shlukování ve funkci `partition()` a kolem shluků jsou vytvořeny obalující obdélníky, které jsou následně publikovány. Poté je segmentační maska uložena do příslušné instance `ProtectedImage`, aby ji hlavní vlákno mohlo později vizualizovat. Dále jsou vydány příkazy



Obrázek 4.2: Příklad vizualizace detektoru.

pro třídu `ViBE`, aby pomocí funkcí `visualizePanorama()` a `visualizeBGModel()` vytvořila vízualizace obsazenosti dlaždic v panoramatu, aktuální polohy snímku v panoramatu a modelu pozadí v oblasti snímku.

Třída ViBE

Třída `ViBE` obsahuje zejména funkcionalitu stejnojmenné metody BGS. Při inicializaci této třídy jsou vygenerována tři pole náhodných hodnot o šířce $5 \cdot w_{Fr} + 419$:

- Hodnoty relativních souřadnic souseda
- Náhodné indexy vzorků historie
- Rozhodnutí, zda má být pixel aktualizován (dle update factoru)

Dopředné vygenerování a uložení těchto hodnot šetří výpočetní čas při detekci. Šířka polí byla zvolena tak, aby nedocházelo k tvorbě nežádoucích artefaktů v modelu pozadí z důvodu příliš častých nebo pravidelně se opakujících vygenerovaných hodnot. Pětinásobek šířky řádku zajistí, že pole bude vždy dostatečně velké a 419 jakožto prvočíslo zajistí, že se nebudou artefakty opakovat pravidelně napříč různými řádky. V dřívějším stádiu vývoje

aplikace, kdy byla velikost polí zvolena méně vhodně, docházelo např. k tomu, že byly pravidelně aktualizovány právě takové pixely pozadí, aby tvořily při vizualizaci modelu pozadí šikmé přímky v pravidelných intervalech.

Pokud je mód detektoru nastaven na inicializaci, volá detektor funkci `initializeBG()` v této třídě. Funkce se skládá ze dvou kroků a oba dva spadají do funkcionality třídy `Panorama`: `setFramePosition()` a `fillBG()`.

Pokud je mód nastaven na detekci, volá se opět `setFramePosition()` třídy `Panorama` a poté probíhá detekce očekávaným způsobem: segmentace ve funkci `segment()` a aktualizace modelu pozadí ve funkci `updateModel()`. Obě tyto operace využívají funkce OpenCV `Mat::forEach()` pro paralelní zpracování jednotlivých pixelů.

Pro čtení a zápis správných vzorků z panoramatu používá `ViBE` příslušné funkce třídy `Panorama`: `isPixelInitialized()`, `getPixelHistoryPtr()` a `replace()`.

Třída Tile

Instance této třídy jsou stavební bloky celého panoramatu – spravuje je instance třídy `Panorama`. Obsahují dvě OpenCV matice:

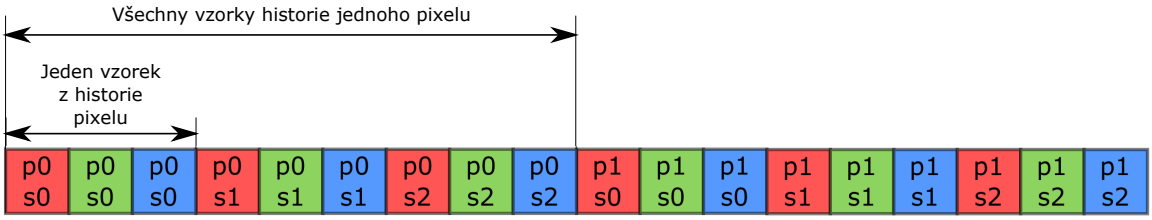
- Pozadí – pixely reprezentující část řádku modelu pozadí včetně N vzorků ze své historie. Paměťové rozložení dat v této matici je naznačeno na obr. 4.3.
- Inicializační matice – matice čísel inicializovaných na hodnotu 0, kdy každému pixelu (nikoliv vzorku) pozadí náleží jedno číslo. To značí, kolik vzorků již bylo pro tento pixel vloženo a zároveň index pro vložení dalšího vzorku.

Při inicializačním módu detektoru je zavolána metoda `Panorama::fillBG()`, která postupně pro každý pixel přidává vzorky historie pomocí funkce `Tile::addSample()` – ta vloží barevnou hodnotu do vzorku s indexem, který je uložený v příslušné buňce inicializační matice, poté tento index inkrementuje. Může se stát, že bude přepnut mód detektoru z inicializace na detekci dříve, než budou všechny vzorky pro některý pixel zaplněné. Proto existuje přechodná finalizační fáze, při které se pro všechny dlaždice zavolá metoda `finalize()`. V ní je do všech neinicializovaných vzorků zkopírována hodnota některého z inicializovaných. Pokud jsou všechny pixely dlaždice plně inicializované, je celá inicializační matice smazána z paměti, čímž uvolní místo.

Kromě zmíněných patří k významným členským metodám také `replace()`, která přepíše hodnotu konkrétního vzorku konkrétního pixelu novou hodnotou, a `getPixelHistoryPtr()` která vrací ukazatel na celou historii konkrétního pixelu. Metoda `isInitialized()` ověřuje, že pro daný pixel existuje alespoň jeden vzorek.

Třída Panorama

Tato třída má přehled o stavu modelu pozadí a pozici aktuálního snímku v něm. Stará se také o mapování a převod mezi různými systémy souřadnic, které jsou v aplikaci použity. Mezi tyto převodní funkce patří `frameCoords2Panorama()`, která převádí souřadnice z prostoru snímku do prostoru panoramatu dle rovnic 3.17-3.32. Dále `isPixelInitialized()`, `getPixelHistoryPtr()` a `replace()` přijímají souřadnice v prostoru panoramatu, přepočítají je do prostoru dlaždic dle rovnic 3.39-3.41 a zavolají příslušné metody odpovídající instance třídy `Tile`, kterým předají pouze index pixelu v rámci této dlaždice dle rovnic 3.39-3.41. Dlaždice jsou uspořádány do dvourozměrného pole, avšak pokud ještě nebyl inicializován žádný pixel v dané dlaždici, není vůbec konstruována.



Obrázek 4.3: Ilustrace uspořádání dat v dlaždici o šířce 2 px a výšce 1 px, kdy jsou pro každý pixel uchovávány 3 vzorky historie ($N = 3$). Pixely jsou označeny písmenem `p` a číslicí značící index tohoto pixelu. Vzorky jednotlivých pixelů jsou označeny písmenem `s` a číslem. Barvy pozadí buněk značí barevné složky vzorku.

Klíčovou funkcí třídy `Panorama` je `setFramePosition()`. Ta nejdřív nastaví azimut a elevaci, ve které se manipulátor nacházel při pořízení snímku, pomocí stejnojmenné funkce třídy `PositionModel`. Když zná pozici, může sestavit poziční tabulku. V té mají být pro každý pixel v prostoru kamerového snímku uloženy jeho souřadnice v prostoru panoramatu (a dlaždic). Jsou v ní tedy předpočítány výsledky funkce `frameCoords2Panorama()`.

Bohužel nebyla implementována funkcionalita průběžného ukládání a načítání dlaždic z pevného disku.

Třída PositionModel

Úlohou instance této třídy je výpočet polohy manipulátoru v kterémkoliv okamžiku v blízké minulosti. Za tímto účelem neustále přijímá ROS zprávy o statusu manipulátoru. Povelem k výpočtu polohy je právě volání funkce `setFramePosition()`, které je předána časová značka, ke které má polohu vypočítat. Tento výpočet je popsán v sekci 3.4.

Je nutné znát co nejpřesnější čas pořízení snímku, ideálně okamžik na půli cesty mezi začátkem a koncem jeho expozice. To je úkol ovladače kamery, který by měl tuto časovou značku posílat jako zprávu typu OLSFrameTimestamp. Ovladač manipulátoru navíc každých 20 ms posílá zprávy o statusu, které taky mají časovou značku. Tyto statusy jsou ukládány do kruhového bufferu – instance třídy `CircularBuffer` – který má uložených posledních 50 statusů. Dle časové značky snímku je pak možné v tomto bufferu vyhledat status, který bezprostředně předcházel nebo následoval jeho pořízení, nebo v případě extrapolace dva nejnovější, případně nejstarší, statusy. Výpočet polohy v době pořízení snímku je popsán v sekci 3.4.

Dodatek

Práce detektoru je závislá na mnohých parametrech. Některé lze zjistit a uložit předem, jako vnitřní matice kamery a její jiné parametry, jiné je ale třeba ručně přizpůsobit konkrétní aplikaci detektoru. Parametrům metody ViBE, zejména update factoru a prahovému rozdílu hodnot pixelů, je třeba věnovat zvláštní pozornost.

Některé parametry uzlů lze nastavit za běhu pomocí mechanismů ROSu pro dynamickou rekonfiguraci. Tu zprostředkovává balík `dynamic_reconfigure`[11]. Pokud je aplikace spuštěná, lze otevřít grafické uživatelské rozhraní (GUI) pro nastavení těchto parametrů pomocí příkazu `rosrun rqt_reconfigure rqt_reconfigure`. GUI obsahuje seznam aktu-

[11] Více informací na http://wiki.ros.org/dynamic_reconfigure

álně spuštěných a rekonfigurovatelných uzlů. Pokud je některý vybrán, zobrazí se všechny parametry, které lze nastavit. Rekonfigurovatelnými uzly jsou `detector` a `pt_driver`.

Vzhledem k omezené míře optimalizace a paralelizace výkonu nepracuje detektor v reálném čase, zejména při vysokém rozlišení příchozích snímků. Pro práci v reálném čase je kritické převést výpočty zpracování obrazu na GPGPU. Aktuální stav tedy slouží spíš pro demonstraci správnosti konceptu, který byl v této práci navržen, a bude vyžadovat další práci.

Kapitola 5

Testování a experimenty

Cílem této kapitoly je především experimentálně vyhodnotit korektnost mapovacích funkcí navržených v sekci 3.3 a řešení časové nesynchronicity pořízení obrázku a statusů z PT jednotek, které je popsané v sekci 3.4. Kromě toho budu pozorovat i chování modelu pozadí, které se kvantifikuje obtížněji. Samotný ViBE byl již mnohokrát vyhodnocen jinými vědeckými týmy. Vzhledem k tomu, že mé modifikace ViBE jsou minimální – změnila se pouze funkce pro hledání souseda popsaná rovnicemi 3.33-3.38 – nebudu ho testovat.

V sekci 5.1 popisuji testovací scénář a způsob vyhodnocení jeho výsledků. Sekce 5.2 obsahuje návrh a rozbor konkrétní realizace scénáře. Výsledky testů uvádím a diskutuji v sekci 5.3. Nakonec v sekci 5.4 uvádím další scénáře, které jsou určeny spíše pro demonstraci funkcionality.

5.1 Metodika

Protože jsem nenalezl žádné datové sady pro odečítání pozadí, které by obsahovaly ground truth a zároveň informace o poloze manipulátoru, vyvstává nutnost vytvořit si vlastní datovou sadu. Navrhuji následující scénář: Manipulátor hýbe kamerou přes scénu, která je plná hran, ale obsahuje pouze pozadí. Žádná část scény se nebude pohybovat. V prvním běhu se bude manipulátor otáčet pomalu – cca $7°/s$, ve druhém svou maximální rychlostí $70°/s$.

Výhody tohoto přístupu jsou následující: Díky apriorní znalosti, že snímky datové sady obsahují pouze pozadí, není potřeba složitě vytvářet ground truth snímky – všechny pixely těchto snímků jsou černé. Dle výsledků předběžných testovacích běhů aplikace však ve výstupní segmentační masce nějaké bílé pixely v oblastech hran obrazu očekávám. Pokud se objeví, tento scénář umožní najít jejich příčinu, kterou může být buď špatná projekce a mapování pixelů, nebo chyba v odhadu polohy kamery, např. kvůli zavádějícím časovým značkám ROS zpráv. Rozlišit mezi nimi půjde zejména podle toho, jestli se bílé pixely budou objevovat pouze za pohybu manipulátoru a jestli s jeho rostoucí úhlovou rychlostí vzroste jejich četnost.

Navrhuji tedy čtyři scénáře:

1. Rychlost $7°/s$, žádné morfologické operace.
2. Rychlost $7°/s$, morfologické operace.
3. Rychlost $70°/s$, žádné morfologické operace.
4. Rychlost $70°/s$, morfologické operace.

Morfologickými operacemi se myslí operace jednoho cyklu eroze a třech cyklů následné dilatace, obojí s oknem o velikosti 3×3. Ty jsou typickým krokem dodatečného zpracování snímku, zejména pokud záleží pouze na přibližné poloze pohybujícího se objektu, jako tomu je v mém případě.

Pro vyhodnocení byly použity metriky Specificity dle rovnice 3.4 a Percent Wrong Classification (PWC) dle rovnice 3.7. Vzhledem k nulovému počtu pravých pozitiv nemají ostatní metriky uvedené v sekci 3.2 smysl.

Každý scénář bude vyhodnocen také ze třech aspektů: První vezme do úvahy všechny snímky, druhý pouze ty, které byly pořízeny, když se manipulátor nehýbal, a třetí pouze ty, které byly pořízeny, když manipulátor příliš nezrychloval nebo nezpomaloval, tedy jeho absolutní hodnota zrychlení v každém směru je menší než 0.005 rad/s^2. Z tohoto důvodu průjezdy v testovacích scénářích budou mít pravidelné přestávky – okamžiky, kdy manipulátor stojí.

5.2 Testovací prostředí

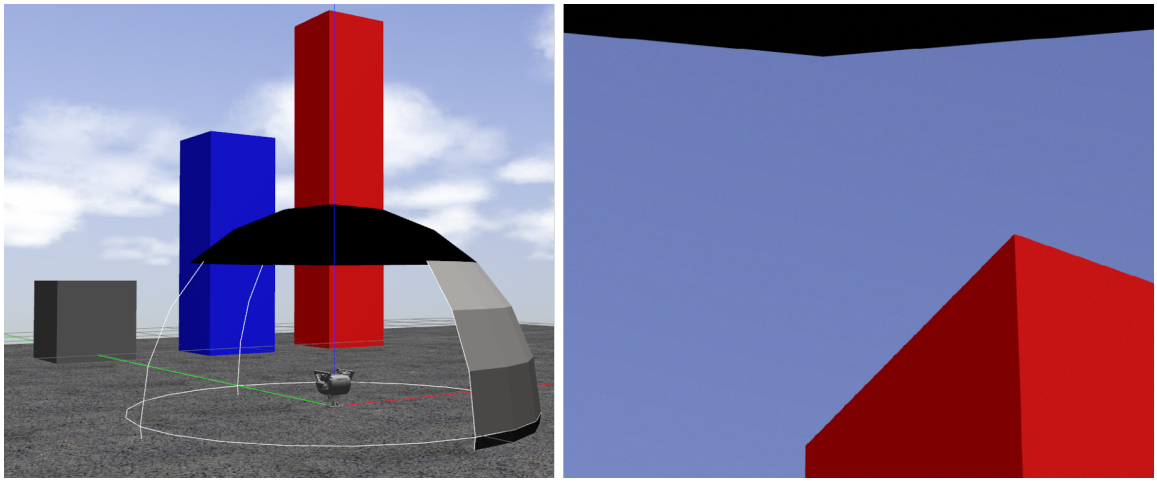
Svou aplikaci jsem vyvíjel a testoval především na PT jednotce MSO-2/A od společnosti EVPÚ Defence[1] a kameře Blackfly S Color 5.0 MP GigE Vision od společnosti FLIR[2]. Práce s tímto HW však obnáší několik neduhů, které by mohly narušovat vyhodnocení korektnosti mé metody:

- Mapovací rovnice 3.17-3.23 nepočítají s translací kamery vůči osám otáčení manipulátoru (to je jeden z námětů pro budoucí práce).
- Pokud je kamera vyosená, pak se snižující se vzdáleností snímaných objektů roste chyba mapovací funkce. Vzhledem k tomu, že průsečík os otáčení manipulátoru se nachází v jeho vnitřku, není možné do ní kameru umístit.
- Zjištění času pořízení snímku z kamery se projevilo jako problematické. Tento čas je potřeba znát nejlépe s přesností na desítky mikrosekund, ale přestože kamera má mechanismy např. pro zjištění časového okamžiku konce expozice, její časové údaje nebyly uspokojivě přesné.
- Obdobně je obtížné zjistit přesný časový okamžik, kterému náleží statusy přicházející od PT jednotky, protože TCP komunikace s ní přináší neurčité zpoždění.
- Je nutné velmi přesně znát vnitřní matici používané kamery. Kalibrační program OpenCV se však v tomto ohledu projevil jako nedostatečně přesný a spolehlivý.
- Čočka kamery způsobuje zkreslení v ohledu geometrie i jasu obrazu, zejména na okraji obrazu.

Místo testování s reálným HW tedy volím kontrolované prostředí simulátoru Gazebo, které si lze přizpůsobit dle libosti. Toto prostředí je vidět na obr. 5.1. Je v něm umístěn model manipulátoru MSO-2/A, jehož pohybové možnosti jsou naprogramovány tak, aby co nejvíce odpovídaly možnostem reálného manipulátoru. O jeho kontrolu a komunikaci s ostatními

[1] Více informací na adrese http://www.evpudefence.com/cs/produkty/manipulatory/mso-2-a

[2] Více informací na adrese https://www.ptgrey.com/blackfly-s-50-mp-color-gige-vision-sony-imx264



Obrázek 5.1: Vlevo: Pohled na simulační prostředí. Vpravo: Obraz z kamery při pohledu na vrch červeného kvádru. Nahoře je vidět kus černé kulové úseče, která náleží kouli obalující manipulátor.

Manipulátor MSO-2/A	
Rozsah pohybu v azimutu	$n \times 360°$
Rozsah pohybu v elevaci	$\pm 40°$
Max. rychlost v azimutu	$70°/s$
Max. rychlost v elevaci	$30°/s$
Přesnost	$\pm 0,5$ mrad

Kamera	
Rozlišení	1024×820 px
Zorné úhly	$10° \times 8,02°$
Frekvence snímání	20 FPS

Tabulka 5.1: Parametry simulovaného HW použitého při testování.

uzly se stará uzel `pt_driver` ze stejnojmenného ROS balíku. Definice celého modelu je v balíku `mso2a_description`. Ten obsahuje 3D modely částí manipulátoru a jeho URDF popis, který je dynamicky generovaný pomocí XML makro jazyka Xacro[3]. Společně s manipulátorem je v balíku definovaná i kamera, která je umístěná v průsečíku os otáčení manipulátoru. Parametry manipulátoru i kamery jsou uvedené v tabulce 5.1.

Kolem manipulátoru je umístěno něco co připomíná drátový model koule s plnou vrchní úsečí černé barvy. „Dráty“ této koule díky své tenkosti a blízkosti ke kameře opravdu ozkouší pixelovou přesnost mapovací funkce, zatímco vrchní úseč otestuje správnost mapování v nejvyšší dosažitelné elevaci manipulátoru. Dále je v simulačním světě asfaltový podklad, na kterém vedle sebe opodál stojí tři kvádry: šedý o výšce 1 m, modrý o výšce 3 m a červený o výšce 5 m. Oproti obloze vytvářejí v obraze velmi kontrastní hrany. Namíření kamery na vrch každého z nich má demonstrovat správnost mapování při pohybu manipulátoru ve všech směrech a nezávisle na elevaci.

Nahrál jsem tedy dva `rosbag` soubory – první realizuje scénáře 1 a 2, druhý scénáře 3 a 4. Ve všech případech se bude jednat o různé přejezdy kamerou přes oblast třech kvádrů a následně černé úseče na vrchu koule. Tyto přejezdy

Veškeré testování bylo vyhodnoceno na počítači s procesorem Intel Core i5 3350P (4 jádra, 3,2 GHz), grafickou kartou s čipem NVIDIA 660 Ti (využívá ji Gazebo) a 8 GB

[3]Více informací na adrese http://wiki.ros.org/xacro

	N	FP	TN	Sp	PWC
Scénář 1 – celkem	704	339 332	562 297 468	0,999397	0,060311
Scénář 1 – zastavený	112	24 701	89 485 699	0,999724	0,0275957
Scénář 1 – stálý směr	704	339 332	562 297 468	0.999397	0.060311
Scénář 2 – celkem	480	1 107 784	382 508 216	0,997112	0,288774
Scénář 2 – zastavený	74	52 148	59 088 652	0,999118	0,088176
Scénář 2 – stálý směr	224	402 458	178 618 342	0,997752	0,224811
Scénář 3 – celkem	696	166 526	556 076 674	0,999701	0,0299376
Scénář 3 – zastavený	110	6325	87 905 675	0,999928	0,00719469
Scénář 3 – stálý směr	651	144 580	520 134 620	0,999722	0,0277889
Scénář 4 – celkem	476	1 088 789	379 330 411	0,997138	0,286208
Scénář 4 – zastavený	74	9 692	59 131 108	0,998193	0,016388
Scénář 4 – stálý směr	223	322 073	177 899 527	0,998193	0,180715

Tabulka 5.2: Výsledky testovacích scénářů uvedených v sekci 5.1. N značí počet vyhodnocených snímků, FP značí celkový počet falešných pozitiv, TN pravých negativ, Sp je Specificity a PWC je Percent Wrong Classification.

paměti RAM. Software byl spouštěn v operačním systému Linux Mint 18.02 64-bit, který je založen na Linux Ubuntu 16.04.1.

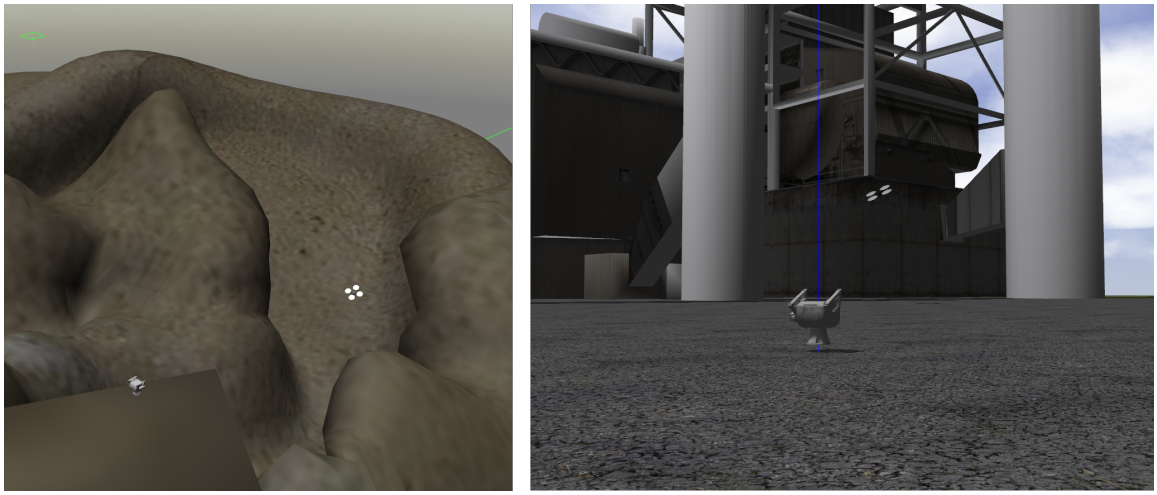
Parametry algoritmu ViBE při testování jsou následující: $N = 20$, $\sharp_{min} = 2$ a $\phi = 5$. Kromě toho se Gazebo projevilo jako nespolehlivé v ohledu správných časových značek statusů od manipulátoru. Bylo třeba najít magickou konstantu, která může být odečtena od časové značky všech statusů. Tuto konstantu jsem hledal a ladil ručně – její hodnota je 0.00951 s. I s ní však dochází k tomu, že spočítaná poloha manipulátoru v čase pořízení snímku mírně nesouhlasí s reálnou. Automatický proces, který by takovou konstantu našel (rozhodně bude potřeba i u HW zařízení), je námětem pro budoucí práci.

5.3 Výsledky experimentů

Numerické výstupy testování jsou vidět v tabulce 5.2. Ve všech případech se objevily bílé pixely na hranách mezi hranoly a oblohou. Jejich četnost však výrazně klesla, pokud se braly v úvahu pouze snímky, kdy je manipulátor zastavený. Částečně také klesla, pokud se ignorovaly snímky během prudkého zastavení nebo rozjetí manipulátoru. Naopak zase stoupla v případě scénáře 2 a 4. Použití morfologických operací zjevně situaci zlepšuje, zejména při nízkých rychlostech manipulátoru.

Z výsledků lze jednoznačně vyvodit, že problém toho, aby časová značka odeslaných ROS zpráv odpovídala času skutečného pořízení dat, která s sebou nesou, je kritický. Pokud se manipulátor hýbe dostatečně rychle, je problém znatelný, pokud se časová značka liší od skutečného času v řádech stovek mikrosekund.

Problém falešných pozitiv se také částečně řeší procesem aktualizace pozadí ViBE. Při stacionárním kameře lze sledovat rychlý úbytek bílých pixelů v segmentační masce, které vznikly tím, že kamera byla při inicializaci daného místa zrovna v pohybu. Problém by se



Obrázek 5.2: Vlevo: Pohled na simulační prostředí terénu s manipulátorem a kvadrokoptérou prolétající skrz zakřivený kaňon. Vpravo: Pohled na simulační prostředí města s manipulátorem a prolétající kvadrokoptérou.

tedy z velké části dal vyřešit ponecháním detektoru v módu detekce a pomalým projetím celé oblasti, kterou je třeba skenovat. Tím se prolnou vzorky pozadí jednotlivých pixelů mezi vzorky svých sousedů a správné pozicování polohy manipulátoru do časového okamžiku již nebude tak přísné.

Dá se tedy s jistotou prohlásit, že samotné mapovací funkce a model pozadí pracují korektně. Pouze je třeba nastolit jiné mechanismy pro zpřesnění polohy kamery v rámci modelu pozadí. Za tímto účelem byl testován například algoritmus, který počítá optický tok mezi klíčovými body po sobě následujících snímků, aby z něj odhadl translaci nového snímku oproti předchozímu. To však s sebou přináší několik problémů: Výpočet rotace kamery z translace snímku není triviální a je třeba důmyslný mechanismus, který dokáže rozhodnout, ve které situaci je vhodné použít polohu vypočítanou ze statusů, kdy použít polohu vypočítanou z optického toku, zda použít jejich průměr atd. Kandidátem na tento mechanismus je Kalmanův filtr, ovšem to je námětem pro budoucí práce.

5.4 Další demonstrační prostředí

Simulační prostředí popsané v sekci 5.2 se nachází v balíku `env_simulation` a spouští se souborem `sphere.launch`. V tomto balíku se však nachází dvě další prostředí, kde v obou figuruje kvadrokoptéra Hector [20] – realisticky ovladatelný dron, kterému jsem naprogramoval několik tras v balíku `hector_quadrotor_flyover`. Prvním prostředím je malý úsek města, ve kterém nad manipulátorem v elipsovité dráze létá Hector. To se spouští souborem `city_flyover_1.launch`. Druhé prostředí je kopcovitý terén, ve kterém má Hector vytyčené 3 možné dráhy. Jejich výběr je proveden spuštěním příslušného souboru `terrain_flyover_1.launch`, `terrain_flyover_2.launch`, nebo pro složitější průlet kaňonem `terrain_flyover_canyon.launch`. Podrobnosti o spuštění jsou uvedeny v souboru `README.txt` na přiloženém DVD. Obě prostředí jsou zachycena na obr. 5.2.

Kapitola 6

Závěr

Cílem této práce bylo vytvořit aplikaci, která dokáže využít obraz z kamery umístěné na pan tilt jednotce a informace o poloze této jednotky pro segmentaci sémanticky významných změn ve svém okolí v rozsahu 360° horizontálně a alespoň 84° vertikálně. Výsledný detektor měl pracovat v reálném čase a za co nejrychlejšího pohybu manipulátoru.

Za tímto účelem jsem nastudoval a v kapitole 2 popsal hardware, který se na podobné úlohy typicky používá. Také jsem v ní popsal problematiku odečítání pozadí a shrnul přístupy některých dosavadních metod. V kapitole 3 jsem vybral jednu z metod odečítání pozadí a navrhl jsem její rozšíření tak, aby korektně fungovala nezávisle na rotaci kamery a to i přes svou vysokou paměťovou náročnost. Také řeším nesynchronnost obrázků a hlášení o poloze manipulátoru.

Funkcionalita konečné implementace do značné míry splňuje cíl, jak je popsáno v kapitole 4 a dokázáno v kapitole 5. Vzhledem k omezené míře optimalizace a paralelizace výkonu aplikace nepracuje v reálném čase, zejména při vysokém rozlišení příchozích snímků. Pro práci v reálném čase je kritické převést výpočty zpracování obrazu na GPGPU. Aktuální stav tedy slouží spíš pro demonstraci správnosti konceptu, který byl v této práci navržen, a bude vyžadovat další práci. Aplikace je závislá na nastavení mnoha parametrů – některé z nich lze detekovat automaticky, např. parametry kamery, jiné je ale nutné přizpůsobit ručně konkrétní aplikaci celého systému.

Mezi další možná rozšíření patří rektifikace snímku (odstranění zkreslení a vinětace způsobených objektivem kamery), implementace vylepšení metody ViBE, změna barevného prostoru snímků a jeho případná redukce, odstranění rozmazání způsobeného pohybem kamery během expozice snímku nebo širší spektrum automaticky nastavujících se parametrů aplikace. Také mohou být implementovány mechanismy pro zpřesnění odhadu polohy manipulátoru nebo pro určení zpoždění časových značek ROS zpráv vůči času skutečného pořízení dat.

Literatura

[1] Barnich, O.; Droogenbroeck, M. V.: ViBE: A powerful random technique to estimate the background in video sequences. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2009, ISSN 1520-6149, s. 945–948, doi:10.1109/ICASSP.2009.4959741.

[2] Barnich, O.; Droogenbroeck, M. V.: ViBe: A Universal Background Subtraction Algorithm for Video Sequences. *IEEE Transactions on Image Processing*, ročník 20, č. 6, 2011: s. 1709–1724, ISSN 1057-7149, doi:10.1109/TIP.2010.2101613.
URL http://orbi.ulg.ac.be/bitstream/2268/145853/1/Barnich2011ViBe.pdf

[3] Brutzer, S.; Höferlin, B.; Heidemann, G.: Evaluation of background subtraction techniques for video surveillance. In *Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2011, ISBN 978-1-4577-0395-9, ISSN 1063-6919, s. 1937–1944, doi:10.1109/CVPR.2011.5995508.

[4] Caris, M.; Johannes, W.; Stanko, S.; aj.: Millimeter wave radar for perimeter surveillance and detection of MAVs (Micro Aerial Vehicles). In *2015 16th International Radar Symposium (IRS)*, červen 2015, ISSN 2155-5745, s. 284–287, doi:10.1109/IRS.2015.7226314.

[5] Codruta, I.; Vasileb, G.; Corneliu I., T.; aj.: A fast algorithm for background tracking in video surveillance, using nonparametric kernel density estimation. *Facta universitatis – series: Electronics and Energetics*, ročník 18, 2005: s. 127–144, ISSN 0353-3670, doi:10.2298/FUEE0501127I.

[6] Cucchiara, R.; Grana, C.; Piccardi, M.; aj.: Detecting moving objects, ghosts, and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 25, č. 10, říjen 2003: s. 1337–1342, ISSN 0162-8828, doi:10.1109/TPAMI.2003.1233909.

[7] Daniel Senčuch: *Modul pro otevření dveří pro PR2*. Bakalářská práce, FIT VUT v Brně, 2015.

[8] Droogenbroeck, M. V.; Paquot, O.: Background subtraction: Experiments and improvements for ViBe. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, IEEE, 2012, ISSN 2160-7508, s. 32–37, doi:10.1109/CVPRW.2012.6238924.

[9] Elgammal, A.; Duraiswami, R.; Harwood, D.; aj.: Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE*, ročník 90, č. 7, červenec 2002: s. 1151–1163, ISSN 0018-9219, doi:10.1109/JPROC.2002.801448.

[10] Elgammal, A.; Harwood, D.; Davis, L.: Non-parametric Model for Background Subtraction. In *Computer Vision – ECCV 2000*, editace D. Vernon, Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, ISBN 978-3-540-45053-5, s. 751–767.

[11] Goyette, N.; Jodoin, P.-M.; Porikli, F.; aj.: *ChangeDetection.NET (CDNET)*. [Online; navštíveno 20.01.2018].
URL http://changedetection.net/

[12] Goyette, N.; Jodoin, P. M.; Porikli, F.; aj.: Changedetection.net: A new change detection benchmark dataset. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, červen 2012, ISBN 978-1-4673-1611-8, ISSN 2160-7508, s. 1–8, doi:10.1109/CVPRW.2012.6238919.

[13] Haritaoglu, I.; Harwood, D.; Davis, L. S.: W^4: real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 22, č. 8, srpen 2000: s. 809–830, ISSN 0162-8828, doi:10.1109/34.868683.

[14] Heikkila, M.; Pietikainen, M.: A texture-based method for modeling the background and detecting moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 28, č. 4, duben 2006: s. 657–662, ISSN 0162-8828, doi:10.1109/TPAMI.2006.68.

[15] Hofmann, M.; Tiefenbacher, P.; Rigoll, G.: Background segmentation with feedback: The Pixel-Based Adaptive Segmenter. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, červen 2012, ISSN 2160-7508, s. 38–43, doi:10.1109/CVPRW.2012.6238925.

[16] Huang, S. S.; Fu, L. C.; Hsiao, P. Y.: Region-Level Motion-Based Foreground Segmentation Under a Bayesian Network. *IEEE Transactions on Circuits and Systems for Video Technology*, ročník 19, č. 4, duben 2009: s. 522–532, ISSN 1051-8215, doi:10.1109/TCSVT.2009.2013507.

[17] Hägelen, M.; Kulke, R.; Jetten, R.; aj.: Perimeter surveillance using a combination of radar and optical sensors. In *2016 European Radar Conference (EuRAD)*, říjen 2016, ISBN 978-2-8748-7045-3, s. 318–321.

[18] Kim, K.; Chalidabhongse, T. H.; Harwood, D.; aj.: Background modeling and subtraction by codebook construction. In *2004 International Conference on Image Processing*, ročník 5, říjen 2004, ISSN 1522-4880, s. 3061–3064, doi:10.1109/ICIP.2004.1421759.

[19] Maddalena, L.; Petrosino, A.: A Self-Organizing Approach to Background Subtraction for Visual Surveillance Applications. *IEEE Transactions on Image Processing*, ročník 17, č. 7, červenec 2008: s. 1168–1177, ISSN 1057-7149, doi:10.1109/TIP.2008.924285.

[20] Meyer, J.; Sendobry, A.; Kohlbrecher, S.; aj.: Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo. In *Simulation, Modeling, and Programming for Autonomous Robots*, editace I. Noda; N. Ando; D. Brugali; J. J. Kuffner, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ISBN 978-3-642-34327-8, s. 400–411.

[21] Oliver, N. M.; Rosario, B.; Pentland, A. P.: A Bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 22, č. 8, srpen 2000: s. 831–843, ISSN 0162-8828, doi:10.1109/34.868684.

[22] Ramezani, R.; Angelov, P.; Zhou, X.: A fast approach to novelty detection in video streams using recursive density estimation. In *2008 4th International IEEE Conference Intelligent Systems*, ročník 2, září 2008, ISSN 1541-1672, s. 142–147, doi:10.1109/IS.2008.4670523.

[23] Sobral, A.: BGSLibrary: An OpenCV C++ Background Subtraction Library. In *IX Workshop de Visão Computacional (WVC'2013)*, Rio de Janeiro, Brazílie, červen 2013.
URL https://github.com/andrewssobral/bgslibrary

[24] Sobral, A.; Vacavant, A.: A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding*, ročník 122, květen 2014: s. 4–21, ISSN 1077-3142, doi:https://doi.org/10.1016/j.cviu.2013.12.005.
URL http://www.sciencedirect.com/science/article/pii/S1077314213002361

[25] St-Charles, P. L.; Bilodeau, G. A.: Improving background subtraction using Local Binary Similarity Patterns. In *IEEE Winter Conference on Applications of Computer Vision*, březen 2014, ISSN 1550-5790, s. 509–515, doi:10.1109/WACV.2014.6836059.

[26] St-Charles, P. L.; Bilodeau, G. A.; Bergevin, R.: SuBSENSE: A Universal Change Detection Method With Local Adaptive Sensitivity. *IEEE Transactions on Image Processing*, ročník 24, č. 1, leden 2015: s. 359–373, ISSN 1057-7149, doi:10.1109/TIP.2014.2378053.

[27] St-Charles, P. L.; Bilodeau, G. A.; Bergevin, R.: Universal Background Subtraction Using Word Consensus Models. *IEEE Transactions on Image Processing*, ročník 25, č. 10, říjen 2016: s. 4768–4781, ISSN 1057-7149, doi:10.1109/TIP.2016.2598691.

[28] Stauffer, C.; Grimson, W. E. L.: Adaptive background mixture models for real-time tracking. In *1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, ročník 2, 1999, ISSN 1063-6919, s. 246–252, doi:10.1109/CVPR.1999.784637.

[29] Szeliski, R.; Shum, H.-Y.: Creating Full View Panoramic Image Mosaics and Environment Maps. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, ISBN 0-89791-896-7, s. 251–258, doi:10.1145/258734.258861.

[30] Tanaka, T.; Shimada, A.; Arita, D.; aj.: Non-parametric Background and Shadow Modeling for Object Detection. In *Computer Vision – ACCV 2007*, editace Y. Yagi; S. B. Kang; I. S. Kweon; H. Zha, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, ISBN 978-3-540-76386-4, s. 159–168.

[31] Tavakkoli, A.; Nicolescu, M.; Bebis, G.: An adaptive recursive learning technique for robust foreground object detection. In *International Workshop on Statistical Methods*

in Multi-image and Video Processing (in conjunction with ECCV06), květen 2006, ISBN 978-3-540-33833-8, doi:10.1007/11744023.

[32] Tavakkoli, A.; Nicolescu, M.; Bebis, G.: Automatic Statistical Object Detection for Visual Surveillance. In *2006 IEEE Southwest Symposium on Image Analysis and Interpretation*, březen 2006, ISBN 1-4244-0069-4, s. 144–148, doi:10.1109/SSIAI.2006.1633739.

[33] Tavakkoli, A.; Nicolescu, M.; Bebis, G.: Robust Recursive Learning for Foreground Region Detection in Videos with Quasi-Stationary Backgrounds. In *18th International Conference on Pattern Recognition (ICPR'06)*, ročník 1, srpen 2006, ISSN 1051-4651, s. 315–318, doi:10.1109/ICPR.2006.1015.

[34] Toyama, K.; Krumm, J.; Brumitt, B.; aj.: Wallflower: principles and practice of background maintenance. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, ročník 1, září 1999, ISBN 0-7695-0164-8, s. 255–261, doi:10.1109/ICCV.1999.791228.

[35] Tsai, T. H.; Lin, C. Y.; Li, S. Y.: Algorithm and Architecture Design of Human-Machine Interaction in Foreground Object Detection With Dynamic Scene. *IEEE Transactions on Circuits and Systems for Video Technology*, ročník 23, č. 1, leden 2013: s. 15–29, ISSN 1051-8215, doi:10.1109/TCSVT.2012.2202193.

[36] Wang, H.; Suter, D.: Background Subtraction Based on a Robust Consensus Method. In *18th International Conference on Pattern Recognition (ICPR'06)*, ročník 1, srpen 2006, ISSN 1051-4651, s. 223–226, doi:10.1109/ICPR.2006.312.

[37] Wang, H.; Suter, D.: A consensus-based method for tracking: Modelling background scenario and foreground appearance. *Pattern Recognition*, ročník 40, č. 3, 2007: s. 1091–1105, ISSN 0031-3203, doi:https://doi.org/10.1016/j.patcog.2006.05.024.

[38] Wolff, C.: *Frequency-Modulated Continuous-Wave Radar (FMCW Radar)*. [Online; navštíveno 20.01.2018].
URL http://www.radartutorial.eu/02.basics/Frequency%20Modulated%20Continuous%20Wave%20Radar.en.html

[39] Wren, C. R.; Azarbayejani, A.; Darrell, T.; aj.: Pfinder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 19, č. 7, červenec 1997: s. 780–785, ISSN 0162-8828, doi:10.1109/34.598236.

[40] Xu, Y.; Dong, J.; Zhang, B.; aj.: Background modeling methods in video analysis: A review and comparative evaluation. *CAAI Transactions on Intelligence Technology*, ročník 1, č. 1, 2016: s. 43–60, ISSN 2468-2322, doi:10.1016/j.trit.2016.03.005.
URL http://www.sciencedirect.com/science/article/pii/S2468232216000068

[41] Yao, G.; Lei, T.; Zhong, J.; aj.: Comparative Evaluation of Background Subtraction Algorithms in Remote Scene Videos Captured by MWIR Sensors. *Sensors*, ročník 17, č. 9, 2017, ISSN 1424-8220, doi:10.3390/s17091945.
URL http://www.mdpi.com/1424-8220/17/9/1945

[42] Zivkovic, Z.: Improved adaptive Gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004*, ročník 2, srpen 2004, ISSN 1051-4651, s. 28–31, doi:10.1109/ICPR.2004.1333992.

Příloha A

Rejstřík použitých symbolů

Symboly jsou seřazeny dle výskytu v textu.

TP	Počet pravých pozitiv
FP	Počet falešných pozitiv
FN	Počet falešných negativ
Pr	Precision
Re	Recall
w_{Fr}	Šířka snímku kamery v pixelech
h_{Fr}	Výška snímku kamery v pixelech
R_{Fr}	Rozlišení snímku kamery v pixelech
M_{Px}	Paměťová náročnost jednoho pixelu modelu pozadí
M_{Fr}	Paměťová náročnost modelu pozadí pro jeden snímek
P	Procentuální část panoramatu, která by se vešla do RAM
M_{Pan}	Velikost panoramatu v paměti
M_{RAM}	Velikost paměti RAM
α_H	Úhel [°] v horizontálním směru, který by zabrala část panoramatu, která by se vešla do paměti a přitom by byl zachován stejný poměr stran, jaký má obraz kamery
α_V	Úhel [°] ve vertikálním směru, který by zabrala část panoramatu, která by se vešla do paměti a přitom by byl zachován stejný poměr stran, jaký má obraz kamery
S	Rychlost SSD při současném čtení i zápisu dat v [B/s]
ω_H	Maximální úhlová rychlost, kterou by se manipulátor mohl pohybovat v horizontálním směru tak, aby aplikace stíhala zapisovat a číst části panoramatu z SSD
ω_V	Maximální úhlová rychlost, kterou by se manipulátor mohl pohybovat ve vertikálním směru tak, aby aplikace stíhala zapisovat a číst části panoramatu z SSD
ω_B	Maximální úhlová rychlost, kterou by se manipulátor mohl pohybovat šikmo v horizontálním i ve vertikálním směru tak, aby aplikace stíhala zapisovat a číst části panoramatu z SSD
H_{Fr}	Horizontální zorný úhel kamery
V_{Fr}	Vertikální zorný úhel kamery

F_x, F_y	Souřadnice x a y v rámci snímku kamery
θ, ϕ	Sférické souřadnice
P^S	3D souřadnice bodu na promítací ploše kamery
K	Vnitřní matice kamery
P^H	Homogenní souřadnice bodu na promítací ploše kamery
a	Azimut pan tilt jednotky (náklon v horizontálním směru)
e	Elevace pan tilt jednotky (náklon ve vertikálním směru)
z_{corr}	Korekce rotace kamery kolem osy z (normála k promítací ploše)
R	Rotační matice kamery
w_{Pan}	Počet pixelů v panoramatu v řádku kolem rovníku
h_{Pan}	Počet řádků panoramatu
H_{Pan}	Horizontální úhel záběru panoramatu
V_{Fr}	Vertikální úhel záběru panoramatu
l_{Proj}	Délka promítací plochy
l_{Px}	Délka pixelu na promítací ploše
α_E	Úhel zabraný krajním pixelem na promítací ploše
α_M	Úhel zabraný prostředním pixelem promítací plochy
w_E, w_M	Váhy při výpočtu váženého průměru α_E a α_M
p_x, p_y	Souřadnice v prostoru panoramatu
p_x^R, p_y^R	Relativní souřadnice v prostoru panoramatu
w_R	Délka řádku v pixelech
t_x, t_y	Indexy dlaždice
o_x	Offset (index) v rámci dlaždice
w_T	Šířka dlaždice v pixelech
φ	Úhel v azimutu nebo elevaci
ω	Úhlová rychlost v azimutu nebo elevaci
ε	Úhlové zrychlení v azimutu nebo elevaci
t	čas

Příloha B

Obsah DVD

Umístění	Obsah
`/README.txt`	Popis obsahu DVD, návod na překlad a spuštění
`/Opticky_radar_s_vyuzitim_dvouoseho_kameroveho_manipulatoru.pdf`	Text této práce
`/plakat.svg`	Plakát
`/tex`	Zdrojové soubory textové části ve formátu LaTeX
`/catkin_ws/src`	Zdrojové kódy ROS uzlů