VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

LOKALIZACE MOBILNÍHO ROBOTA POMOCÍ KAMERY

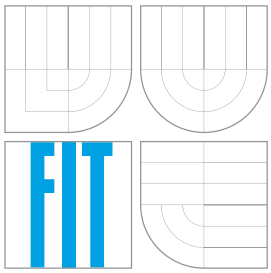
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. FILIP VAVERKA

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

LOKALIZACE MOBILNÍHO ROBOTA POMOCÍ KAMERY

MOBILE ROBOT LOCALIZATION USING CAMERA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. FILIP VAVERKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAROSLAV ROZMAN, Ph.D.

BRNO 2015

Abstrakt

Tato práce popisuje návrh a realizaci metody lokalizace mobilního robota. Metoda je založena čistě na obrazových datech získaných pomocí monokulární kamery. Lokalizace je v popisovaném řešení chápána jako asociační problém a jde tedy o lokalizaci v topologickém modelu prostření, který je předem vytvořen. Základem metody je generativní pravděpodobnostní model vzhledu prostředí. Tento způsob lokalizace umožňuje eliminovat některé obtížné problémy, kterými trpí klasické lokalizační metody.

Abstract

This thesis describes design and implementation of an approach to the mobile robot localization. The proposed method is based purely on images taken by a monocular camera. The described solution handles localization as an association problem and, therefore, falls in the category of topological localization methods. The method is based on a generative probabilistic model of the environment appearance. The proposed solution is capable to eliminate some of the difficulties which are common in traditional localization approaches.

Klíčová slova

vizuální lokalizace, monokulární kamera, topologický prostor, pravděpodobnostní model, generativní model, náhodné grafy

Keywords

appearance-based localization, monocular camera, topological space, probabilistic model, generative model, random graphs

Citace

Filip Vaverka: Lokalizace mobilního robota pomocí kamery, diplomová práce, Brno, FIT VUT v Brně, 2015

Lokalizace mobilního robota pomocí kamery

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Rozmana, Ph.D.

.....
Filip Vaverka
27. května 2015

Poděkování

Děkuji Ing. Davidovi Hermanovi za odbornou pomoc při realizaci této práce.

Obsah

Kapitola 1

Úvod

Cílem této práce je navrhnout a realizovat systém lokalizace mobilního zařízení (mobilního robota) vybaveného monokulární kamerou v rámci předem zmapovaného prostředí. Popisovaný systém je vhodný pro aplikace kde robot nemusí v prostředí plánovat nové cesty. Pro tento typ úloh tedy není nutné vytvářet, nebo se orientovat v rámci kompletní metrické mapy prostředí. Místo mapy systém využívá pouze vizuální informace bez nutnosti odvozování globálních prostorových vztahů mezi význačnými body. Model prostředí pak může být pro tento typ úloh tvořen množinou diskrétních míst (lokací), které lze popsat vizuálními daty (tedy snímky). Tato množina může být navíc částečně nebo plně uspořádaná.

Úlohou lokalizačního systému je nalézt v tomto prostoru popis místa, který nejlépe odpovídá aktuálnímu pozorování, čímž dochází k určení aktuální pozice (a orientace) robota ve fyzickém prostoru. Tento přístup k lokalizaci je inspirován způsobem, jakým probíhá v biologických systémech (jako je mozek), kde se tento přístup dá přirovnat k lokálním mapám míst (tzv. "sketch maps" dle [14]).

V následujícím textu pak budou nejdříve popsány klasické přístupy k metrické lokalizaci a následně srovnány s vizuální lokalizací (v topologickém prostoru). Následuje popis použitého modelu prostředí, pozorování a na nich založené metody lokalizace v topologickém prostoru. Dále je popsána implementace této metody a její testování. V závěru práce jsou pak zhodnoceny dosažené výsledky a navrženy možné směry dalšího rozvoje zvoleného přístupu.

Kapitola 2

Metody mapování a lokalizace

V tomto oddíle bude popsáno rozdělení a několik základních metod lokalizace mobilních robotů. Za základní rozdělení přístupů k lokalizaci bývá (dle [13]) považováno rozdělení na metody relativní (někdy též označované jako lokální) a absolutní (globální). Klíčovými rozdíly mezi těmito přístupy je referenční rámec v jakém pracují a požadavky, které mají na okolní prostředí a senzorickou výbavu. Jako zástupce rodiny metod relativní lokalizace je možné uvést například techniky SLAM (Simultaneous Localization and Mapping), kde není cílem pouze samotná lokalizace, ale také aktualizace a tvorba mapy prostředí, ve které se robot pohybuje. Blíže se porovnání těchto metod bude věnovat následující text.

Tyto metody mohou navíc využívat kombinaci uvedených přístupů, tak aby byla chyba určení polohy statisticky minimalizována (ať už ve vztahu k absolutní přesnosti či driftu s postupujícím časem). Mnoho z těchto technik je založeno na Kalmanově filtru [24]. Jinou metodou, kterou lze zlepšit výsledky lokalizace jsou částicové filtry [21] (někdy též "Sequential Monte Carlo" nebo "Monte Carlo Localization"). V případě částicového filtru je stav systému (přesněji funkce rozložení pravděpodobnosti tohoto stavu) reprezentován populací částic (vzorků). Každá z těchto částic představujících jeden z možných stavů systému má přiřazenu váhu, která udává její kvalitu. Po každé akci robota jsou všechny částice (jejich stavy) aktualizovány dle předem známého modelu. V čase, kdy jsou k dispozici data senzorů robota, dojde k omezení možných stavů a aktualizaci vah částic (částice s příliš nízkou váhou jsou pak odstraněny a nahrazeny). Celkový stav systému (odhad reálného stavu) je pak možné získat váženým součtem stavů všech částic. Nevýhoda tohoto systému je zřejmá: její přesnost závisí na počtu provedených měření.

2.1 Lokalizace v metrickém prostoru

Úkolem lokalizace v metrickém prostoru (kterým může být například prostor $\mathbb{R}^3$) je umožnit určení pozice robota v kterémkoli bodě jeho pohybu. Tato pozice je vztažena k nějakému rámci, který může být globální nebo lokální (počáteční pozice v případě relativní lokalizace).

2.1.1 Relativní lokalizace

Techniky relativní lokalizace jsou založeny na inkrementální aktualizaci polohy a orientace robota v prostoru na základě jejich změn, které jsou získávány pomocí různých senzorů. Mezi rozšířené senzory patří enkodéry (např. senzor rotace kola), gyroskopy, akcelerometry a další. Mezi používané metody tohoto typu patří odometrie, která může být založena

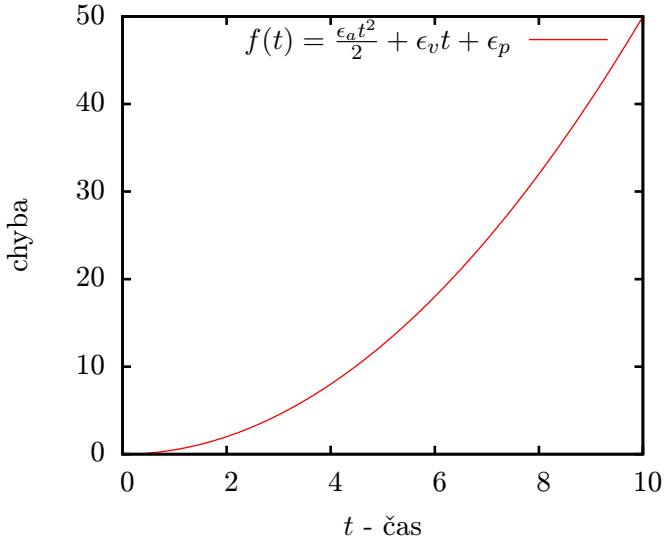
na výše zmíněných senzorech, nebo počítačovém vidění (pak hovoříme o vizuální odometrii 2.3.1). Jinou metodou v této kategorii je tzv. “dead reckoning” (inerciální navigační systémy).

Problémem těchto metod je postupná integrace chyb senzorů, což způsobuje neohraničený růst chyby v průběhu času. Uvažujme například velmi zjednodušený případ odometrie založené na senzoru rotace kola. V tomto případě lze novou pozici určit integrací uražené vzdálenosti dané rotací kola a jeho obvodem (rovnice 2.1).

$$x_{t+\Delta t} = x_t + \Delta t w \phi + \epsilon \tag{2.1}$$

Kde $x_{t+\Delta t}$ je nová pozice (uvažujeme jednorozměrný případ), x_t je pozice v čase t, Δt je perioda vzorkování, w je obvod kola, ϕ je úhlová rychlost jeho otáčení a $\epsilon \sim \mathcal{N}(0, N)$ je aditivní bílý šum (tedy chyba).

V případě inerciálních systémů dochází při výpočtu nové pozice k dvojité integraci vstupních dat ze senzorů, například při určování polohy z údajů o akceleraci. Velikost akcelerace totiž změříme s jistou chybou ϵ, tedy $f'' = \epsilon_a$. První integrací pak dostaneme chybu rychlosti rostoucí v čase dle $f' = \epsilon_a t + \epsilon_v$. A konečně finální integrací dostaneme chybu polohy rostoucí kvadraticky v čase $f = \frac{1}{2}\epsilon_a t^2 + \epsilon_v t + \epsilon_p$ (viz. obrázek 2.1). Výhodou těchto systémů je nezávislost na vstupu dat z okolního prostředí (nedochází k jeho pozorování, ani spoléhání se na data z externích zdrojů).

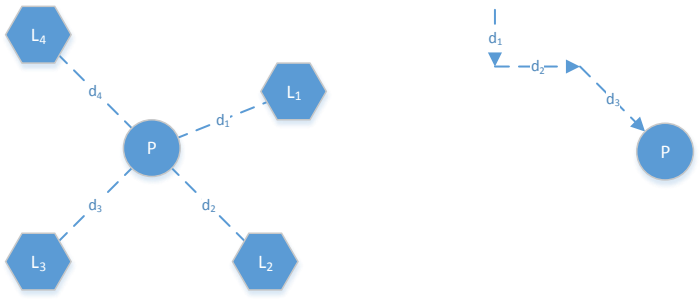


Obrázek 2.1: Průběhu chyby při lokalizaci na základě akcelerace

2.1.2 Absolutní lokalizace

Základem této rodiny metod je nějaký globální referenční rámec, vůči kterému určujeme pozici robota. Za typického zástupce této rodiny metod lze považovat *Globální polohovací systém* (GPS), který pro určení absolutní pozice na povrchu Země v podobě šířky, délky a nadmořské výšky využívá satelitních signálů. Jiným příkladem tohoto způsobu lokalizace mohou být metody založené na význačných bodech (tzv. “landmarks”), nebo majících (tzv. “beacons”). Zde navržená metoda (blíže popsaná v kapitole 3) má určité znaky společné s metodami založenými na význačných bodech.

Nevýhodou metod založených na majácích je nutnost jejich rozmístění (což může být nákladné), navíc lokalizace může probíhat pouze v dané oblasti. V případě GPS je nevýhodou nutnost dostupnosti signálu satelitů (což může být problém v uzavřených prostorech, jako jsou budovy). Druhým problémem je přesnost této metody, která je jednak proměnlivá (v závislosti na počtu dostupných satelitů), jednak pro některé účely nedostatečná. Přesnost systému GPS lze zlepšit použitím technik jako je *diferenciální GPS*, což však zvyšuje cenu řešení a přidává další omezení.



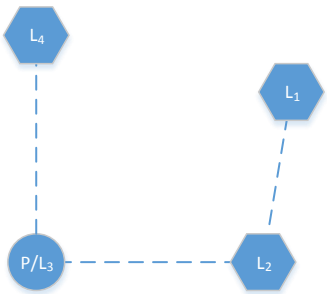
(a) Globální lokalizace (b) Relativní lokalizace

Obrázek 2.2: Tradiční metody lokalizace

2.2 Lokalizace v topologickém prostoru

Na rozdíl od lokalizace v metrickém prostoru je cílem metod lokalizace v topologickém prostoru nalezení nejbližšího bodu (nejpodobnějšího snímku) z množiny bodů (snímků, nebo jejich popisů), která může být (ale není to podmínkou) vytvořena před samotným použitím. Účelem těchto metod tedy není umožnit plánování libovolného pohybu na základě mapy prostoru a souvislé lokalizace v ní, ale pouze umožnit navigaci vždy mezi dvěma klíčovými body.

Množina klíčových bodů pak musí být uspořádána v nejjednodušším případě lineárně v čase nebo v grafu, kde vrcholy představují klíčové body a hrany vztahy mezi nimi. Tyto body pak mohou mít tvar n-tic $n = (x, t)$ respektive $n = (x, i)$ a $e = (n_1, n_2, t)$, kde x je popis význačného bodu (místa), e je hrana mezi vrcholy n_1 a n_2, t je čas mezi aktuálním a následujícím bodem a i je přídavná informace využitelná pro navigaci v grafu. Poznamenejme, že toto uspořádání není vynucováno samotnou lokalizační metodou (i když i zde je výhodou), ale spíše požadavky navigačního algoritmu.



Obrázek 2.3: Lokalizace v topologickém prostoru

2.3 Vizuální data v lokalizaci

Vizuální data jsou krom metod lokalizace v topologickém prostoru využívána i při lokalizaci metrické. Zde se s úspěchem využívá jak dat monokulární kamery [10] tak dat z hloubkových kamer [22].

Na poli lokalizace v topologickém prostoru byly zpočátku využívány metody založené na celkovém popisu vzhledu snímku, kde se využívalo redukce dimenzionality. Mezi tyto metody se řadí práce [23], kde je ke snížení dimenze využíváno barevných histogramů, nebo [15], kde je dimenze snímku redukována pomocí *Analýzy hlavních komponent* (PCA). Nevýhodou těchto metod je právě zachycení informace z celého snímku, čímž dochází ke snížení robustnosti při nepodstatných změnách ve scéně, jako jsou změny osvětlení, změna perspektivy a pohyb dynamických objektů.

Jedním z možných řešení těchto problémů je omezení zachycené informace ze snímku a to tak, aby zůstala zachována především struktura scény. K tomuto úkolu lze přistupovat například využitím detekce klíčových bodů a jejich deskriptorů (tzv. "local features"). Jako příklad takového detektoru a deskriptoru lze uvést například algoritmus *SIFT* [16]. Snímek je pak popsán množinou lokálních příznaků a snímky jsou porovnávány na základě těchto množin příznaků. Takto vytvořené lokální příznaky pak zajišťují (do určité míry) řešení problémů způsobených změnou perspektivy nebo osvětlení a umožňují korektní detekci scény i v případě jejího částečného překrytí. Přetrvává však problém s výběrem příznaků, které nejlépe charakterizují daný snímek a zároveň ho odlišují od ostatních. S tímto problémem souvisí problematika opakujících se textur (případně struktur) a dochází tak k chybám, kdy se dvě různá místa mohou jevit jako místo jedno (tzv. "Perceptual aliasing").

Problematiku opakujících se textur, objektů a na vyšší úrovni celých architektonických prvků řešila práce [18] v rámci algoritmu pro vizuální detekci uzavírání smyček systému *SLAM*. Tato práce využívala podobně jako zde prezentované řešení popis snímků pomocí lokálních příznaků spolu s vizuálním slovníkem. Následně byla vypočtena matice podobnosti snímků založená na kosinové vzdálenosti slov popisujících dané snímky. V jednoduchém prostředí se pak možné smyčky jeví jako proužky s vysokou hodnotou podobnosti mimo hlavní diagonálu matice. V repetitivním prostředí pak byla využita dekompozice (založená na *SVD*) matice podobnosti a extrakce sekvencí byla prováděna až následně.

Vrcholem v této oblasti je pak algoritmus *FABMAP* [7], ten je obdobně jako před-

chozí přístup založen na lokálních příznacích a vizuálním slovníku, avšak snaží se pomocí pravděpodobnostního modelu prostředí zachytit závislosti mezi výskyty jednotlivých slov ve snímcích. Tento model umožňuje krom zlepšení celkových výsledků a robustnosti metody také detekovat pozorování dosud neznámého prostředí. Tento algoritmus byl proto zvolen jako jedna z komponent dále popisovaného řešení a je proto dále podrobně rozebrán a popsán v oddíle 3.3.

2.3.1 Vizuální odometrie

Vizuální odometrie umožňuje nahradit ostatní senzory využívané v odometrii vizuálním snímačem (ideálně monokulární kamerou). To umožňuje snížení nákladů a zároveň nabízí řešení problémů jako je např. podklouznutí kol v případě odometrie založené na enkodérech. S výhodou lze vizuální odometrii kombinovat s detekcí hazardů a překážek (např. práce [5]), případně objektů pohybujících se v cestě robota. Cenou za tyto výhody je pak zřejmě zvýšená výpočetní náročnost (ve srovnání se zmíněným sledováním rotace kol). Princip této techniky pak lze popsat několika kroky:

1. zachycení snímku monokulární kamerou (případně stereo kamerou)
2. korekce snímku (jako je odstranění distorze kamery)
3. detekce klíčových bodů, jejich párování a sestavení optického toku
4. filtrace vektorů optického toku a vyřazení chybných (případně náležejících samostatně se pohybujícím objektům)
5. odhad pohybu kamery na základě filtrovaného optického toku (často za pomoci Kalmanova filtru)

2.3.2 Výhody vizuálních metod

Jedním ze základních problémů metrických metod lokalizace, který se snaží vizuální metody odstranit, je datová asociace mezi pozorováním a samotnou lokací (objektem) v mapě robota. Tento problém si lze přiblížit na příkladě lokalizace založené pouze na měření vzdálenosti k objektům v okolí. K lokalizaci na základě těchto měření je však nutné vědět, ke kterému objektu v mapě se toto měření vztahuje. Metrické lokalizační systémy tento problém řeší pomocí heuristik, případně sledováním několika různých hypotéz (částicové filtry).

V praxi se pak takovýto problém projevuje při uzavírání smyček, kdy se robot po průchodu neznámým terénem vrací zpět do již zmapované oblasti. V takovéto situaci často nastává stav, kdy během průchodu neznámého terénu dojde k integraci drobných chyb a znovu navštívené místo se ve tvořené mapě jeví jako místo nové. Obdobný stav nastane, pokud není známa výchozí pozice robota, nebo dojde k výpadku snímání některého z jeho senzorů. Vizuální metody k řešení tohoto druhu problémů využívají vzhledu prostředí, který nese více informací použitelných k identifikaci lokace, než například zmíněné měření vzdálenosti.

Některé další vlastnosti vizuálních metod

- nevyžaduje drahou a komplexní senzorickou výbavu
- navigační úlohu typu "map-and-replay" lze řešit pouze jedním senzorem; odpadá tedy problém časové synchronizace dat z více senzorů a jejich vzájemná kalibrace
- schopnost lokalizace v rozsáhlých vnitřních i vnějších prostorech za pomoci přírodních markantů (nevyžaduje externí zásah do prostředí - GPS, radiolokační majáky, vizuální markanty)
- absolutní lokalizace v rámci interního modelu; nedochází ke kumulaci chyby v čase
- možnost použití jako podpůrný podsystém u metrických systémů, zejména pro zmíněnou detekci uzavírání smyček, kde konvenční přístupy mohou selhávat
- metoda není závislá na konkrétních vlastnostech použitého senzoru, potenciální možnost sdílení vizuální mapy mezi více roboty
- inspirace biologickými systémy, které se rovněž umí orientovat v prostoru bez exaktní informace o své poloze a poloze okolních objektů

Tyto vlastnosti umožňují použití těchto metod v netradičních aplikacích, jako například lokalizace uživatele za pomoci kamery jeho mobilního telefonu v budovách, kde může být problém s dostupností GPS signálu. Výhodou je také možnost lokalizace na základě dat, která k tomuto účelu původně nebyla určena (např. pro nalezení cesty a navigaci po ní může postačovat libovolná videosekvence, která prochází aktuální polohou i hledaným cílem).

Kapitola 3

Navržená metoda lokalizace

V této kapitole bude popsána samotná metoda vizuální lokalizace v topologickém prostoru, který je zde chápán jako prostá množina míst bez uspořádání. K tomuto účelu je tedy nejdříve definován model prostředí, jednotlivých lokací v něm a jejich pozorování. Tyto modely využívají detekce klíčových bodů a vizuálního slovníku pro reprezentaci jejich deskriptorů (tzv. "Bag-of-Words") a tyto techniky jsou tedy také popsány. Hlavní část kapitoly pak tvoří popis pravděpodobnostních modelů výskytu a prostorové konfigurace lokálních příznaků ve snímcích a scénách.

Cílem navržené lokalizační metody je tedy lokalizace v topologickém prostoru na základě obrazových dat, bez závislosti na dalších senzorech a v předem zmapovaném prostředí. Na základě těchto požadavků a předchozího vývoje popsaného v oddíle 2.3, byl jako jádro řešení zvolen algoritmus *FABMAP* [7]. Shodně s přístupem *FABMAP* zůstala zachována reprezentace snímků pomocí lokálních příznaků a slovníku. Tento popis však byl rozšířen o prostorovou strukturu lokálních příznaků ve snímku, která je zachycena pomocí vzdáleností mezi dvojicemi příznaků. Důsledkem je rozlišitelnost dvojice snímků, které obsahují shodná vizuální slova, ale v jiné prostorové konfiguraci. Výhodou této reprezentace struktury (v porovnání s uvažováním pozice příznaků) je její menší závislost na orientaci kamery (rotace kolem pohledové osy) a také vůči posunům kolmým k pohledové ose. V případě použití polohy příznaků by bylo nutné najít vzájemné (homomorfní) zobrazení mezi snímky (např. použitím algoritmu *RANSAC*) a následně porovnat pozice příznaků.

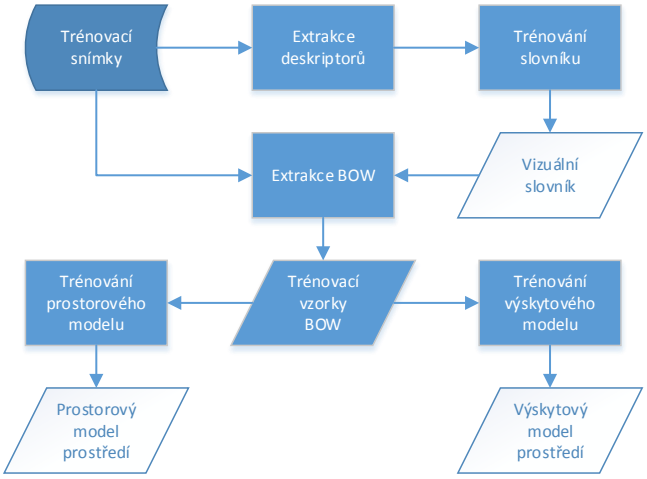
3.1 Činnost lokalizační metody

Navržená metoda je založena na generativních modelech výskytů a prostorové konfigurace klíčových bodů (které jsou reprezentovány vizuálním slovníkem) ve snímcích. Je tedy nutné před jejím nasazením v nějakém prostředí tyto modely na dané prostředí natrénovat (obrázek 3.1). První fází procesu trénování je extrakce klíčových bodů a deskriptorů z trénovací sady (tento proces popisuje oddíl 3.2.1) následně je třeba na základě takto získaných dat vytvořit vizuální slovník (jeho princip a tvorbu popisuje oddíl 3.2.2).

Pomocí vytvořeného slovníku jsou dále extrahovaná trénovací data převedena do podoby vhodné pro trénování pravděpodobnostních modelů metody. V průběhu tohoto procesu dochází k omezení množiny rozeznávaných lokálních deskriptorů a zároveň zobecnění jejich popisu, na jeho konci jsou pak trénovací data tvořena množinou pozic klíčových bodů přiřazených ke slovům ve slovníku (pomocí původních lokálních deskriptorů).

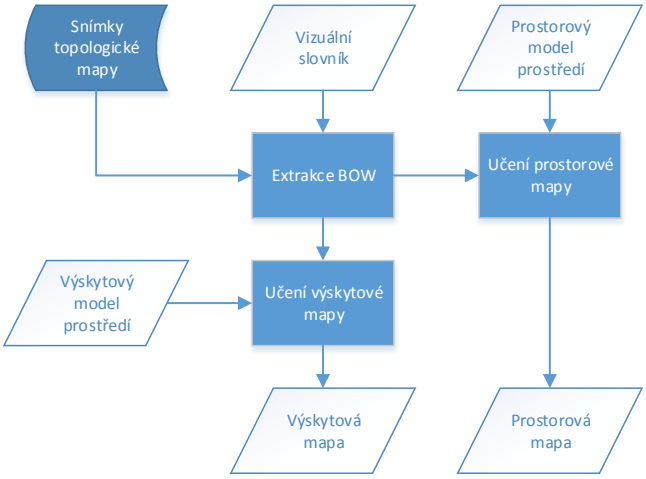
Na základě těchto dat je již možné trénovat zmíněné modely prostředí, cílem modelů je

zachytit apriorní vlastnosti prostředí, tedy jak často lze v prostředí narazit na scénu (scénou rozumíme pohled jistým směrem z jisté pozice), ve které se vyskytuje dané slovo vizuálního slovníku (pro výskytový model - oddíl 3.3) a jaká je pravděpodobnost, že mezi dvojicí slov bude ve scéně vzdálenost d (prostorový model popsaný v sekci 3.4). Tyto modely však zachycují také závislosti mezi výskyty a vzdálenostmi vizuálních slov. Díky tomu umožňují vytvořit scény, které jsou v daném prostředí typické a na jejich základě odhalit význačné rysy scén náležejících do topologické mapy.



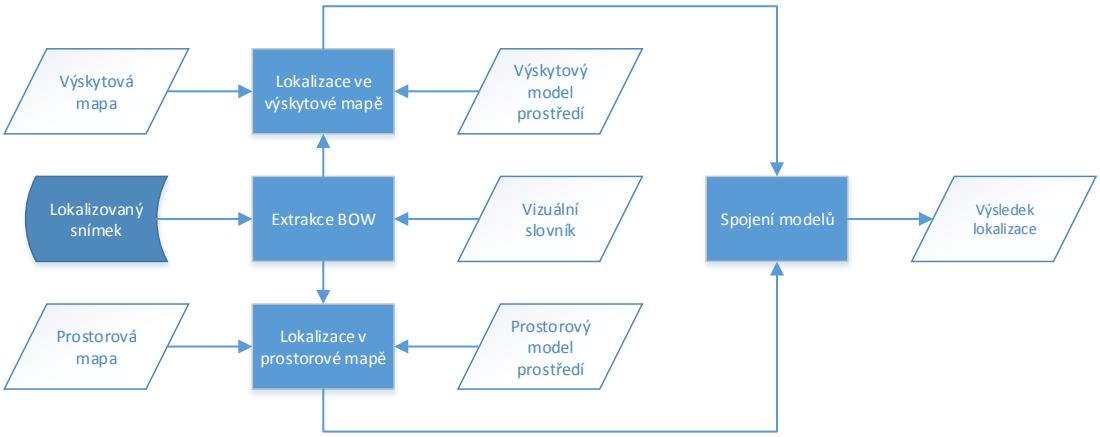
Obrázek 3.1: Proces trénování modelu

Druhou fází (obrázek 3.2) je vytvoření topologické mapy založené na snímcích scén (podobně jako v případě trénovacích dat), které jsou však asociovány s určitou fyzickou lokací (a pohledem) v prostoru. Opět jsou tedy zaznamenány snímky, identifikovány klíčové body a přiřazeny ke slovům ve slovníku. Na základě těchto dat a apriorního modelu prostředí jsou následně vytvořeny modely těchto scén.



Obrázek 3.2: Proces učení topologické mapy

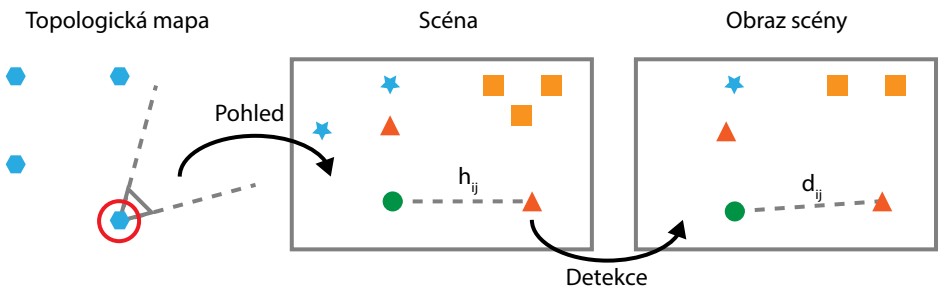
Poslední fází je pak samotné použití algoritmu, kde jsme prostřednictvím modelu senzoru schopni porovnat pozorování aktuální scény s modely scén vytvořených v předchozí fázi a nalézt mezi nimi nejlépe odpovídající (a tím dochází k lokalizaci). Tuto fázi procesu lokalizace zachycuje obrázek 3.3.



Obrázek 3.3: Proces lokalizace v topologické mapě

3.2 Reprezentace prostředí a pozorování

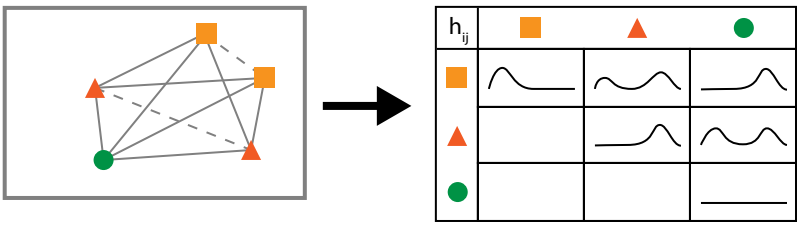
Modelem prostředí je tedy množina disjuktních míst (lokací) reprezentovaných snímkem scény pořízeným z dané lokace. Místem rozumíme bod v prostoru, kde se robot nachází, scénou pak pohled z tohoto místa v určitém směru. Snímek tedy představuje dvourozměrnou projekci scény. Toto rozdělení zavádíme pro potřeby pravděpodobnostních modelů, kde je třeba rozlišit, které klíčové body a deskriptory se ve scéně fyzicky vyskytují a které jsou zachyceny ve snímku a detekovány (projevy nedokonalostí detektoru a zachycení scény). Tyto pojmy zachycuje obrázek 3.4. Klíčový bod je pak opakovatelně detekovatelný význačný prvek ve snímku, který je obvykle spojen s deskriptorem tvořícím jeho popis. Důležitou vlastností klíčových bodů a jejich deskriptorů je invariance vůči transformacím obrazu, jako jsou rotace, změna měřítka či zkosení. Díky tomu je možné je detekovat i při mírné změně polohy (místa) ze které je scéna pozorována a hlavně úhlu natočení kamery. Způsobem detekce klíčových bodů a jejich popisem se dále zabývá oddíl 3.2.1.



Obrázek 3.4: Vztah mezi lokací, scénou a snímkem.

Aby bylo možné vytvořit model nad výskyty konkrétních lokálních příznaků (tedy klíčového bodu s určitým deskriptorem) je nutné omezit počet možných deskriptorů a také nad nimi zavést operátor rovnosti. V obvyklém případě (mimo binární deskriptory) jsou deskriptory tvořeny vektorem reálných čísel se značným počtem dimenzí ($d \in \mathbb{R}^{64}$ pro *SIFT* deskriptor). Je proto výhodné zavést vizuální slovník, tedy strukturu, která tento prostor diskretizuje za pomoci shlukování. Takto vznikne konečná množina středů shluků o velikosti $|V|$ a dva deskriptory d_1, d_2 jsou považovány za shodné, pokud mají minimální vzdálenost ke stejnému středu $v \in V$. Výhodou použití této struktury je možnost reprezentace deskriptorů nalezených ve snímku pomocí binárního vektoru w délky $|V|$, pro který platí, že $w_i = 1$ právě když se ve snímku vyskytuje alespoň jeden deskriptor $d : d = v_i \in V$. Dále se strukturou a tvorbou vizuálního slovníku zabývá oddíl 3.2.2.

Na základě pojmů definovaných v předchozích odstavcích lze tedy prostředí popsat množinou disjunktních míst $\mathcal{L}^k = \{L_1, \ldots, L_{n_k}\}$, tvořící model zmapovaného prostředí, každé toto místo je popsáno scénou, kterou tvoří náhodný graf $L_i = (E_i, H_i)$, kde $E_i = \{p(e_1 = 1|L_i), \ldots, p(e_{|V|} = 1|L_i)\}$ je množina zachycující pravděpodobnost, že se slovo $e_i \in V$ vyskytuje v dané scéně. $H_i = \{h_{ij}(x)|1 \leq i, j \leq |V|\}$ je množina funkcí hustoty pravděpodobnosti nad vzdálenostmi mezi dvojicemi klíčových bodů ve 2D prostoru.



Obrázek 3.5: Vizualizace množiny H_i (vpravo) vytvořené ze scény (vlevo).

Zachycením scény v podobě snímku a následnou detekcí lokálních příznaků vzniká její pozorování, které je popsáno, v souladu s modelem scény, jako dvojice (Z_k, D_k), kde $Z_k = \{z_1, \ldots, z_{|V|}\}$ je množina binárních proměnných, pro které platí, že $z_i = 1$ pokud bylo detekováno (alespoň jedenkrát) i-té slovo vizuálního slovníku. $D_k = \{d_{ij}^n|1 \leq i, j \leq |V|, 1 \leq n \leq c_{ij}\}$ je množina vzdáleností mezi dvojicemi slov. Hodnota c_{ij} pak je počet možných dvojic vzdáleností mezi všemi lokálními příznaky, které náleží slovům i nebo j ve slovníku. Tedy $c_{ij} = 1$ pokud se slova i a j každé vyskytují pouze jednou, obecně $c_{ij} = c_i c_j$, kde c_i, c_j je počet výskytů i-tého (resp. j-tého) slova.

3.2.1 Detekce klíčových bodů

Detekce klíčových bodů a jejich popisů (deskriptorů) je jedním z ústředních a stále se živě vyvíjejících témat v oblasti počítačového vidění. Jeho cílem je nalézt ve snímcích snadno a především opakovaně detekovatelné, tedy klíčové body. Metrikou kvality detektorů je jejich schopnost opakovaně detekovat shodné význačné body navzdory transformacím snímku, změnám světelných podmínek a podobně.

Často používaným přístupem je využití Hessovy matice (*Hessian matrix*) nad konvolucí druhé derivace Gaussovy funkce a snímku, její determinant (hessián) pak slouží jako ukazatel významnosti daného bodu. Pro zajištění invariance k změně měřítka pak lze využít

detekci nad obrazovou pyramidou, kde se jeden detektor aplikuje postupně na zmenšeniny původního obrazu (v každém kroku se například obraz zmenší na polovinu), jinou možností je změna rozměru samotného filtru, zde se s výhodou využívá integrálního obrazu (*Summed Area Tables*), který umožňuje výpočet integrálu nad libovolnou čtvercovou oblastí obrazu v konstantním čase. Přístup hessiánu a integrálního obrazu využívá například detektor (a deskriptor) použitý v této práci - *SURF* [4] (zde je však použit pouze v roli deskriptoru). V roli detektoru ho však zastupuje *STAR* [1], jež má lepší vlastnosti než výše popsaný *SURF* detektor (jak v kvalitě detekce, tak výpočetní náročnosti).

Deskriptory lze rozdělit do dvou hlavních skupin, binární a reálné. Toto rozdělení je založeno na způsobu, jakým je okolí daného bodu popsáno, výstupem binárního detektoru je binární vektor určité délky (např. $d_{\text{bin}} \in \{0,1\}^{512}$). Naproti tomu reálné deskriptory popisují klíčové body pomocí vektoru reálných čísel (obvykle v prostoru s mnoha dimenzemi, např. $d_{\text{real}} \in \mathbb{R}^{64}$ pro zde použitý deskriptor *SURF*).

Výhodou binárních deskriptorů je často rychlost jejich výpočtu, pro příklad uveďme deskriptor *FREAK* [2], jehož autoři proklamují zrychlení v porovnání se *SURF* v řádu stovek procent. Problematické může být jejich shlukování (obvykle nelze využít klasické metriky vzdálenosti a jednotlivé bity navíc často mívají různou významnost). Často se vyskytujícím principem výpočtu binárního deskriptoru je porovnání výsledků rozdílu gausiánů (*Difference of Gaussians*) v definovaných oblastech v okolí popisovaného klíčového bodu. Výsledek je pak kvantizovaný do binární hodnoty a dostáváme jeden bit výsledného deskriptoru. Naproti tomu reálné deskriptory jsou často (např. *SURF*) založeny na výpočtu gradientů v okolí klíčového bodu, tyto pak mohou sloužit jako základ výsledného deskriptoru.

3.2.2 Vizuální slovník

Vizuální slovník (nebo původně pouze "slovník") je ve strojovém učení pojem pocházející z oblasti zpracování přirozené řeči (textu). V tomto kontextu byl (a stále je) využíván jako reprezentace textových dokumentů, například pro výpočet jejich podobnosti. Slovník byl v tomto případě tvořen vektorem celých čísel $V = (w_1, \ldots, w_n) : w_i \in \mathbb{N}_0$, kde každá položka představovala jedno slovo vyskytující se v textu (obvykle slovní základ, tedy bez vlivu skloňování) a položka $w_i = |\{j|W(i) = T(j), 1 \leq j \leq N\}|$ měla význam počtu výskytů i-tého slova ze slovníku $W(i)$ v textu T o N slovech. Podobnost dvojice dokumentů pak mohla být dána například kosinovou vzdáleností jejich vektorů V_1 a V_2 (rovnice 3.1). Poznamenejme, že v tomto jednoduchém příkladě zanedbáváme normalizaci vektoru V dle délky dokumentu a další problémy, které by bylo třeba ošetřit. Odtud se tento způsob reprezentace rozšířil i do oblasti počítačového vidění a odtud tedy název "vizuální slovník".

$$T_{\text{sim}} = \frac{V_1 \cdot V_2}{\|V_1\|\|V_2\|} \tag{3.1}$$

Kde T_{sim} má význam podobnosti dokumentů a $V_1 \cdot V_2$ je skalární součin vektorů V_1,V_2.

V této práci je popsaný způsob reprezentace využíván v mírně modifikované podobě, kde není udržován počet výskytů, ale pouze informace o alespoň jednom, nebo žádném výskytu. Tedy $w_i = 1$ pro $|\{j|W(i) = T(j), 1 \leq j \leq N\}| \geq 0$, jinak $w_i = 0$. Ze slov v textovém dokumentu se ve vizuálním slovníku stávají deskriptory (lokální příznaky) detekované ve snímku. Zde ovšem nastává první problém, pokud jsou deskriptory reálného typu (např. $d \in \mathbb{R}^{64}$) máme nekonečnou množinu unikátních deskriptorů, což implikuje nekonečný slovník, který jednak není realizovatelný a navíc by byl nepoužitelný (pravděpodobnost, že bude deskriptor detekován dvakrát se blíží nule). Je tedy v první řadě nutné

tento prostor omezit, což lze realizovat vhodným zvolením určitého počtu bodů v prostoru všech možných deskriptorů, které budou tvořit slova ve slovníku. Detekované deskriptory pak budou přiřazovány ke slovům na základě jejich vzdálenosti od bodu, který dané slovo představuje (zřejmě vždy k nejbližšímu).

Zůstává však problém, jak tyto body zvolit. Častým přístupem je použití shlukovacích algoritmů nad trénovací sadou dat, která by měla obsahovat deskriptory podobné těm, které se budou vyskytovat při následném používání takto vytvořeného slovníku. Shlukovacích algoritmů je dostupná celá řada, od klasického *k-means* po aktuální *m-BIRCH* [17], který je zástupcem online shlukovacích algoritmů a umožňuje tedy postupnou aktualizaci shluků. V prezentovaném řešení však byl využit jednoduchý shlukovací algoritmus (1), jehož implementaci obsahuje *FABMAP* (konkrétně jeho implementace *OpenFABMAP* [12]). Tento algoritmus umožňuje shlukování parametrizované pomocí maximální velikosti shluku a nevyžaduje předchozí znalost požadovaného počtu shluků. Shlukovací algoritmus 1 je dále popsán v oddíle 4.3 zabývajícím se jeho paralelní verzí a její implementací na *GPU*.

Data: Množina deskriptorů, $X = \{x_1, \ldots, x_n\}$, $x_i \in \mathbb{R}^{64}$
Data: Maximální velikost shluku, $D_{\max} \in \mathbb{R}^{+}$
Result: Množina středů shluků, $C = \{c_1, \ldots, c_k\}$, $c_i \in \mathbb{R}^{64}$
Nastav $I = \emptyset$;
foreach *deskriptor* $x_i \in X$ **do**
  Nastav $D_{\min} = \infty$;
  foreach *střed* $c_j \in I$ **do**
    $D_{\min} = min(\|x_i - c_j\|, D_{\min})$;
  end
  if $D_{min} \geq D_{max}$ **then**
    $I = I \cup \{x_i\}$;
  end
end
Nastav $S_i = \emptyset : 1 \leq i \leq |I|$;
foreach *deskriptor* $x_i \in X$ **do**
  $k = \arg\min_{1 \leq j \leq |I|} \|x_i - c_j\|$;
  $S_k = S_k \cup \{x_i\}$;
end
foreach $i \in \{1 \leq i \leq |I|\}$ **do**
  $C = C \cup \{\frac{\sum_{s \in S_i} s}{|S_i|}\}$;
end

Algoritmus 1: Generování vizuálního slovníku

3.3 Výskytový model

Algoritmus *FABMAP* tvořící jeden ze dvojice pravděpodobnostních modelů zachycuje vztahy mezi výskyty vizuálních slov ve snímcích a na jejich základě je schopný určit pravděpodobnost, že dva snímky jsou zobrazením shodné scény. Případně za pomoci generativního modelu prostředí dokáže určit pravděpodobnost, že snímek pochází z dosud neznámé lokace. Pracuje tedy s první složkou reprezentace lokací a jejich pozorování (E_i a Z_k). Činnost algoritmu lze rozdělit do tří režimů:

- Režim učení modelu prostředí (nejde o učení samotné topologické mapy).

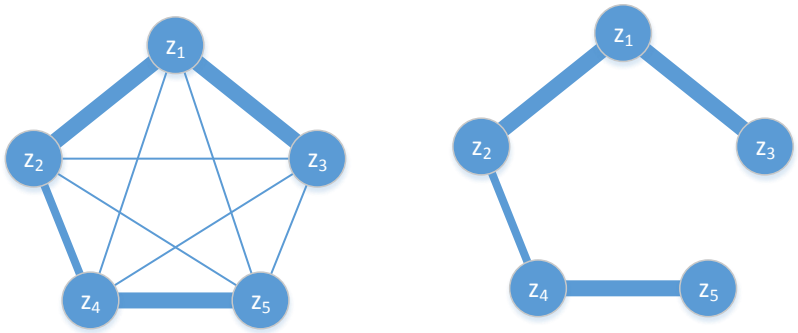
- Režim učení (záznamu) topologické mapy (trasy pohybu).
- Lokalizace v naučeném topologickém modelu.

Poslední dva body se mohou prolínat, algoritmus je tedy schopný rozšiřovat model prostředí v průběhu lokalizace (například umožňuje přidat novou lokaci v případě pozorování dosud neznámé scény), jde tedy o obdobu *SLAM*.

3.3.1 Model pozorování

Aby bylo možné použít výše zmíněnou reprezentaci pozorování pomocí tzv. "Bag-of-Words" pro výpočet pravděpodobnosti, že dvě pozorování pochází ze stejné lokace, je třeba vytvořit jejich generativní model. Účelem tohoto modelu je zachytit jak závislosti mezi slovy ve slovníku, tak informační zisk o aktuální lokaci na jejich základě. Vypovídací hodnotu má nejen výskyt slov, či jejich společný výskyt, ale stejně tak jejich absence. Tyto vztahy lze popsat rozložením pravděpodobnosti nad diskrétními proměnnými v množině $Z = \{z_1, z_2, \ldots, z_n\}$ (obrázek 3.6a). Problémem je náročnost výpočtu takové distribuce, proto je využito aproximace těchto závislostí pomocí stromové struktury v podobě tzv. "Thin Junction Tree" [3] šířky 1, který lze vytvořit pomocí algoritmu *Chow Liu* popsaného v [6]. Mějme tedy distribuční funkci $Q(Z)$ nad n diskrétními proměnnými, algoritmus pak sestrojí Bayesovskou síť v podobě stromu, která reprezentuje rozložení pravděpodobnosti $P(Z)$, které aproximuje původní rozložení $Q(Z)$, tak aby byla minimalizována ztráta informace. Cílem je tedy minimalizovat *Kullback-Leibler divergenci* danou rovnicí 3.2.

$$D_{KL}(Q\|P) = \sum_{Z} Q(Z) ln \frac{Q(Z)}{P(Z)} \tag{3.2}$$



(a) Úplný graf vzájemné informace (b) Maximální kostra (Chow Liu tree)

Obrázek 3.6: Graf vzájemné informace (šířka hrany představuje množství informace)

Chow Liu Tree je vlastně maximální kostra grafu vzájemné informace (obrázek 3.6b) proměnných $z_i, z_j \in Z$. Jeho výpočet je možné provést sestrojením úplného grafu nad n vrcholy, kde váha hrany (z_i, z_j) je rovna vzájemné informaci mezi proměnnými $I(z_i, z_j)$ na vrcholech. Vzájemná informace $I(z_i, z_j)$ je pak dána vztahem 3.3.

$$I(z_i, z_j) = \sum_{z_i \in \Omega, z_j \in \Omega} p(z_i, z_j) log \frac{p(z_i, z_j)}{p(z_i)p(z_j)} \tag{3.3}$$

Kde $\Omega = \{0, 1\}$ a $z_i, z_j \in Z$ jsou tedy binární proměnné.

3.3.2 Reprezentace lokace

Z pohledu výskytového modelu je tedy svět modelován jako kolekce n_k nepřekrývajících se míst, v čase k tedy $\mathcal{L}^k = \{L_1, \ldots, L_{n_k}\}$. Každá lokace je pak popsána dvojicí $L_i = (E_i, H_i)$, kde $E_i = \{p(e_1 = 1|L_i), \ldots, p(e_{|V|} = 1|L_i)\}$, kde $p(e_j = 1|L_i)$ představuje pravděpodobnost, že dané slovo bylo senzorem v lokaci L_i pozorováno. Pojem času k zde zavádíme ze dvou důvodů, jednak umožňuje zachytit aktualizace mapy a také integraci modelu pohybu robota.

Aby bylo možné vypočítat pravděpodobnost $p(L_i|\mathcal{Z}^k)$, kde $\mathcal{Z}^k$ je množina všech pozorování od počátku do času k, je třeba definovat model senzoru (tedy kamery, detektoru a deskriptoru klíčových bodů). Ten lze popsat pomocí pravděpodobnosti "přehlédnutí" vizuálního slova ve scéně $p(z_i = 0|e_i = 1)$ a pravděpodobnosti falešné detekce $p(z_i = 1|e_i = 0)$. Zde vycházíme z předpokladu, že tyto pravděpodobnosti jsou nezávislé, jak na lokaci, tak na konkrétním pozorovaném slově.

3.3.3 Výpočet lokace

K výpočtu pravděpodobnosti $p(L_i, \mathcal{Z}^k)$ využívá *FABMAP* rekurzivní aplikace Bayesova teorému, výsledkem je tedy rovnice 3.4.

$$p(L_i, \mathcal{Z}^k) = \frac{p(Z_k|L_i)p(L_i|\mathcal{Z}^{k-1})}{p(Z_k|\mathcal{Z}^{k-1})} \tag{3.4}$$

Kde $p(L_i|\mathcal{Z}^{k-1})$ je apriorní pravděpodobnost, že se nacházíme v lokaci L_i založená na předchozím odhadu lokace případně modifikovaném pomocí modelu pohybu. $p(Z_k|\mathcal{Z}^{k-1})$ je normalizační koeficient, který také zachycuje možnost že pozorování Z_k pochází z dosud neznámé lokace (viz. 3.3.4).

Pravděpodobnost pozorování $p(Z_k|L_i)$ pak lze vyjádřit za použití modelu pozorování popsaného v oddíle 3.3.1, rovnicí 3.5.

$$p(Z_k|L_i) \approx p(z_r|L_i) \prod_{q=1}^{|v|} p(z_q|z_{p_q}, L_i) \tag{3.5}$$

Pro vyhodnocení $p(z_r|L_i)$ a $p(z_q|z_{p_q}, L_i)$ je třeba vzít v úvahu model senzorů popsaný výše, dostáváme rovnici 3.6 (obdobně pro $p(z_r|L_i)$, kde není třeba uvažovat rodiče). Dalším zjednodušením je považovat chyby detektorů za nezávislé na lokaci a pravděpodobnost výskytu slova $p(e_j)$ za nezávislou na pozorování jiného slova z_i pro $\forall i \neq j$. Dostáváme tedy vztah 3.7.

$$p(z_q|z_{p_q}, L_i) = \sum_{s_{e_q} \in \{0,1\}} p(z_q|e_q = s_{e_q}, z_{p_q}, L_i)p(e_q = s_{e_q}|z_{p_q}, L_i) \tag{3.6}$$

$$p(z_q|z_{p_q}, L_i) = \sum_{s_{e_q} \in \{0,1\}} p(z_q|e_q = s_{e_q}, z_{p_q})p(e_q = s_{e_q}|L_i) \tag{3.7}$$

Posledním krokem je vyjádření $p(z_q|e_q, z_{p_q})$ daty, která jsme schopni získat během procesu trénování modelu, dostáváme tak rovnici 3.8.

$$p(z_q = s_{z_q}|e_q = s_{e_q}, z_p = s_{z_p}) = (1 + \frac{\alpha}{\beta})^{-1} \tag{3.8}$$

Kde $s_{z_q}, s_{e_q}, s_{z_p} \in \{0, 1\}$ a
$\alpha = p(z_q = s_{z_q})p(z_q = \overline{s_{z_q}}|e_q = s_{e_q})p(z_q = \overline{s_{z_q}}|z_p = s_{z_p})$
$\beta = p(z_q = \overline{s_{z_q}})p(z_q = s_{z_q}|e_q = s_{e_q})p(z_q = s_{z_q}|z_p = s_{z_p})$

3.3.4 Detekce neznámých lokací

Jak již bylo zmíněno, algoritmus zachycuje možnost navštívení nové lokace v normalizačním členu $p(Z_k|\mathcal{Z}^{k-1})$ v rekurzivním výpočtu Bayesova teorému (rovnice 3.4). Pokud připustíme existenci lokací, které nebyly navštíveny, dojde k hypotetickému rozdělení množiny všech lokací $W = M \cup \overline{M}$, kde M je množina již známých lokací, $\overline{M}$ množina dosud nenavštívených lokací a zřejmě platí, že $M \cap \overline{M} = \emptyset$. Pravděpodobnost $p(Z_k|\mathcal{Z}^{k-1})$ lze pak vyjádřit rovnicí 3.9 (v případě, že nepřipustíme možnost existence nenavštívených míst se v rovnici vypustí suma přes prvky množiny $\overline{M}$).

$$p(Z_k|\mathcal{Z}^{k-1}) = \sum_{m \in M} p(Z_k|L_m)p(L_m|\mathcal{Z}^{k-1}) + \sum_{n \in \overline{M}} p(Z_k|L_n)p(L_n|\mathcal{Z}^{k-1}) \tag{3.9}$$

Je zřejmé, že sumu přes neznámá místa v $\overline{M}$ není možné explicitně vyčíslit. Autoři algoritmu se proto rozhodli využít náhodného vzorkování nových míst za pomoci Bayesovské sítě vytvořené jako generativní model pozorování (tento model popisuje oddíl 3.3.1). Takto lze vypočítat $p(Z_k|L_n)$ pro každé vzorkované místo, apriorní pravděpodobnost výskytu nového místa $p(L_n|\mathcal{Z}^{k-1})$ je pak specifikována uživatelem.

Použití normalizačního členu 3.9 spoléhá na předpoklad, že pokud pozorování Z_k je unikátní a navíc je výrazně podobné některému z již známých míst, pak je pravděpodobnost, že se bude podobat některému z náhodně vzorkovaných modelů míst velmi nízká. Naopak, pokud toto místo není unikátní a (nebo) se výrazně nepodobá některému ze zmapovaných míst, pak je pravděpodobnost, že bude odpovídat náhodně generovaným vzorkům vyšší. Tím dochází k růstu $p(Z_k|\mathcal{Z}^{k-1})$ a poklesu $p(L_i|\mathcal{Z}^k)$ pro všechna známá místa $L_i \in M$.

Alternativní možností vyčíslení sumy přes neznámá místa je využití modelu průměrné lokace ("mean field approximation"). Aproximace tohoto typu by pak měla tvar rovnice 3.10.

$$\sum_{n \in \overline{M}} p(Z_k|L_n)p(L_n|\mathcal{Z}^{k-1}) \approx p(Z_k|L_{\text{avg}}) \sum_{n \in \overline{M}} p(L_n|\mathcal{Z}^{k-1}) \tag{3.10}$$

Kde $p(Z_k|L_{\text{avg}})$ lze vyčíslit podobně jako $p(Z_k|L_i)$ v rovnici 3.5, kde se neuvažuje L_i.

3.3.5 Model nové lokace

Chybějícím článkem algoritmu je zařazení nových míst do množiny M. Mějme tedy nové místo $E_j = \{p(e_1 = 1|L_j), \ldots, p(e_{|V|} = 1|L_j)\}$ a jeho pozorování Z_k. Přidání místa je pak provedeno tak, že $p(e_i = 1|L_j)$ je nejdříve pro všechna $i : 1 \leq i \leq |V|$ inicializováno minimální pravděpodobností $p(e_i = 1)$ (tedy všechna slova se v daném místě vyskytují s nějakou minimální pravděpodobností, která je odvozena při učení modelu prostředí) a následně je provedena aktualizace této pravděpodobnosti na základě pozorování dle rovnice 3.11. Zde

lze provést další aproximace o nezávislosti: $p(e_i|Z) \approx p(e_i|z_i)$ (tedy výskyt slova závisí pouze na jeho pozorování) a $p(z_i|e_i, L_j) \approx p(z_i|e_i)$ (pozorování slova závisí pouze na jeho existenci), tím dostáváme rovnici 3.12 pro aktualizaci nové lokace.

$$p(e_i = 1|L_j, \mathcal{Z}^k) = \frac{p(Z_k|e_i, L_j)p(e_i|L_j, \mathcal{Z}^{k-1})}{\sum_{s_e \in \{0,1\}} p(Z_k|e_i = s_e, L_j, \mathcal{Z}^{k-1})p(e_i = s_e|L_j, \mathcal{Z}^{k-1})} \tag{3.11}$$

$$p(e_i = 1|L_j, \mathcal{Z}^k) = \frac{p(z_i|e_i)p(e_i|L_j, \mathcal{Z}^{k-1})}{\sum_{s_e \in \{0,1\}} p(z_i|e_i = s_e)p(e_i = s_e|L_j, \mathcal{Z}^{k-1})} \tag{3.12}$$

3.4 Prostorový model

Prostorový model pozorování a prostředí zachycuje strukturu snímku jako vzdálenosti mezi dvojicemi klíčových bodů. Jeho cíl je shodný s výskytovým modelem, ale pracuje nad prostorovou konfigurací a proto umožňuje rozlišit scény, které se ve výskytovém modelu jeví jako shodné. Zvláště významnou schopností je rozlišit měřítko scény, které je indikátorem pohybu podél osy pohledu. Na druhou stranu je (jak bylo zmíněno) tato reprezentace invariantní vůči transformacím jako rotace kamery kolem pohledové osy. Formálně tedy tento pravděpodobnostní model pracuje nad složkami H_i a D_k modelů scény a jejího pozorování.

Činnost tohoto modelu lze opět rozdělit do shodných fází jako v případě modelu výskytového. V prvním kroku dochází k učení modelu prostředí, kde je cílem získání funkcí rozložení pravděpodobností $\{h_{ij}(x)|1 \leq i, j \leq |V|\}$ v rámci prostředí ve kterém se robot pohybuje. Tyto apriorní pravděpodobnosti pak mají dvojí využití, jednak jsou využity jako apriorní rozložení pro dvojice slov, které nebyly ve snímku pozorovány, dalším využitím je detekce neznámých lokací. Druhým krokem je tedy učení topologické mapy, kde je globální rozložení $h_{ij}(x)$ aktualizováno pomocí pozorování scény, která danému místu náleží. Posledním krokem je pak výpočet aktuální lokace v topologické mapě na základě aktuálního pozorování.

3.4.1 Model pozorování

Pozorování zachycuje vzdálenost (v pixelech) mezi klíčovými body ve snímku, každé pozorování scény je tedy tvořeno hranově značeným grafem $G = (K, D, \Delta)$, kde $K \in \mathbb{N}^2 \times V$ je množina klíčových bodů ve snímku přiřazených za pomoci deskriptorů k příslušným slovům vizuálního slovníku, $D \in K \times K$ je množina hran a funkce značení $\Delta : D \to \mathbb{R}^+$ zachycuje vzdálenost mezi koncovými body dané hrany.

Model však neuvažuje klíčové body jako takové, ale pouze přítomnost daného bodu s deskriptorem náležejícím určitému slovu ve slovníku. Upravíme proto graf G sloučením vrcholů na základě jejich složky $v \in V$, touto úpravou dostáváme multigraf $G' = (V', D')$, kde $V' = \{v|(x, y, v) \in K\}$ a $D' = \{(v_1, v_2, \Delta(k_1, k_2))|k_1 = (x_1, y_1, v_1), k_2 = (x_2, y_2, v_2) \in K, (k_1, k_2) \in D\}$. Pokud se tedy vyskytne několik klíčových bodů, které náleží stejnému slovu vznikne v G' násobná hrana ohodnocená více vzdálenostmi.

Nad tímto modelem je dále třeba definovat model senzoru měření vzdálenosti mezi slovy, ten zahrnuje nepřesnost detekce klíčového bodu a také umožňuje vnesení záměrné tolerance v případě malých změn orientace či pozice kamery. Tyto nepřesnosti jsou modelovány pomocí normálního rozložení pravděpodobnosti kolem detekované vzdálenosti, jehož rozptyl je dán uživatelským parametrem. Tedy $p(d_{ij}|h_{ij} = x) \sim \mathcal{N}(x, \sigma_{\text{err}})$, kde d_{ij} je pozorovaná vzdálenost a h_{ij} je skutečná vzdálenost mezi klíč. body ve scéně.

3.4.2 Reprezentace lokace

Na základě modelu pozorování (ohodnoceného multigrafu G') a modelu senzoru dále definujeme lokaci. Jak již bylo popsáno, je svět modelován jako kolekce n_k nepřekrývajících se míst v čase k, tedy $\mathcal{L}^k = \{L_1, \ldots, L_{n_k}\}$. Lokace je pak dvojice $L_i = (E_i, H_i)$, kde pro tento model má význam pouze druhá složka $H_i = \{h_{ij}(x) | 1 \leq i, j \leq |V|\}$, kde $h_{ij}(x)$ je funkce hustoty pravděpodobnosti a tedy $p(h_{ij} = x | L_i) = h_{ij}(x)$. Souvislost mezi lokací (scénou) a odpovídající množinou H_i zachycuje obrázek 3.5.

3.4.3 Výpočet lokace

Výpočet lokace je opět obdobou výpočtu použitého v algoritmu *FABMAP* a jde tedy o rekurzivní aplikaci Bayesova teorému. V případě prostorového modelu bude mít tvar rovnice 3.13.

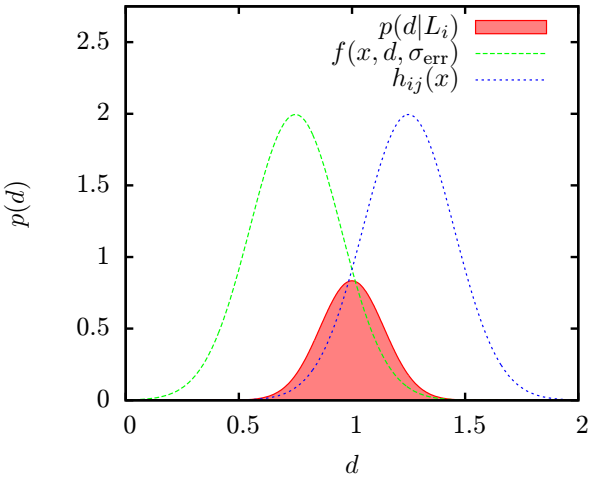
$$p(L_i, \mathcal{D}^k) = \frac{p(D_k | L_i) p(L_i | \mathcal{D}^{k-1})}{p(D_k | \mathcal{D}^{k-1})} \tag{3.13}$$

Kde $p(L_i | \mathcal{D}^{k-1})$ je apriorní pravděpodobnost, že se nacházíme v lokaci L_i dle předchozích pozorování modifikovaných modelem pohybu. $p(D_k | \mathcal{D}^{k-1})$ je normalizační koeficient, který také zachycuje výskyty pozorování neznámých lokací. Zde však podobnost s výskytovým modelem končí, pravděpodobnost $p(D_k | L_i)$ je v tomto modelu dána rovnicí 3.14.

$$p(D_k | L_i) = \prod_{i,j=1}^{|V|} \prod_{d \in C_{ij}} p(d | L_i) \tag{3.14}$$

Kde $C_{ij} = \{d | (v_i, v_j, d) \in D'\}$ je množina všech pozorovaných vzdáleností (v pozorování D_k) mezi slovy $v_i, v_j \in V$ a $p(d | L_i)$ je pravděpodobnost, že v lokaci L_i je mezi slovy pozorována vzdálenost d.

Pravděpodobnost $p(d | L_i)$ lze dále vyjádřit pomocí distribuční funkce $h_{ij}(x)$ dané lokace L_i a modelu senzoru (viz. obrázek 3.7), který změřil vzdálenost d při pozorování D_k. Po této úpravě dostáváme rovnici 3.15.



Obrázek 3.7: Výpočet $p(d | L_i)$ pomocí marginalizace.

$$p(D_k|L_i) = \prod_{i,j=1}^{|V|} \prod_{d \in C_{ij}} \int_{-\infty}^{+\infty} f(x, d, \sigma_{\text{err}}) h_{ij}(x)\, \mathrm{d}x \tag{3.15}$$

Kde $f(x, \mu, \sigma) = \frac{1}{\sigma}\phi(\frac{x-\mu}{\sigma})$ je funkce hustoty pravděpodobnosti modelu senzoru se středem v naměřené hodnotě ($p(d_{ij}|h_{ij} = x)$) a funkce $h_{ij}(x) \in H_i$ je distribuční funkce nad vzdálenostmi v lokaci L_i (tedy $p(h_{ij} = x|L_i)$). Marginalizací přes x pak vyloučíme proměnnou $h_{ij} = x$ a dostáváme $p(d|L_i)$.

Aby bylo možné rovnici 3.15 vyčíslit, je nutné zavést aproximaci funkcí $f(x, \mu, \sigma)$ a h_{ij}. Tyto funkce mohou být multimodální a proto byla zvolena aproximace, kde jsou reprezentovány pomocí histogramů, tedy rozdělení definičního oboru funkce na R intervalů šířky Δ. Zde je ovšem nutné tento nekonečný definiční obor omezit, což je řešeno normalizací vzdálenosti mezi slovy do intervalu $\langle 0, 1\rangle$. Normalizaci lze snadno realizovat rovnicí 3.16. Pokud je tedy podstatná oblast těchto funkcí omezena do intervalu $\langle 0, 1\rangle$ dostáváme $R = 1/\Delta$ košů histogramu. Každý takový histogram má pak význam distribuční funkce ve smyslu $p(d = b_k)$, kde b_k je index koše a platí $1 \leq k \leq R$ (3.17).

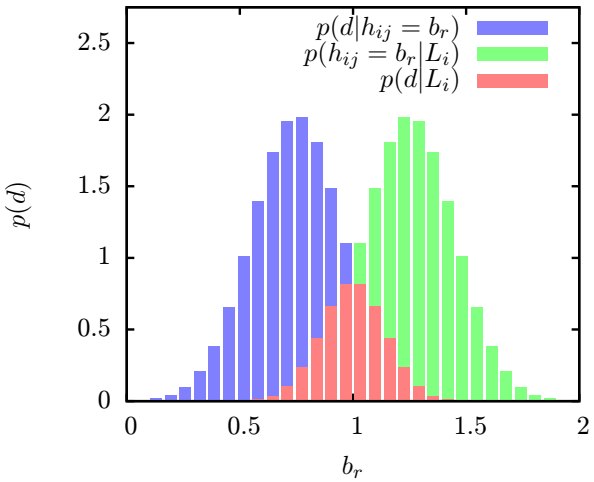
$$D_{\text{norm}}(p_1, p_2) = \frac{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}{\sqrt{w^2 + h^2}} \tag{3.16}$$

Kde $p = (x, y)$ je bod ve snímku a w, h je šířka a výška snímku (v pixelech).

$$p(d = b_k) = p((k - 1)\Delta \leq d \leq k\Delta) \tag{3.17}$$

Kde $p(d = b_k)$ je tedy pravděpodobnost, že hodnota d se bude nacházet v intervalu k-tého koše histogramu.

Vyčíslitelná podoba rovnice 3.15 tedy za použití zmíněné aproximace (viz. obrázek 3.8) bude mít tvar 3.18. Každá známá lokace tedy obsahuje pro každou dvojici slov histogram distribuční funkce jejich vzdáleností $p(h_{ij} = b_r|L_i)$, který je vypočten na základě globálního modelu (3.4.4) a pozorování dané scény (3.4.5).



Obrázek 3.8: Diskrétní výpočet $p(d|L_i)$ pomocí marginalizace nad histogramy.

$$p(D_k|L_i) = \prod_{i,j=1}^{|V|} \prod_{d \in C_{ij}} \sum_{r=1}^{R} p(d|h_{ij} = b_r) p(h_{ij} = b_r|L_i) \tag{3.18}$$

Kde $p(d|h_{ij} = b_r)$ odpovídá diskretizované podobě funkce $f(x, d, \sigma_{\text{err}})$ a $p(h_{ij} = b_r|L_i)$ funkci $h_{ij}(x)$.

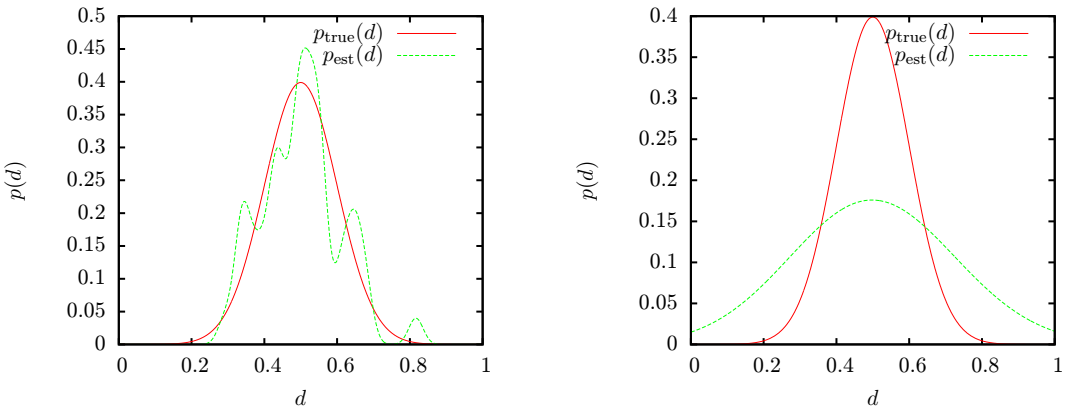
3.4.4 Model prostředí

Jak již bylo zmíněno, modelem prostředí je množina funkcí hustoty pravděpodobnosti $\{h_{ij}(x)|1 \leq i, j \leq |V|\}$, které jsou reprezentovány pomocí histogramů $p(h_{ij} = b_r)$. Cílem je tedy rekonstrukce distribuční funkce na základě konečné množiny vzorků a v případě reprezentace histogramem její opětovné navzorkování (do košů histogramu). Jedním z používaných přístupů k řešení tohoto problému je tzv. "Kernel Density Estimation" (*KDE*) [20]. Základem tohoto přístupu je jádrová funkce $K(\cdot)$, jež musí splňovat podmínky: $K(x) \geq 0$ a $\int K(x)\,\mathrm{d}x = 1$. Výsledná hustota v bodě x je pak dána lineární kombinací jádrových funkcí zarovnaných na trénovacích datech $\{x_1, \ldots, x_n\}$. Na vzorky v trénovacích datech je dále kladen požadavek, aby byly nezávislé a identicky distribuované dle nějaké distribuční funkce (předpoklad *i.i.d*).

Jednou z často používaných jádrových funkcí $K(\cdot)$ je gausián se středem v nule a rozptylem jedna (tedy $\phi(x)$), *KDE* pak lze zapsat rovnicí 3.19. Metoda *KDE* vyžaduje ke své činnosti parametr h, který definuje šířku pásma jádrové funkce (tzv. "bandwidth"). Tento parametr definuje přesnost výsledného modelu. Pokud je tedy zvolena příliš nízká hodnota h dochází k porušení schopnosti zobecnění (tzv. "overfitting") a naopak příliš vysoké hodnoty způsobují vznik modelu, který zanedbává příliš mnoho detailů (obrázek 3.9).

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) = \frac{1}{nh} \sum_{i=1}^{n} \phi\left(\frac{x - x_i}{h}\right) \tag{3.19}$$

Kde x je cílový vzorek, n je počet vzorků trénovací množiny, h je šířka jádra a $\phi(x) = exp(-\frac{1}{2}x^2)/\sqrt{2\pi}$ je Gausova funkce.



(a) Over-fitting

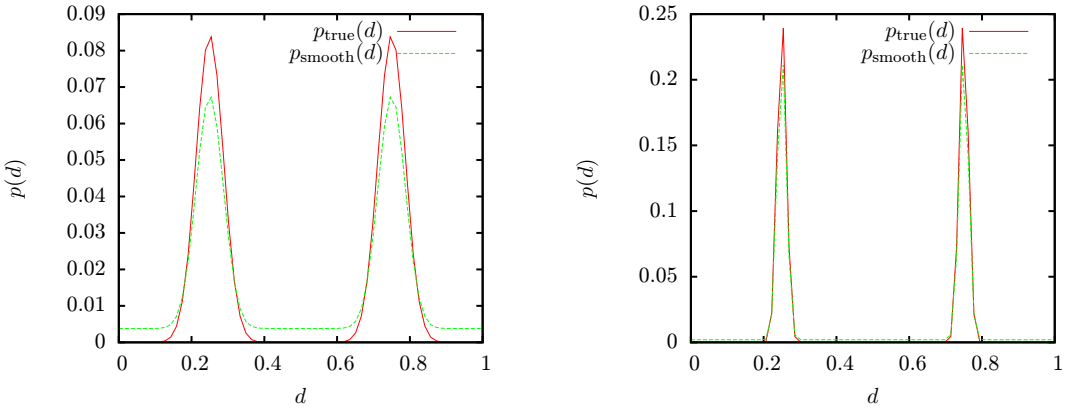
(b) Under-fitting

Obrázek 3.9: Vliv parametru h na výpočet distribuční funkce metodou *KDE*.

Krom problému určení parametru h trápí tuto metodu ještě problémy spojené s nedostatkem vzorků v trénovací sadě. Příkladem může být dvojice slov, která se vyskytne pouze v jediném snímku a může tak způsobit degenerativní hodnoty ($p(h_{ij} = b_r) \in \{0, 1\}$) v některých koších histogramu. Pro řešení tohoto typu problémů se využívá mnoho různých technik vyhlazení distribučních funkcí [9]. V této práci byl použit pro vyhlazení výsledků *KDE* vztah 3.20. Vliv vyhlazení pak zachycuje obrázek 3.10.

$$p_{\text{smooth}} = \frac{n}{n+m}p + \frac{m}{n+m}p_{\text{prior}} \tag{3.20}$$

Kde p_{smooth} je vyhlazená pravděpodobnost, n je počet vzorků trénovacích dat, $m = \sqrt{n}$ je vyhlazovací konstanta odvozená v [11], p zastupuje pravděpodobnost vypočtenou pomocí *KDE* a $p_{\text{prior}} = 1/R$ je apriorní pravděpodobnost bez žádné znalosti o distribuční funkci.



(a) Vyhlazení pro 10 vzorků

(b) Vyhlazení pro 50 vzorků

Obrázek 3.10: Vliv počtu vzorků na odhad distribuční funkce (*KDE*) a vyhlazování.

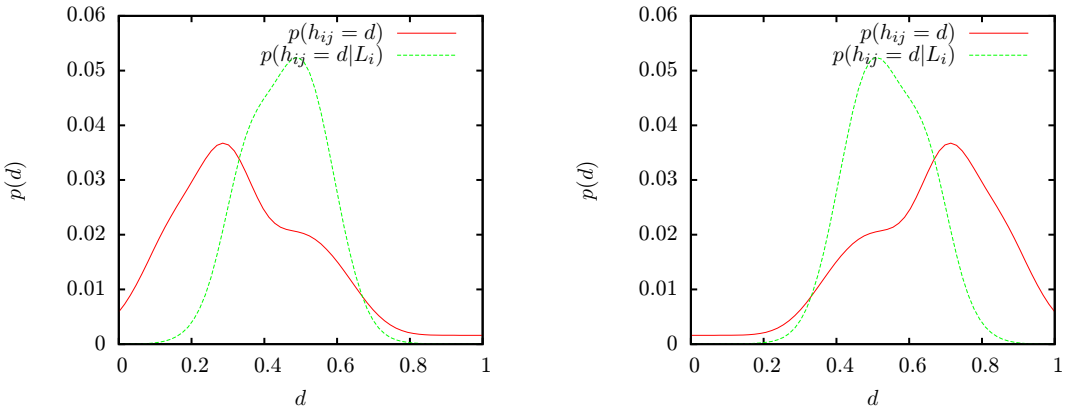
3.4.5 Aktualizace lokace

Fáze učení množiny známých lokací M je pak obdobou učení nových lokací ve výskytovém modelu, máme tedy apriorní znalost o lokacích, které jsou v daném prostředí obvyklé (v podobě histogramů nad vzdálenostmi mezi dvojicemi slov). Model nové lokace je vytvořen aktualizací těchto pravděpodobností na základě pozorování příslušné scény.

Je-li tedy přidávána nová lokace L_{new} jsou její histogramy nastaveny dle příslušného globálního histogramu na $p(h_{ij} = b_r|L_{\text{new}}) = p(h_{ij} = b_r)$ a následně aktualizovány dle rovnice 3.21, která na základě aktuálních znalostí o dané lokaci (pro $L_i = L_{\text{new}}$ je téměř žádná), modelu senzoru a aktuálního pozorování definuje nový model daného místa L_i (obrázek 3.11).

$$p(h_{ij} = b_r|L_i, \mathcal{D}^k) = \frac{p(d_{ij}^n|b_r)p(h_{ij} = b_r|L_i, \mathcal{D}^{k-1})}{p(d_{ij}^n|L_i)} \tag{3.21}$$

Pro $d_{ij}^n \in \{d_{ij}^1, \ldots, d_{ij}^N\}$ tedy všechny pozorované vzdálenosti mezi slovy $v_i, v_j \in V$ v daném snímku, $p(d_{ij}^n|b_r)$ je pak jedna z pozorovaných vzdáleností modifikovaná modelem nejistoty senzoru, $p(h_{ij} = b_r|L_i, \mathcal{D}^{k-1})$, je původní model místa před pozorováním D_k a $p(d_{ij}^n|L_i)$ je normalizační člen.



Obrázek 3.11: Vliv modelu prostředí $p(h_{ij} = d)$ na model snímku při jeho vložení do množiny M.

3.4.6 Detekce neznámé lokace

Detekce neznámých lokací reprezentovaná normalizačním členem $p(D_k|\mathcal{D}^{k-1})$ je pak v duchu výskytového modelu definována pomocí rozdělení světa na množiny zmapovaných a neznámých míst $W = M \cup \overline{M}$, kde $M \cap \overline{M} = \emptyset$. Pravděpodobnost $p(D_k|\mathcal{D}^{k-1})$ pak lze opět vyčíslit dle rovnice 3.22, kde je opět možné sumu přes neznámá místa vyčíslit pomocí náhodných vzorků nad modelem prostředí, nebo za využití průměrné lokace vytvořené taktéž na základě tohoto modelu. Průměrná lokace má vlastnosti marginálních hodnot histogramů výskytu vzdáleností a tedy $p(h_{ij} = b_r|L_{\text{avg}}) = p(h_{ij} = b_r)$, suma přes $n \in \overline{M}$ pak nabude tvaru 3.23.

$$p(D_k|\mathcal{D}^{k-1}) = \sum_{m \in M} p(D_k|L_m)p(L_m|\mathcal{D}^{k-1}) + \sum_{n \in \overline{M}} p(D_k|L_n)p(L_n|\mathcal{D}^{k-1}) \tag{3.22}$$

$$\sum_{n \in \overline{M}} p(D_k|L_n)p(L_n|\mathcal{D}^{k-1}) \approx p(D_k|L_{\text{avg}}) \sum_{n \in \overline{M}} p(L_n|\mathcal{D}^{k-1}) \tag{3.23}$$

Kde $p(D_k|L_{\text{avg}})$ lze vyčíslit pomocí 3.18 a $\sum_{n \in \overline{M}} p(L_n|\mathcal{D}^{k-1})$ je pravděpodobnost výskytu neznámého místa daná uživatelským parametrem.

Kapitola 4

Programová realizace

Kapitola nazvaná programová realizace popisuje návrh struktury programu a některé zajímavé části jeho implementace. Ústředními tématy jsou implementace paralelní verze shlukovacího algoritmu (algoritmus 1) na *GPU* a implementace prostorového modelu lokalizační metody popsané v oddíle 3.4. Přestože se obecné výpočty na *GPU* (tedy tzv. *GPGPU*) postupně stávají standardem uvádíme stručný pohled na moderní architekturu *GPU*, které bylo pro řešení shlukování zvoleno.

Pro realizaci výše popsaného řešení problému lokalizace byl zvolen jazyk *C++* a to především pro svou rozšířenost a s ní spojenou dostupnost potřebných knihoven (jejichž nativním rozhraním je jazyk *C/C++*). Další vlastností jazyka *C++*, která ho činí vhodným kandidátem, je jeho ideální míra abstrakce nad hardwarem, kde umožňuje operace na nízké úrovni a zároveň nabízí podporu vysokoúrovňových abstrakcí, jako je objektově orientované programování.

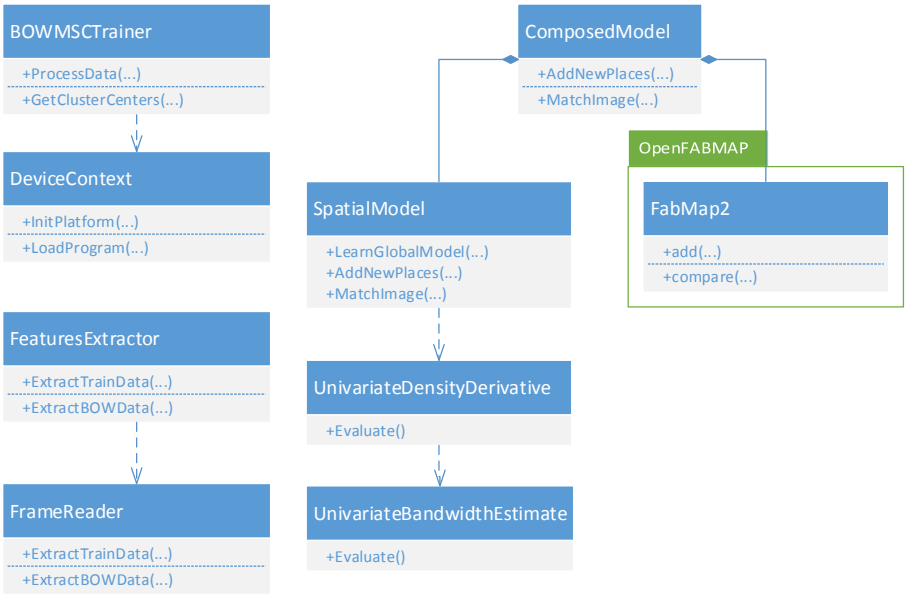
Samotná implementace pak využívá řadu knihoven k řešení různých podproblémů. Pro zpracování obrazu a detekci lokálních příznaků je využita knihovna *OpenCV*, pro podporu operací nad vektory a maticemi je využívána šablonová knihovna *Eigen* a pro zajištění přístupu ke *GPU* je využito rozhraní *OpenCL*. K řešení problému výpočtu distribuční funkce na základě vzorků (*KDE*) a výpočtu potřebného parametru h je využita volně dostupná implementace algoritmu popsaného v [20], která byla doplněna o knihovnu *Levmar* (implementace optimalizačního algoritmu *Levenberg–Marquardt*) pro řešení nelineárního problému minimálních čtverců.

4.1 Struktura implementace

Pro vytvoření implementace navrženého řešení bylo zvoleno objektové paradigma a její funkční celky jsou tedy obaleny třídami, jejichž propojení zachycuje diagram tříd na obrázku 4.1. Následující výčet pak podává bližší vysvětlení účelu jednotlivých tříd.

- **BOWMSCTrainer** – Obaluje funkcionalitu učení vizuálního slovníku, který je základní reprezentací dat v modelech. Obsahuje tedy implementaci paralelního shlukovacího algoritmu postaveného na platformně *OpenCL*.
- **ComposedModel** – Realizuje spojení výsledků lokalizace na základě obou modelů do konečného výsledku.
- **DeviceContext** – Jde o třídu obalující používanou funkcionalitu platformy *OpenCL*.

- **FabMap2** – Je implementací výskytového modelu (algoritmu *FABMAP* v knihovně *OpenFABMAP*).
- **FeaturesExtractor** – Zajišťuje extrakci lokálních deskriptorů a "Bag-of-Words" ze vstupních snímků (jak pro potřeby učení modelů, tak samotné lokalizace).
- **FrameReader** – Obaluje funkcionalitu související s načítáním vstupních dat (jednotlivých snímků, či videosekvence).
- **UnivariateDensityDerivative** – Implementuje výpočet hustoty pravděpodobnosti na základě vzorků (při učení prostorového modelu) metodou *KDE*.
- **UnivariateBandwidthEstimate** – Tvoří algoritmus pro určení šířky pásma h (na základě vzorků) pro použití metodou *KDE*.
- **SpatialModel** – Implementuje samotný prostorový model pro lokalizaci. Obsahuje proto metody sloužící pro vytvoření tohoto modelu, učení topologické mapy a jeho použití.



Obrázek 4.1: Zjednodušený diagram tříd programové realizace

4.2 Prostorový model

Prostorový model implementovaný v rámci třídy *SpatialModel* tedy umožňuje operace učení modelu prostředí, přidávání míst do topologické mapy, porovnání nového snímku s místy v mapě a také uložení a načtení modelu prostředí ze souboru. V tomto oddílu je postupně popsána implementace těchto základních operací nad tímto modelem.

Základní úlohou je zde samotné uložení modelu prostředí, který má podobu trojice (i, j, h_{ij}), kde i,j jsou indexy do slovníku a $h_{ij} \in \mathbb{R}^k$ (v implementaci $k = 64$) je histogram vzdáleností mezi slovy w_i a w_j. Pro tento model prostředí je důležité, aby jej bylo možné

indexovat pomocí první a druhé složky (tedy indexy slov i,j), je proto implementován jako hash tabulka. Tento přístup umožňuje vyhledání položky v čase $O(1)$ a také v praxi se osvědčil jako rychlejší, než binární strom.

4.2.1 Učení modelu prostředí

Učení globálního modelu prostředí pak probíhá tak, že algoritmu jsou předána data v podobě seznamu struktur (kde každá obsahuje: číslo snímku, jeho velikost, jeho popis výskytem slov, seznamem klíčových bodů a jejich přiřazením ke slovům). Ze všech těchto struktur je vytvořena hash tabulka, která však na místo složky h_{ij} obsahuje seznam všech pozorovaných vzdáleností mezi slovy w_i a w_j. Pozorované vzdálenosti jsou normalizovány do intervalu $\langle 0, 1\rangle$ (pro odstranění závislosti na rozlišení snímků). Následně je pro každou položku této hash mapy určen parametr h a metodou *KDE* vytvořen histogram (implementaci těchto výpočtů je možné nalézt v [20]). Histogram je následně vyhlazen (dle 3.20), normalizován a uložen do tabulky tvořící model prostředí.

4.2.2 Tvorba topologické mapy

Vstupem této metody je struktura shodná se vstupem trénovacích dat při tvorbě modelu prostředí. Pro každý vstupní snímek je pak vygenerována množina vzdáleností mezi dvojicí slov, ale tak, že se tato dvojice již musí nacházet v modelu prostředí. Model vzdáleností dvojice slov je pak vytvořen jako kopie jejich histogramu, který je aktualizován dle 3.21, místa jsou pak ukládána do seznamu (který tvoří topologickou mapu). Každé místo pak obsahuje hash tabulku podobnou té, která představuje model prostředí, avšak pouze pro dvojice slov, vyskytujících se v daném snímku.

4.2.3 Vyhodnocení aktuální lokace

Vstupem této části algoritmu je opět struktura reprezentující snímek popsaná v oddíle 4.2.1, výstupem pak seznam pravděpodobností, že vstupní snímek pochází ze stejné lokace jako snímky v topologické mapě (případně, že pochází z neznámé lokace). Samotný proces porovnání pak znamená, že ze vstupního snímku jsou opět vygenerovány seznamy dvojic slov a vzdáleností mezi nimi, nad každým seznamem vzdáleností je vytvořen histogram (zde pouze za pomoci modelu senzoru a každá pozorovaná vzdálenost tak odpovídá posunuté Gaussově funkci). Tento histogram je pak porovnán (dle 3.18) s odpovídajícími histogramy všech míst v topologické mapě. Při tomto výpočtu je již nutné pracovat s logaritmickou pravděpodobností (tzv. “log-likelihood”), protože zde hrozí podtečení (jak pro typ “float”, tak i “double”).

Pro vyhodnocení pravděpodobnosti je zde (na rozdíl od použité implementace algoritmu *FABMAP*) využita metoda průměrné lokace a to tak, že jedna z porovnávaných lokací odpovídá právě globálnímu modelu prostředí. U ostatních lokací v mapě se histogram vzdáleností hledá nejdříve v jejich modelu a až následně (není-li nalezen) v globálním modelu prostředí.

Posledním krokem je normalizace pravděpodobností přes všechny známé lokace, která probíhá stále v logaritmické pravděpodobnosti. Výsledkem je seznam pravděpodobností pro všechna známá místa se sumou jedna.

4.3 Urychlení algoritmu pro shlukování deskriptorů

Algoritmus shlukování je v původní implementaci lokalizační metody *FABMAP* (*OpenFABMAP*) jednou z časově nejnáročnějších operací. Důvod je při letmém pohledu na algoritmus 1 zřejmý, jeho složitost roste s počtem trénovacích vzorků (N) a počtem nalezených středů (k) jako $O(\sum_{i=1}^{N}\lfloor 1+\alpha i \rfloor + \alpha N^2 + N)$, kde $\alpha = k/N$. Pokud tedy chceme zachovat rozumnou škálovatelnost, nabízí se dvě řešení: buď použití shlukovacího algoritmu s lepší časovou složitostí (např. *BIRCH* dosahuje složitosti $O(n*log(k))$), nebo paralelizovat původní algoritmus. Výhodou druhé možnosti je, že odpadá nutnost ladění zbylé části řešení, které může být nutné při změně shlukovacího algoritmu. Z tohoto důvodu bylo zvoleno druhé ze zmíněných řešení, algoritmus byl paralelizován a výpočet přenesen na *GPU* (za předpokladu bezchybné implementace je tedy zaručeno nezměněné chování algoritmu).

4.3.1 OpenCL a architektura GPU

Platforma *OpenCL* je průmyslovým standardem v oblasti heterogenních výpočetních platforem, jejím cílem je unifikace výpočetních zdrojů. Nejrozšířenější využití nachází jako platforma umožňující využití grafických akcelerátorů pro obecné výpočty (*GPGPU* – General Purpose GPU). Ovšem implementace nejsou omezeny pouze na *GPU*, rozšířené jsou také implementace pro více-jádrová *CPU* a existují také implementace pro *FPGA*. Platforma *OpenCL* je tvořena jednak samotnou multiplatformní knihovnou a stejnojmenným programovacím jazykem, který vychází z jazyka *C99* (v posledních verzích jsou do něj však přidávány některé konstrukce *C++*). Obvyklou oblastí využití platformy je především zpracování datově paralelních úloh, avšak nechybí podpora pro úlohově paralelní programování.

Struktura platformy OpenCL

Architekturu platformy *OpenCL* je možné popsat z několika pohledů, kde každý z nich zachycuje část jejích vlastností. V tomto oddíle budou popsány základní pohledy (nebo modely), které jsou: strukturální, exekuční a paměťový. Cílem je také zachytit jejich význam ve vztahu k řešenému problému.

Základním pohledem je model platformy (strukturální model). Ten se skládá z hosta (obvykle *CPU*), ke kterému může být připojeno jedno, nebo více *OpenCL* zařízení (nazývané *OpenCL* device). Zařízení se dále dělí na výpočetní jednotky (Compute units - CUs) a tyto jsou dále děleny na výkonné elementy (Processing elements - PEs). Tyto pak vykonávají samotný výpočet nad daty. Výkonné elementy, tvořící jednu výpočetní jednotku, pak mohou vykonávat program jako *SIMD* jednotky (v daném čase zpracovávají všechny elementy stejnou instrukci programu), nebo *SPMD*, kde všechny jednotky vykonávají stejný program, avšak nemusí provádět stejnou instrukci. V obou případech jednotky mohou pracovat s různými daty.

Modelem popisujícím *OpenCL* z pohledu vykonávání programu je model exekuční, ten je rozdělen do dvou částí: jádra (kernels) a program hosta. Kód tvořící kernel je vykonáván na straně *OpenCL* zařízení. Program hosta je pak klasický program vykonávaný na *CPU*, který volá knihovní funkce *OpenCL* a takto komunikuje se zařízením vykonávajícím kód kernelů. Kernely jsou odesílány ke zpracování na zařízení tak, že je specifikován jejich kód (kernel v podobě funkce bez vedlejších efektů) a prostor indexů (index space), nad kterými má být kernel vykonán. Prostor indexů je 1 až 3 rozměrný kartézský prostor nad celočíselnými indexy, to umožňuje zjednodušení adresování v případě práce nad např. dvourozměrnými

daty, jako jsou obrázky. Na *OpenCL* zařízení je následně vytvořena instance kernelu (work-item nebo vlákno) pro každý bod tohoto prostoru. Všechny instance provádějí stejný kód, avšak díky větvení kódu mohou vykonávat jeho různé části a samozřejmě také mohou pracovat s různými daty. Instance jsou dále sdružovány do skupin (work-groups). Vlákna tvořící jednu skupinu jsou prováděna paralelně pomocí výkonných elementů jedné výpočetní jednotky.

Posledním důležitým modelem je paměťový model, který definuje několik paměťových prostorů, z nichž k některým má přístup pouze kernel, k jiným pouze host a některé jsou sdílené. Platforma *OpenCL* definuje tyto prostory:

- **Globální (Global Memory)**: Do této paměti mají přístup všechny instance kernelu, mohou z ní jak číst, tak do ní i zapisovat. Host do této paměti zařízení přistupuje prostřednictvím paměťových objektů (buffers), které označují části této paměti, a to obvykle pomocí kopírování těchto oblastí do (případně z) paměti hosta.
- **Konstantní (Constant Memory)**: Tento paměťový prostor má z pohledu hosta stejné vlastnosti jako globální. Instance kernelu však nemohou do této paměti zapisovat, ale pouze z ní číst.
- **Lokální (Local Memory)**: Jde o paměť sdílenou mezi vlákny jedné skupiny (work-group), ty ji mohou jak číst, tak do ní zapisovat. Pro hosta je však nedostupná.
- **Privátní (Private Memory)**: Je využívána jako paměť proměnných jednoho vlákna (work-item). Ostatní vlákna, ani host do ní nemají přístup.

Krom těchto prostorů je zde ještě paměť hosta, do které však nelze z kernelů přistupovat. Toto rozdělení paměti na globální paměť zařízení a paměť hosta je značným omezením oblasti využití *OpenCL*, protože vyžaduje kopírování dat mezi akcelerátorem a pamětí hosta. Jsou zde proto snahy toto kopírování eliminovat a docílit sdíleného adresového prostoru mezi zařízením a hostem.

Architektura GPU

Počátky architektur sahají do doby, kdy tyto jednotky byly tvořeny jednoúčelovým hardwarem schopným vykreslování textu, později 2D grafických primitiv bez účasti *CPU*. Postupem času byla však jejich funkcionalita rozšiřována o další specializované jednotky, jejichž úkolem bylo např. vykreslování 3D grafiky (stínování, transformace vrcholů, ořezávání atd.). Tyto jednotky byly řízeny pomocí parametrů a jejich univerzálnost tedy byla velmi omezená. S přibývajícími požadavky na jejich schopnosti však došlo k jejich nahrazení programovatelnými jednotkami a *GPU* tak jsou do dnešní doby hybridními procesory, které obsahují jak specializované, tak programovatelné jednotky. První programovatelné jednotky byly stále uzpůsobeny především pro zpracování 3D grafických dat a proto byly rozděleny na jednotky pracující nad vrcholy ("Vertex Shaders") a pixely ("Pixel Shaders").

V dnešní době bylo toto rozdělení překonáno a *GPU* jsou tak tvořeny především značným množstvím univerzálních výpočetních jednotek, které mohou být uspořádány do několika *VLIW* ("Very long instruction word"), nebo *SIMD* (dnes téměř výhradně používané uspořádání) procesorů. Každý procesor obsahuje krom výpočetních jednotek také značný počet registrů (např. 64KB registrů a 16 *ALU*). Program je pak pomocí *SIMD* prováděn jako množina vláken, která jsou prováděna po skupinách ("wavefront"). Skupiny vláken jsou obvykle větší, než samotná šířka *SIMD* a provedení jedné instrukce tak trvá několik

strojových cyklů (např. 64 vláken, tedy 4 cykly). Vlákna uvnitř skupiny musí vždy provádět stejnou instrukci programu, jinak musí být skupina rozdělena a část jednotek je nečinná - dochází k divergenci vláken ("Thread divergence").

Jednotky *SIMD* jsou dále spojovány do základních výpočetních bloků ("Compute Units"), které sdružují několik *SIMD* a také obsahují jednotky pro dekódování instrukcí, podporu skoků a přístupy do paměti, lokální paměť ("Local Memory") a první úroveň vyrovnávací paměti ("L1 cache"). Výpočetní blok obsahující 4 *SIMD* může v jednom okamžiku provést 64 *ALU* operací, každý *SIMD* provádí jednu instrukci ze stejné skupiny vláken. Pro skrytí latencí (především přístupů do paměti) může výpočetní blok mít rozpracováno až 40 skupin vláken a každý *SIMD* si může vybrat jednu z 10 skupin jejíž instrukci v následujících 4 cyklech provede. Poznamenejme, že se jedná o architekturu, která neumožňuje provádění instrukcí mimo pořadí a toto je tedy jediný způsob skrytí latencí při čekání na data či provádění skoků.

Lokální paměť výpočetního bloku má zvláštní význam, umožňuje sdílení dat mezi vlákny, která jsou prováděna pomocí jednoho výpočetního bloku (a navíc jsou zařazena do stejné pracovní skupiny). Také umožňuje (v ideálním případě) přístup ze všech vláken v jednom cyklu ke stejné adrese, nebo do různých banků. Obvykle má tato paměť velikost několika desítek kilobyte (např. 64 KB) a je rozdělena mezi pracovní skupiny běžící na daném výpočetním bloku. Tato paměť také umožňuje provádění atomických operací. Celé *GPU* je pak tvořeno několika výpočetními bloky, pamětí L2 cache, jednotkami pro rozdělování práce mezi výpočetní bloky a paměťovými kontroléry.

Algoritmus shlukování byl primárně navržen pro *GPU* s kódovým označením "Tahiti" firmy AMD. Toto *GPU* obsahuje 2048 jednotek *ALU* ve 32 výpočetních blocích a při taktu 1 GHz na kterém pracuje dosahuje výpočetního výkonu 4,096 TFLOPS (uvažujeme-li operaci spojeného násobení a přičtení v jednoduché přesnosti). Paměťová sběrnice tohoto *GPU* má šířku 384 bitů a při efektivním taktu 5500 MHz dosahuje propustnosti 264 GB/s. Opět je zřejmá (v této oblasti obvyklá) disproporce mezi paměťovou propustností a výpočetním výkonem a shlukovací algoritmus bude zřejmě limitován právě tímto parametrem.

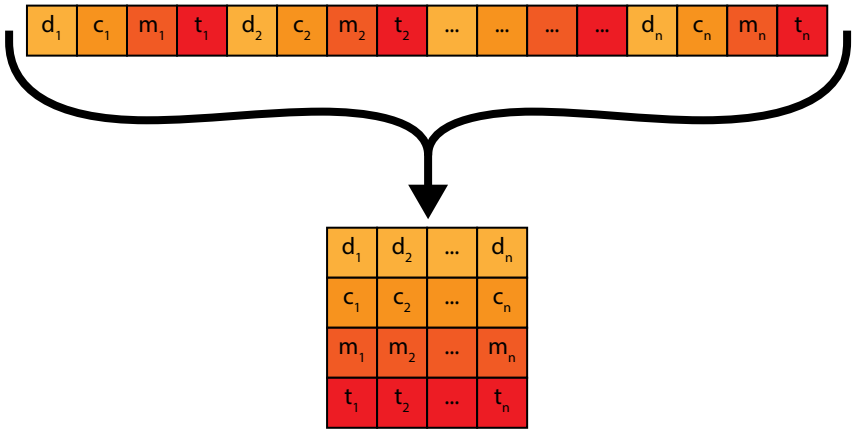
4.3.2 Paralelizace algoritmu na GPU

Paralelní implementace algoritmu 1 je rozdělena do tří kroků: prvním krokem je hledání středů shluků, druhým pak přiřazení všech vzorků do shluků a posledním krokem je na základě přiřazených vzorků výpočet nového středu shluku. První a druhý krok jsou implementovány paralelně, třetí je pak řešen sekvenčně. Toto uspořádání je postačující, jelikož první a druhý krok mají oba téměř kvadratickou složitost $O(N^2)$ vzhledem k počtu shlukovaných vzorků, třetí pak má již pouze lineární složitost $O(N)$.

Pro potřeby shlukování je každý vstupní vzorek reprezentován čtveřicí $(d_i, c_i, d_{\min_i}, t_i)$, kde $d_i \in \mathbb{R}^{64}$ je reálný deskriptor, $1 \leq c_i \leq k$ je index shluku, do kterého vzorek patří, $d_{\min_i} \in \mathbb{R}^+$ je vzdálenost od středu shluku c_i a $t_i \in \{0, 1\}$ je binární proměnná, kde $t_i = 1$ pokud je daný vzorek přiřazen nějakému shluku. Pro zajištění optimálního přístupu do paměti je tato čtveřice uložena do čtyř polí (na místo struktury). Z důvodu stále omezené velikosti dostupné paměti na *GPU* jsou vstupní vzorky následně zpracovávány po blocích o B vzorcích.

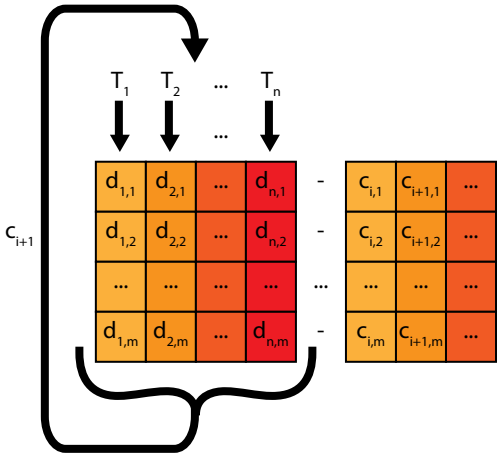
Zpracování každého bloku v prvním kroku algoritmu začíná jeho zapsáním do paměti *GPU*, při této operaci je pomyslná matice vzorků d_i, kde každý vzorek tvořil jeden souvislý blok 64 hodnot (řádek matice), transponována. Souvislý blok délky B tak nyní tvoří první složky všech vzorků, což opět umožňuje souvislý přístup do paměti při výpočtu vzdálenosti

mezi vzorkem a středy shluků (viz. obrázek 4.2). Před samotným hledáním nových shluků v bloku je třeba provést přiřazení vzorků v bloku k již existujícím shlukům a zde se dostává ke slovu paralelní zpracování.



Obrázek 4.2: Transformace dat pro shlukování na *GPU* (m_i zde odpovídá $d_{\min_i}$).

Obrázek 4.3 ukazuje způsob výpočtu vzdáleností mezi středem c_i a blokem vzorků paralelně na *GPU*, každé vlákno tak postupně po jednotlivých složkách vzorku d_j vypočte jeho vzdálenost ke středu c_i, následně se přesouvá k dalšímu středu c_{i+1}. Zároveň si vlákno uchovává minimální vzdálenost a index středu ke kterému se vztahovala. Nakonec uloží novou minimální vzdálenost, index středu a v případě, že minimální vzdálenost je menší než maximální velikost shluku nastaví také bit $t_j = 1$. Porovnávání se středy také probíhá po blocích (podobně jako v případě vzorků), v jednom běhu jsou tedy všechny vzorky v bloku porovnány se středy $c_{m(k-1)}$ až c_{mk}, díky tomu je možné pro uložení středů c_i využít paměť konstant *GPU*, která umožňuje vícenásobné čtení v jenom taktu, ale má omezenou velikost.



Obrázek 4.3: Paralelní Výpočet vzdáleností mezi středy shluků (c_i) a vzorky.

Proces hledání nových shluků je pak založen na přiřazení vzorků do shluků a jejich ozna-

čení (viz. předchozí odstavec), ale po každém průchodu bloku středů je navíc (již sekvenčně) prohledáván právě zpracovávaný blok vzorků (přičemž lze přeskočit všechny vzorky označené $t_i = 1$) a do bloku středů pro další průchod jsou přidávány dosud neoznačené vzorky. Přičemž pro každý nově přidávaný střed musí platit, že jeho vzdálenost od ostatních v právě sestavovaném bloku je větší než maximální velikost shluku. Následující krok přiřazovaní pak vzorky porovnává se středy v tomto nově vytvořeném bloku středů.

Následně je již nutné pouze provést opětovné přiřazení vzorků ze všech bloků do shluků (jak je popsáno výše) s tím rozdílem, že již máme k dispozici kompletní množinu středů ze všech bloků vzorků. Po tomto finálním přiřazení je třeba již pouze na jeho základě vypočítat nové středy shluků (jako v původním algoritmu).

Kapitola 5

Vlastnosti výsledného řešení

V této kapitole bude popsáno testování implementace navržené lokalizační metody, jejímž cílem je vyhodnotit její vlastnosti v oblasti citlivosti, odolnosti vůči vlivu problematických částí prostředí (jako jsou opakující se textury, struktury a architektonické prvky) a vlivu orientace kamery vůči směru jejího pohybu. Krom samotné metody lokalizace jsou uvedeny také testy implementované metody shlukování deskriptorů (a to především výkonnostní testy, jelikož funkčně je implementace identická s její původní verzí).

Jelikož je testované řešení implementace topologické metody lokalizace, navíc založené pouze na pozorování scén, není nutné v testech zohledňovat metrickou mapu s rozložením míst, naopak je třeba vzít v úvahu orientaci kamery (pohledu). Připomeňme, že dva dostatečně různé pohledy z jednoho místa v metrické mapě mohou tvořit různé lokace v mapě topologické. Testovací sady jsou proto založeny na porovnávání scén, navíc není použit pohybový model a výsledky lokalizace jednotlivých pohledů (snímků) jsou tedy vzájemně nezávislé. Pozice robota je dána jako $L_c = \arg\max_{L_i \in M \cup \{L_u\}} p(L_i|D_c)p(L_i|Z_c)$, kde L_c je odhad aktuální lokace, M je množina známých míst, L_u představuje neznámé místo a (Z_c, D_c) je aktuální pozorování.

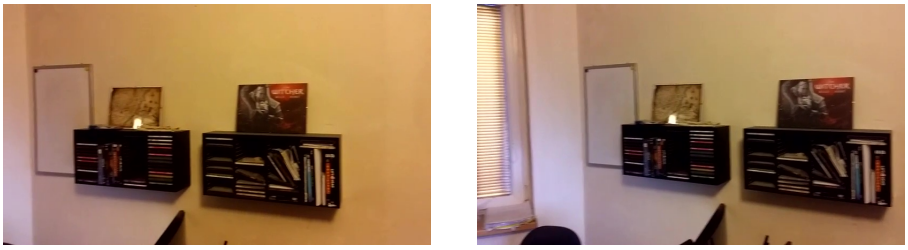
Samotný algoritmus je pak chápán jako klasifikátor s mnoha třídami, kde každá třída tvoří jednu lokaci v topologické mapě. Dostáváme tak matici pravděpodobností $\mathbf{C}$ ("Confusion Matrix"), že testovací snímek I_i pochází ze stejné scény jako učící snímek I_j, tedy $\mathbf{C}_{j,i} = p(L_j|D_i)p(L_j|Z_i)$, kde (Z_i, D_i) je pozorování na snímku I_i. První řádek matice $\mathbf{C}_{1,j} = p(L_u|D_i)p(L_u|Z_i)$, kde $L_u \in \overline{M}$ představuje neznámé místo a první sloupec $\mathbf{C}_{i,1} = 0$. Pro ideální klasifikaci (zde lokalizaci) by tedy platilo, že $\mathbf{C}_{i,i} = 1$ pro $i > 1$ a $\mathbf{C}_{i,j} = 0$ pro $i \neq j$.

5.1 Použité datové sady

Pro vyhodnocení dosažených výsledků algoritmu pak bylo zvoleno několik různých typů datových sad. Prvním typem jsou syntetické datové sady, které jsou tvořené průchodem prostředí z počítačové hry. Důvodem k jejich použití je častý výskyt opakujících se textur a snadná změna parametrů testu (jako změna trasy, prostředí atd.). Výhodou je také možnost přesného určení metrické pozice a orientace kamery, navíc nedochází k problémům s kvalitou snímků (rozostření, přesvětlení či rozmazání pohybem) a jsou omezeny vnější vlivy, jako je změna světelných podmínek. Na druhé straně kvůli těmto vlastnostem je jejich vypovídající hodnota o některých charakteristikách řešení omezená (nebo dokonce žádná). Z toho důvodu jsou také použity datové sady z reálného prostředí a to jak z vnějších, tak

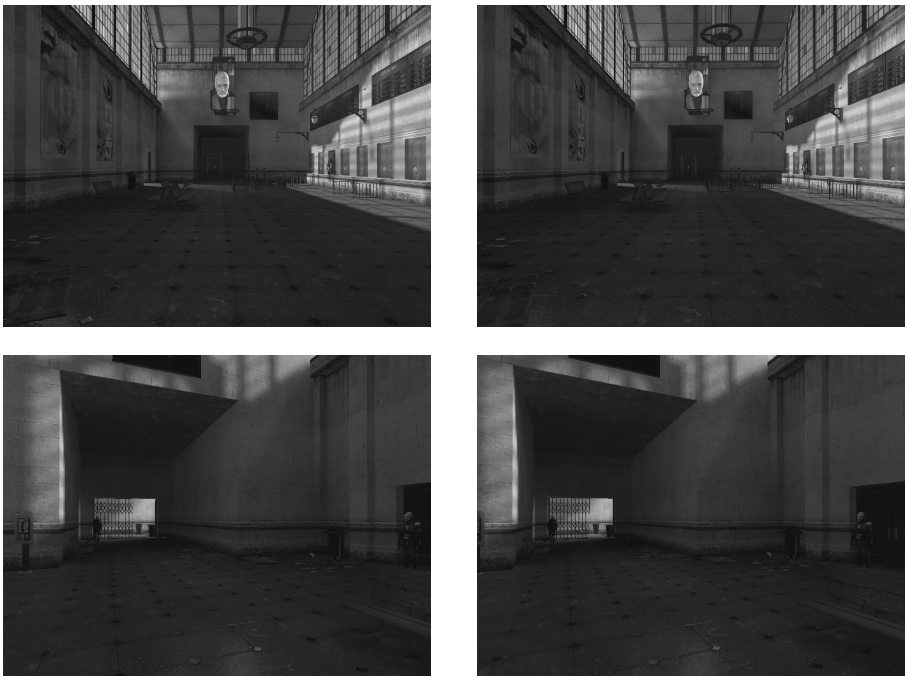
vnitřních prostor. Většina testovacích snímků je v rozlišení 800x600 pixelů, nebo nižší.

Datová sada "Room" Je datovou sadou z reálného vnitřního prostředí, která byla pořízena mobilním telefonem a tvoří ji 2x30 snímků. Tyto snímky jsou pořízeny z jednoho místa v průběhu dvou otáček kolem svislé osy (tedy zhruba jeden snímek každých 12 stupňů). Obrázek 5.1 ukazuje minimální rozdíl mezi dvojicí snímků.



Obrázek 5.1: Snímky z datové sady "Room"

Datová sada "Half-Life 2" Tato datová sada byla pořízena z počítačové hry stejného názvu s frekvencí 16 Hz. Její první část tvoří stejný test, jako v případě datové sady "Room", druhá pak testuje posun ve směru pohledu. Na obrázku 5.2 jsou z obou těchto testů uvedeny dvojice snímků s typickým rozdílem v pozici či rotaci. Tuto datovou sadu tvoří 2x160 a 2x120 snímků.



Obrázek 5.2: Snímky z datové sady "Half-Life 2"

Datová sada "New College" Je uzavřenou smyčkou z vnitřního prostředí, pocházející z větší datové sady používané k testování původního algoritmu *FABMAP*. Je pořízena pojízdným robotem s periodou 1.5 sekundy a snímky jsou pořizovány po dvojicích kamerou

natočenou vlevo a vpravo od směru pohybu robota. Testovací smyčku tvoří 2x185 po sobě jdoucích snímků, na obrázku 5.3 jsou uvedeny příklady levého a pravého snímku.

Obrázek 5.3: Snímky z datové sady "New College"

Datová sada "City Centre" Je obměnou datové sady "New College", ale je pořízena z vnějšího městského prostředí. Způsob záznamu zůstal nezměněn a datovou sadu tvoří 2x85 snímků ze dvou průjezdů robota stejnou cestou. Příklad snímků z této datové sady je na obrázku 5.4.

Obrázek 5.4: Snímky z datové sady "City Centre"

5.2 Testy citlivosti

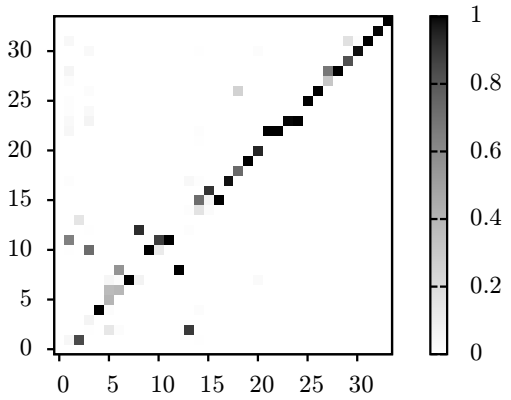
Cílem testů citlivosti je vyhodnocení činnosti algoritmu v případě minimálních změn v pozici či orientaci kamery a tedy dosažitelné přesnosti lokalizace. Citlivost je testována na třech datových sadách, kde první je z reálného vnitřního prostředí (sekvence zaznamenaná kamerou mobilního telefonu) a zbylé dvě jsou syntetické datové sady tvořené záznamem z počítačové hry. Na těchto datových sadách je testována citlivost ke změně pohledu (rotace kamery kolem svislé osy) a citlivost ke změně polohy (zde pohyb ve směru pohledu). Výsledky těchto testů jsou zapsány v tabulce 5.1 (kde TPR - "True Positive Rate" rozumíme procento správně lokalizovaných snímků a FPR - "False Positive Rate" chybně zařazených).

	Rotace reál. data		Rotace synt. data		Posun synt. data	
	TP	FP	TP	FP	TP	FP
Navržená metoda	78.79%	21.21%	85.83%	14.17%	90.42%	9.58%
Algoritmus *FABMAP*	57.58%	42.42%	87.5%	12.5%	89.22%	10.78%
Prostorový model	54.55%	45.45%	80.83%	19.17%	72.46%	27.54%

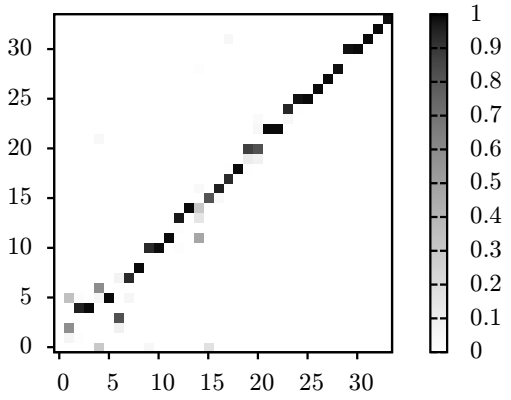
Tabulka 5.1: Tabulka výsledků pro datové sady

První datovou sadu tedy tvoří 60 snímků, které byly pořízeny v průběhu dvou otáček (tedy 30 snímků na otáčku s rozestupy asi 12 stupňů). Pravděpodobnosti korespondence mezi snímky pro tento pokus zachycují matice na obrázku 5.5. V tomto reálném testu lze pozorovat znatelný přínos navržené metody, který je způsoben relativně nízkým počtem stabilně detekovaných klíčových bodů v této datové sadě (tím jsou získány stabilní prostorové vztahy, konzistentní mezi průchody).

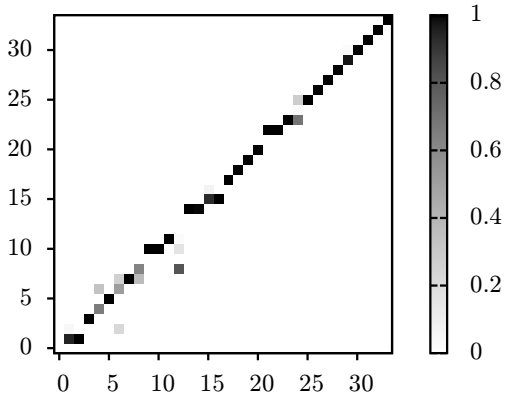
Druhá a třetí datová sada patří do skupiny syntetických a jedná se o záznamy ze známé počítačové hry. Příklady snímků z těchto datových sad jsou na obrázku **??**. Druhá sada opět testuje citlivost na rotaci, v tomto případě ale eliminuje vertikální nepřesnosti a umožňuje porovnat přínos navržené metody mezi reálnými a ideálními podmínkami. Výsledky tohoto testu zachycuje obrázek 5.6. Poslední datová sada zaměřující se na citlivost testuje přesnost v posuvu ve směru pohledu a její výsledky zachycuje obrázek 5.7. Za zmínku stojí, že v případě syntetických datových sad je přínos prostorového modelu minimální, což může být způsobeno opakujícími se texturami a tím porušení konzistence modelu vzdáleností mezi dvojicí snímků. Druhou možností je, že samotný výskytový model zde dosahuje výjimečně dobrých výsledků a přínos druhého modelu se neprojeví.



(a) Prostorový model

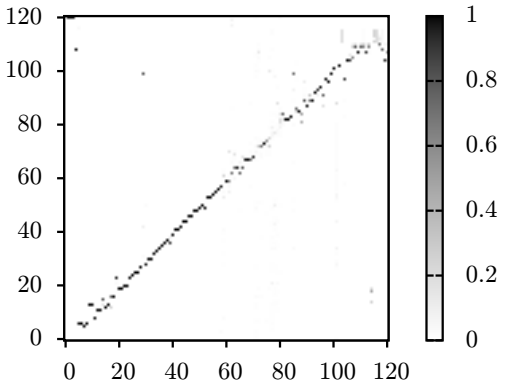


(b) Algoritmus *FABMAP*

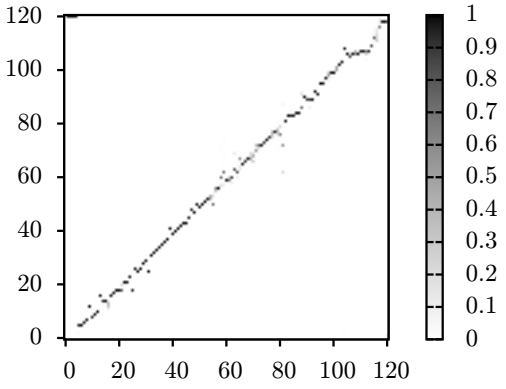


(c) Navržená metoda

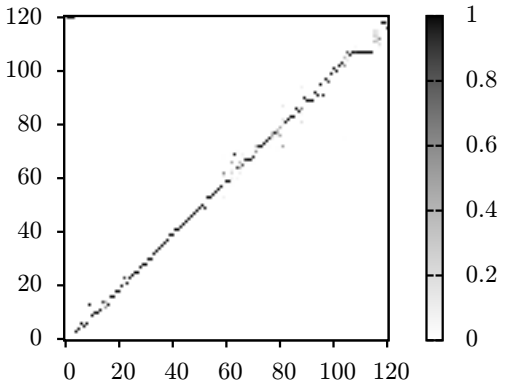
Obrázek 5.5: Výsledky testu rotace kamery kolem svislé osy ve vnitřním prostředí



(a) Prostorový model

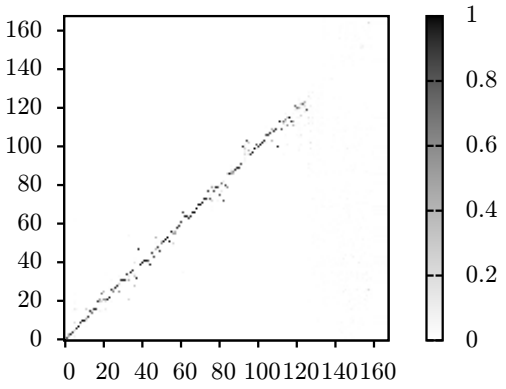


(b) Algoritmus *FABMAP*

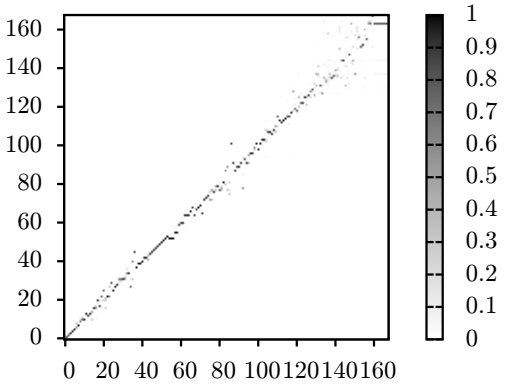


(c) Navržená metoda

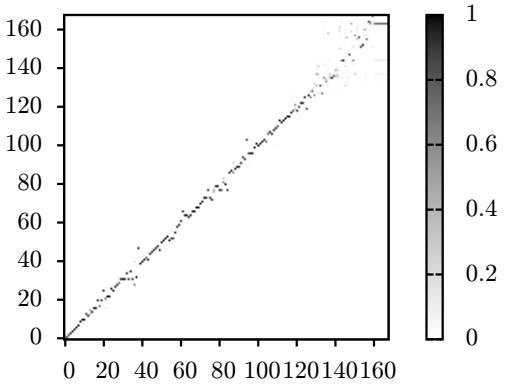
Obrázek 5.6: Výsledky testu citlivosti k rotaci v syntetickém prostředí



(a) Prostorový model



(b) Algoritmus *FABMAP*



(c) Navržená metoda

Obrázek 5.7: Výsledky testu citlivosti k posuvu v syntetickém prostředí hry

5.3 Testy lokalizace

Testy lokalizace jsou opět reprezentovány pomocí korespondence snímků, rozdílem je použití větších datových sad a jejich cílem není rozlišování velmi podobných pohledů, ale naopak hledání podobných dvojic ve větším počtu snímků. Data jsou opět tvořena jako dvojice průchodů shodnou oblastí.

Pro tento test byla zvolena podmnožina snímků z datové sady "New College" [8]. Výsledky na této datové sadě jsou zachyceny obrázkem 5.10 a tabulkou 5.2. Ve výsledných maticích je možné si povšimnout rozptýlení hodnot (asi od 125 snímku), což je způsobeno tím, že testovací snímky sice pocházejí z téměř stejné lokace, ale jsou vůči sobě posunuty a také zaznamenány pod rozdílným úhlem pohledu (obrázek 5.8).

Druhým testovaným prostředím bylo vnější městské prostředí tvořené vzorkem jedné ze smyček datové sady "City Centre" používané v rámci *FABMAP*. Opět se jedná o synchronizaci dvojice nezávislých průchodů dané oblasti, kde však jeden z průchodů byl pomalejší a matice pravděpodobností není diagonální. Vyčíslení hodnot TP a FP by tak vyžadovalo přesnou masku a proto nejsou uvedeny, avšak přímo z matice pravděpodobností (obrázek 5.11) lze usoudit, že došlo k mírnému zlepšení lokalizace (menší rozptýlení).

	TP	FP
Navržená metoda	67.58%	32.42%
Algoritmus *FABMAP*	51.1%	48.9%
Prostorový model	58.79%	41.21%

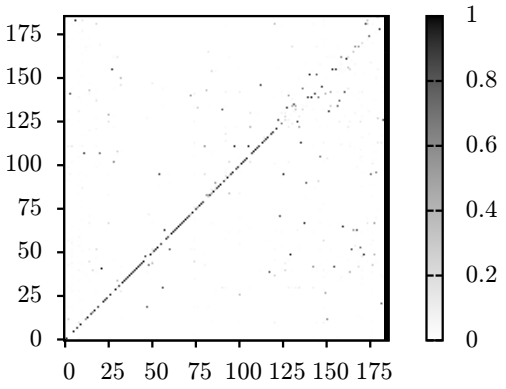
Tabulka 5.2: Tabulka výsledků pro test nad datovou sadou "New College"

(a) Průchod 1 (b) Průchod 2

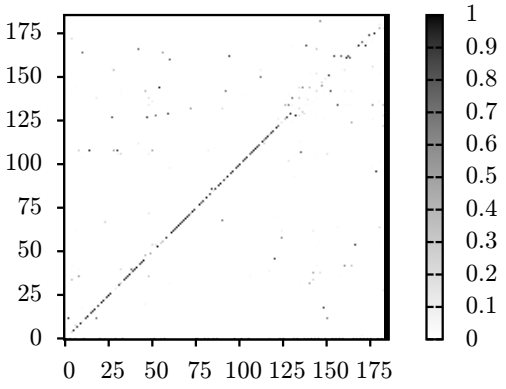
Obrázek 5.8: Posun v datové sadě "New College"

(a) Příklad 1 (b) Příklad 2

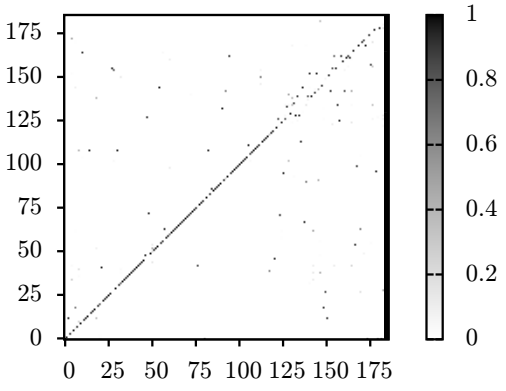
Obrázek 5.9: Příklad snímků ze smyčky v datové sadě "City Centre"



(a) Prostorový model

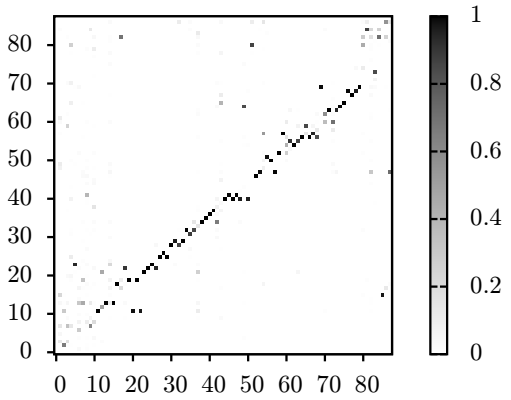


(b) Algoritmus *FABMAP*

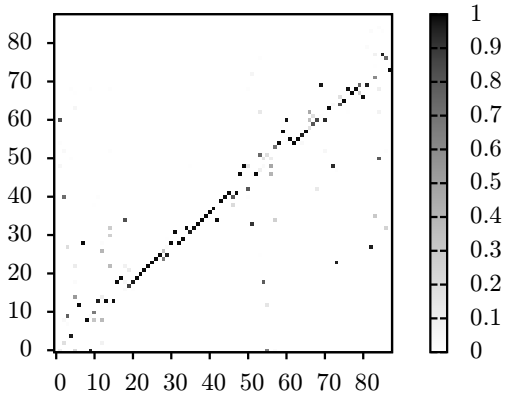


(c) Navržená metoda

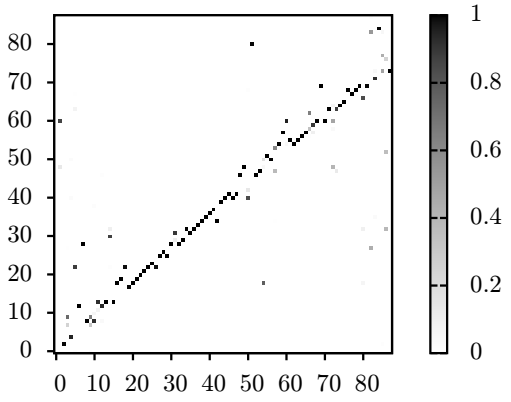
Obrázek 5.10: Výsledky testu lokalizace na podmnožině datové sady “New College”



(a) Prostorový model



(b) Algoritmus *FABMAP*



(c) Navržená metoda

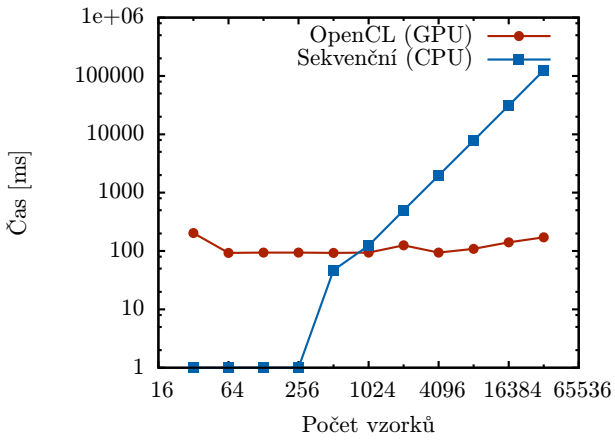
Obrázek 5.11: Výsledky testu lokalizace na podmnožině datové sady “City Centre”

5.4 Výkonnostní charakteristiky algoritmu

V tomto oddíle bude provedena stručná analýza časové a prostorové složitosti algoritmů implementovaných v rámci zde prezentovaných algoritmů. Je zřejmé, že algoritmus lokalizace musí být schopný dosáhnout rozumné frekvence aktualizace polohy v dostatečně rozsáhlé mapě. Z tohoto důvodu je nutné při návrhu takto použitelných algoritmů dbát na jejich složitost a škálovatelnost.

5.4.1 Algoritmus shlukování

Ačkoliv tato část řešení není využívána při samotné lokalizaci, či tvorbě topologické mapy, může se snadno stát problémovým místem (především při vývoji a testování zbylých částí řešení). Z toho důvodu byl v rámci této práce navržen paralelní shlukovací algoritmus (implementovaný na *GPU*), který je ekvivalentní původnímu sekvenčnímu řešení. Na obrázku 5.12 je zachyceno dosažené zrychlení v závislosti na počtu vstupních vzorků. Posun charakteristiky sekvenčního řešení je způsoben počtem shluků (který je zde $\sqrt{N}$) a tím, že jeho složitost závisí na rozložení nalezených středů shluků ve vstupu (složitost sekvenčního algoritmu je uvedena v oddíle 4.3).



Obrázek 5.12: Dosažené zrychlení paralelního shlukovacího algoritmu na GPU

5.4.2 Algoritmus lokalizace

Měření rychlosti algoritmu lokalizace nebylo prováděno z toho důvodu, že řešení prostorového modelu je neoptimální a má kvadratickou složitost vzhledem k počtu klíčových bodů v každém snímku (několik řešení je navrženo v závěru). Celkově pak učení lokalizačního algoritmu má kvadratickou složitost vzhledem k velikosti slovníku (tvorba *Chow Liu Tree*, vyžaduje výpočet vzájemné informace mezi všemi dvojicemi slov), tedy $O(|V|^2 N)$, kde $|V|$ je velikost slovníku a N je počet trénovacích snímků. Učení prostorového modelu má pak také kvadratickou složitost, avšak vzhledem k počtu různých slov ve snímku, tedy $O(\alpha^2 N)$, kde α je průměrný počet slov ve snímku v trénovací sadě a N je opět počet snímků v sadě. Na rozdíl od výskytového má však prostorový model také kvadratické nároky na prostor pro jeho uložení (reálně však bývá nižší, jelikož ne každá dvojice slov se vyskytuje v nějakém snímku).

Učení topologické mapy má pak v případě výskytového modelu konstantní časovou

složitost, prostorový model má i v této úloze složitost kvadratickou vzhledem k výskytu slov v přidávaném snímku (stejně tak prostorovou).

Předchozí časové složitosti obou algoritmů jsou však amortizované (jsou prováděny pouze jednou). Hlavní nevýhodou prostorového modelu je tak kvadratická časová složitost při samotné lokalizaci (opět vzhledem k počtu slov ve snímku). Výskytový model má pak složitost lineární k počtu známých míst. Řešení problému složitosti prostorového modelu jsou diskutována v závěru.

Kapitola 6

Závěr

V této práci byla prezentována metoda lokalizace v topologickém prostoru založená na algoritmu *FABMAP* hlavním přínosem práce je rozšíření této metody o model zachycující prostorovou strukturu rozložení klíčových bodů v obraze, což přináší drobné zlepšení lokalizace především v oblasti její přesnosti, kde může mít výskytový model nedostatečnou sílu (nedojde-li ke změně lokálních deskriptorů mezi dvěma snímky, jsou nerozlišitelné).

Na druhé straně tento prostorový model přináší značnou výpočetní zátěž, jelikož je založen na vztazích mezi klíčovými body (z toho vyplývá kvadratická složitost v prostoru i čase). Zde je jedna z oblastí možného rozšíření a dalšího vývoje tohoto modelu, kterou je omezení zachycených vztahů. Takové rozšíření může být založeno na rozložení klíčových bodů ve snímcích, kde se často vyskytují shluky (generované objekty), tyto shluky by bylo možné v modelu považovat za jediný objekt a vztahy v obraze řešit na této úrovni, případně definovat hierarchickou strukturu a prostorové vztahy tak řešit na různých úrovních (opět dochází k omezení jejich celkového počtu). Jinou možností je založit výběr uchovávaných vztahů mezi dvojicemi bodů na jejich společném výskytu ve snímcích v trénovací sadě (obdoba omezení závislostí kterou využívá *FABMAP* nad slovy ve slovníku).

V rámci práce rovněž došlo k paralelizaci algoritmu pro tvorbu slovníku, a to konkrétně části týkající se shlukování. Stávající verze algoritmu pro shlukování byla nahrazena její ekvivalentní paralelní variantou běžící na *GPU* (využiti technologie *OpenCL*). Tím došlo přibližně k 1000 násobnému urychlení výpočtu v případě tvorby slovníků z většího počtu vzorků.

Další možností, která má potenciál na zlepšení vlastností navrženého řešení je užší propojení se stávajícím algoritmem *FABMAP*, kdy by se (zjevně podobné) pravděpodobnostní modely propojily již na úrovni výpočtu pravděpodobnosti nad jednou dvojicí slov ve snímcích. Tento přístup využívá algoritmus *FABMAP 3D* [19], který však pracuje s reálným trojrozměrným prostorem.

Literatura

[1] Agrawal, M.; Konolige, K.; Blas, M. R.: CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching. In *ECCV (4)*, *Lecture Notes in Computer Science*, ročník 5305, editace D. A. Forsyth; P. H. S. Torr; A. Zisserman, Springer, 2008, ISBN 978-3-540-88692-1, s. 102–115.

[2] Alahi, A.; Ortiz, R.; Vandergheynst, P.: FREAK: Fast Retina Keypoint. In *CVPR*, IEEE Computer Society, 2012, ISBN 978-1-4673-1226-4, s. 510–517.

[3] Bach, F. R.; Jordan, M. I.: Thin Junction Trees. In *NIPS*, editace T. G. Dietterich; S. Becker; Z. Ghahramani, MIT Press, 2001, s. 569–576.

[4] Bay, H.; Tuytelaars, T.; Gool, L. V.: Surf: Speeded up robust features. In *In ECCV*, 2006, s. 404–417.

[5] Campbell, J.; Sukthankar, R.; Nourbakhsh, I.; aj.: A robust visual odometry and precipice detection system using consumer-grade monocular vision. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, IEEE, 2005, s. 3421–3427.

[6] Chow, C. K.; Liu, C. N.: Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, ročník 14, č. 3, 1968: s. 462–467.

[7] Cummins, M.; Newman, P. M.: Probabilistic Appearance Based Navigation and Loop Closing. In *ICRA*, IEEE, 2007, s. 2042–2048.

[8] Cummins, M. J.; Newman, P. M.: FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *I. J. Robotic Res.*, ročník 27, č. 6, 2008: s. 647–665. URL http://www.robots.ox.ac.uk/~mobile/IJRR_2008_Dataset/data.html

[9] Cussens, J.: Bayes and Pseudo-Bayes Estimates of Conditional Probabilities and Their Reliability. In *ECML*, *Lecture Notes in Computer Science*, ročník 667, editace P. Brazdil, Springer, 1993, ISBN 3-540-56602-3, s. 136–152.

[10] Davison, A. J.; Reid, I. D.; Molton, N. D.; aj.: MonoSLAM: Real-Time Single Camera SLAM. *IEEE TPAMI*, ročník 29, č. 6, 2007: s. 1052–1067, ISSN 0162-8828, doi:10.1109/TPAMI.2007.1049.

[11] Fienberg, S. E.; Holland, P. W.: On the choice of flattening constants for estimating multinomial probabilities. *Journal of Multivariate Analysis*, ročník 2, č. 1, 1972: s. 127–134.

[12] Glover, A. J.; Maddern, W. P.; Warren, M.; aj.: OpenFABMAP: An open source toolbox for appearance-based loop closure detection. In *ICRA*, IEEE, 2012, ISBN 978-1-4673-1403-9, s. 4730–4735.

[13] González, R.; Rodrıguez, F.; Guzmán, J.; aj.: Comparative study of localization techniques for mobile robots based on indirect kalman filter. In *Proceedings of IFR Int. Symposium on Robotics*, 2009, s. 253–258.
URL http://www.ual.es/~rgs927/papers/ramon-gonzalez-isr09.pdf

[14] Jacobs, L. F.; Schenk, F.: Unpacking the Cognitive Map: The Parallel Map Theory of Hippocampal Function. *Psychological Review*, ročník 110, 2003: s. 285–315.

[15] Kröse, B. J. A.; Vlassis, N. A.; Bunschoten, R.; aj.: A probabilistic model for appearance-based robot localization. *Image Vision Comput.*, ročník 19, č. 6, 2001: s. 381–391.

[16] Lowe, D. G.: Object Recognition from Local Scale-Invariant Features. In *Proc. of the International Conference on Computer Vision, Corfu*, 1999.

[17] Madan, S.; Dana, K. J.: Modified balanced iterative reducing and clustering using hierarchies (m-BIRCH) for visual clustering. *Pattern Analysis and Applications*, 2015: s. 1–18.

[18] Newman, P. M.; Cole, D. M.; Ho, K. L.: Outdoor SLAM using Visual Appearance and Laser Ranging. In *ICRA*, IEEE, 2006, s. 1180–1187.

[19] Paul, R.; Newman, P.: FAB-MAP 3D: Topological mapping with spatial and visual appearance. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, 2010, s. 2649–2656.

[20] Raykar, V. C.; Duraiswami, R.: Fast optimal bandwidth selection for kernel density estimation. In *SDM*, editace J. Ghosh; D. Lambert; D. B. Skillicorn; J. Srivastava, SIAM, 2006, ISBN 978-1-61197-276-4, s. 524–528.
URL http://www.umiacs.umd.edu/labs/cvl/pirl/vikas/Software/optimal_bw/optimal_bw_code.htm

[21] Rekleitis, I. M.: A particle filter tutorial for mobile robot localization. *Centre for Intelligent Machines, McGill University, Tech. Rep. TR-CIM-04-02*, 2004.
URL ftp://202.191.56.69/vinhlt/Papers/MustReadPapers/Particle%20Filter%20Tutorial.pdf

[22] Salas-Moreno, R. F.; Newcombe, R. A.; Strasdat, H.; aj.: SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In *CVPR*, IEEE, 2013, s. 1352–1359.

[23] Ulrich, I.; Nourbakhsh, I. R.: Appearance-Based Place Recognition for Topological Localization. In *ICRA*, IEEE, 2000, ISBN 0-7803-5889-9, s. 1023–1029.

[24] Welch, G.; Bishop, G.: An Introduction to the Kalman Filter. Technická Zpráva 95-041, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995.
URL http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html

Seznam příloh

Příloha A

Obsah DVD

- Zdrojové kódy lokalizační metody v adresářích **/VisualLocalization/src/** a **/VisualLocalization/kernels/**
- Projekty pro překlad na platformě Windows v adresáři **/VisualLocalization/win/**
- Programová dokumentace vytvořená nástrojem Doxygen v adresáři **/VisualLocalization/doc/**
- Tato práce ve formátu PDF v adresáři **/Thesis/**
- Zdrojové kódy této práce ve formátu LaTeXv adresáři **/Thesis/latex/**
- Přeložené spustitelné soubory lokalizační metody v adresáři **/VisualLocalization/bin/**
- Testovací datové sady v adresáři **/VisualLocalization/big_data/**
- Připravené experimenty v adresáři **/VisualLocalization/experiments/**

Příloha B

Možný způsob řízení robota

V této kapitole je navržen jednoduchý způsob řízení pohybu robota po naučené cestě na základě implementovaného algoritmu lokalizace pomocí vizuální informace. Aby nebylo nutné vybavovat robota dalšími senzory, je tato metoda řízení založena také pouze na informacích z lokalizačního algoritmu a vizuální informaci získané kamerou.

Mapa tedy tvoří jednoduchou cestu $P = (p_1, p_2, \ldots, p_k, p_{k+1}, \ldots, p_{k+m}, p_n)$ v podobě uspořádané sekvence lokací je třeba nalézt bod p_k, kde se robot právě nachází. K tomuto účelu je využit navržený algoritmus lokalizace, který je schopný tento bod nalézt na základě aktuálního pozorování f_a. Řízení pak spočívá ve volbě takové akce, po jejímž provedení se pozorování f_a změní na f'_a a bude platit, že $p(p_k|f'_a) < p(p_{k+i}|f'_a)$ pro $0 < i \leq m$. Tedy pravděpodobnost, že se nacházíme v některé z následujících lokací je za předpokladu nového pozorování větší, než pravděpodobnost, že se nacházíme stále v lokaci p_k. Protože při sledování cesty není problémem, aby byla některá lokace přeskočena, není nutné, aby platilo $i = 1$. Maximální počet přeskočených lokací je však shora omezen pomocí m.

Samotný pohyb v každém kroku je pak vypočten na základě transformace mezi aktuálním a nejbližším snímkem v sekvenci f_k jako zobrazení h_a: $f_a \rightarrow f_k$, v případě, že h_a je identita, však dojde k porušení podmínky uvedené výše (jelikož $f_a = f'_a$). Z tohoto důvodu je akce volena tak, aby vždy došlo ke změně f_a. To je zajištěno tak, že $h = h_{k+m} \circ \cdots \circ h_{k+1} \circ h_k \circ h_a$, kde h_i: $f_i \rightarrow f_{i+1}$. Transformace jsou skládány tak dlouho, dokud jejich velikost nedosáhne daného limitu, tím je vyřešen problém popsaný výše a také dochází ke zlepšení plynulosti pohybu.

B.1 Výpočet transformace a pohybu

Výpočet transformace mezi dvěma snímky je založen na vyhodnocení homografie H mezi jejich odpovídajícími klíčovými body. Pohyb pak můžeme popsat jako vektor $m \in \mathbb{R}^3$ (rotace ve dvou osách a pohyb dopředu/dozadu). Pro hledání homografie mezi snímky f_a a f_k je třeba spárovat jejich klíčové body (na základě jejich deskriptorů d^i_a a d^j_k). Následně je třeba nalézt zobrazení, které splňuje podmínku, že počet klíčových bodů, pro které platí: $\|H(l^i_a) - l^j_k\| > \alpha$ bude minimální. Tedy maximum nalezených korespondencí bude odpovídat tomuto zobrazení. Za tímto účelem je možné využít známý algoritmus *RANSAC*.

V případě, kdy l^j_i jsou body v $\mathbb{R}^2$ bude toto zobrazení tvořeno maticí 3x3, kterou lze rozložit na její jednotlivé složky: rotace, měřítko, zkosení a posun. Následně je možné použít tyto hodnoty pro odhad pohybu kamery (posun jako rotace ve svislé nebo horizontální ose a změnu měřítka jako pohyb vpřed/vzad). Mimo rozložení matice homografie (tzv.

dekompozice homografie) lze využít jednodušší metodu založenou na promítnutí předem známé a dobře definované množiny bodů získaným zobrazením H. Popisované řešení pak využívá promítnutí čtverce (tedy čtveřice bodů kolem středu snímku). Posun jejich středu pak má význam rotace kamery a dle změny velikosti čtverce jsme schopni odhadnout pohyb kamery vpřed/vzad. Tedy $m = (p \cdot (1,0), p \cdot (0,1), \frac{\|q_1 - q_3\| + \|q_2 - q_4\|}{2})$, kde q_i jsou vrcholy čtverce, $p = \frac{\sum_{i=1}^{4} q_i}{4} - s$ a s je střed snímku. Výsledný vektor m je následně normalizován a vynásoben koeficienty odvozenými od rychlosti pohybu a rotace robota. Je zřejmé, že zde zanedbáváme dva rozměry volnosti - pohyb kamery kolmo ke směru jejího pohledu. Navíc je také zanedbán časový odstup mezi počátkem a koncem transformace obrazu, ze kterého by bylo možné snadno odvodit informaci o rychlosti provedení pohybu.

Výhodou této metody je snadná možnost validace získaného zobrazení H a to tak, že ověříme deformaci promítnutého čtverce (pokud tato přesáhne danou mez je zobrazení prohlášeno za chybné a snímek zahozen). Jednoduchou metrikou deformace čtverce může být například ověření úhlu, který svírají jeho diagonály jako $\frac{q_1 - q_3}{\|q_1 - q_3\|} \cdot \frac{q_2 - q_4}{\|q_2 - q_4\|} < \phi$.

B.2 Navigace po cestě

Sledování cesty lze tedy řešit za pomoci lokalizace pomocí navrženého algoritmu, pomocí kterého je získána lokace p_k a její popis f_k. Dále lze tedy z reprezentace aktuálního snímku kamery f_a a lokace f_k vypočítat vektor pohybu $m_{a \to k}$, který ovšem může být nulový. Proto konečný vektor pohybu m tvoří $m_{a \to k} + \sum_{i=k}^{n} m_{i \to i+1}$, tak aby velikost m byla nejmenší taková, že $\|m\| > \alpha$. Kde n je počet snímků v modelu cesty a α je minimální velikost pohybu v jednom kroku. Pohyb mezi snímky ($m_{i \to i+1}$) může být pak vypočten již během vytváření modelu. Člen $m_{a \to k}$ pomáhá omezit chybu, která by vznikla a neomezeně rostla v případě, kdy se agent nenachází přesně v bodě p_k cesty.