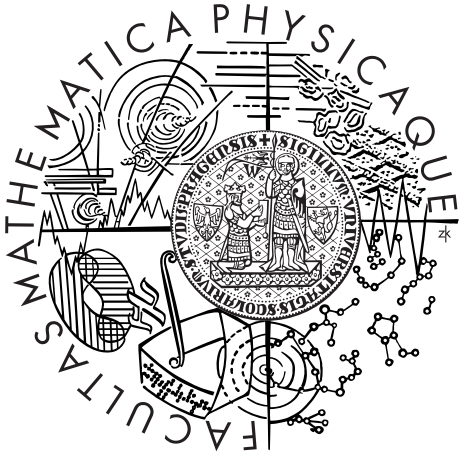
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Petr Fejfar

Rozpoznávání a sledování objektů

Katedra softwarového inženýrství

Vedoucí bakalářské práce:	RNDr. David Obdržálek, Ph.D.
Studijní program:	Informatika
Studijní obor:	Informatika - obecná informatika

Praha 2014

Jsem velkým dlužníkem RNDr. Davidovi Obdržálkovi Ph.D., vedoucímu této práce. Jeho podněty velkou mírou směrovaly mé bádání a vždy mi pomohly zorientovat se v tématu. Tímto mu chci poděkovat.
Děkuji své rodině za podporu a občasné nastavení zrcadla, které mě vždy posunulo dále.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V Praze dne 12.5.2014 Petr Fejfar

Název práce: Rozpoznávání a sledování objektů

Autor: Petr Fejfar

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. David Obdržálek, Ph.D., Katedra teoretické informatiky a matematické logiky

Abstrakt: Práce se zabývá rozpoznáváním a sledováním objektů pomocí dálkoměrných laserových senzorů. Práce zkoumá řešení podobných problémů jinými autory a analyzuje problém jako takový. Je vybrána aplikace úlohy v prostředí robotické soutěže jako referenční problém, který je následně vyřešen. Je kladen důraz na případný port řešení na ne-PC platformy. Základním kamenem řešení jsou Kalmanovy filtry a identifikace objektů podle jejich pozice a rychlosti.

Klíčová slova: rozpoznávání a sledování objektů, Kalmanův filtr, dálkoměrný senzor

Title: Object detection and tracking

Author: Petr Fejfar

Department: Department of Software Engineering

Supervisor: RNDr. David Obdržálek, Ph.D., Department of Theoretical Computer Science and Mathematical Logic

Abstract: This work focuses on object recognition and tracking by using laser range finder sensor. The topic examines various solutions proposed by other authors and analyzes the issue as a whole. The application of the task then takes place in a robot competition, which is used as a reference problem and solved subsequently. Port solutions are emphasized over PC platform solutions. The basis of the solution utilizes the principle of Kalman filters and the identification of objects by their position and speed.

Keywords: object detection and tracking, rangefinder, Kalman filter

Obsah

Úvod

Rozpoznávání a sledování objektů

Úlohou rozpoznávání objektů je vyhledávat v senzorických datech (obrázky, videa, vzdálenostní informace atd.) objekty reálného světa. Při sledování bereme sérii takovýchto dat v čase a hledáme množinu cest jednotlivých objektů v době sledování. V praxi nikdy nemáme dostatek informací o sledovaných objektech, abychom mohli jednoznačně určit model světa[1]. Problém má i filosofický rozměr. Hledáme totiž matematický model reálného světa. K vyřešení úlohy rozpoznávání a sledování objektů v její obecné poloze potřebujeme vyřešit mnoho velkých problémů umělé inteligence. To je velmi obtížné. I přes tyto obtíže je motivace sledovat objekty obrovská. Požadavky přichází z mnoha odvětví lidské činnosti, jmenujme například oblast autonomních vozidel, zabezpečovacích zařízení nebo dopravních systémů. Relaxací problému a jeho aplikací na konkrétní situaci lze dosáhnout alespoň parciálních výsledků. S tímto přístupem často dosahujeme velmi přesvědčivých výsledků (blíže v kapitole Analýza).

Motivace a cíl

Pokládáme si za cíl vyřešit konkrétní aplikaci úlohy rozpoznávání a sledování objektů. Experimentálně naimplementujeme systém, který za použití běžně dostupných prostředků otestujeme v reálných situacích. Výstupem práce bude konkrétní implementace systému i analýza chování systému v modelových situacích.
Jednou z výzev, kterou vidíme, je rozšíření a popularizace pokročilých odvětví robotiky. Pro osoby bez hlubších matematických znalostí je obtížné proniknout do teoretických oblastí. V robotice lze

[1] Praktické nástrahy jsou demostrovány v sekci 1.2 v Image Processing. Analysis. and Machine Vision[8].

najít mnoho teoreticky vyřešených úloh a hotových implementací. Pro člověka neznalého problému jsou bohužel první krůčky velmi obtížné. Jedním ze způsobů, jak podpořit rozvoj robotiky je pořádání soutěží pro mladé týmy. Těchto soutěží je mnoho a jsou zaměřené na různě obtížné úlohy. Obzvláště v autonomní robotice[2] jsou některé úlohy velmi rozsáhlé a přechod od jednodušších úloh ke složitějším zahrnuje učení mnoha nových věcí najednou. To mnoho lidí odrazuje od dalšího studia a seberozvoje v robotice. Pokusíme se naše řešení postavit tak, aby jeho použití pomohlo začínajícím týmům při stavbě jejich robotů.

Budeme se inspirovat soutěží, které se účastnily desítky týmů[3]. Problém podřídíme tomu, abychom docílili řešení relevatnímu při konstrukci robota na tuto soutěž. Od výběru referečního problému jako problému řešeného na robotické soutěži si slibujeme přínos do praktických částí robotiky.

Řešení vyhledávání a obecně práce se sledováním objektů, bylo na soutěži SICK Robot Day 2012 řešeno mnoha týmy. Drtivá většina řešení byla implementována naivními algoritmy a výsledky byly v řadě případů nedostatečné[4]. Námi vypracované řešení nabídne možnost zlepšení řídícího softwaru pro roboty konstruované na budoucích robotických soutěžích.

Struktura práce

V kapitole Analýza problém analyzujeme a provedeme řešerši prací ostatních autorů. V kapitole Návrh popíšeme způsob, jakým budeme problém řešit a specifikujeme jádro algoritmu. Myšlenky použité při implementaci aplikace popíšeme v kapitole Implementace. Chování našeho řešení v referečních situacích rozebereme v kapitole

[2]Autonomní robot je takový robot, který vykonává zadané úkoly z velkou mírou samostatnosti, URL `http://en.wikipedia.org/wiki/Autonomous_robot` [citováno 7. května 2014]

[3]SICK Robot Day 2012, URL `http://www.sick.com/group/DE/home/pr/events/robot_day/Seiten/Robot_day_2012.aspx` [citováno 7. května 2014]

[4]Subjetivní pozorování autora.

Diskuze a budoucí práce, zde také navrhneme možné zlepšení a rozšíření algoritmu nad rámec zadání.

1. Analýza

1.1 Stávající řešení

Oblíbeným senzorem, který se používá pro sledování objektů, je videokamera. Kamery se často využívají při sledování pohybujících se osob a automobilů. Aplikace nacházíme například v bezpečnostních systémech, při počítání lidí, kteří projdou sledovanou oblastí atd. Z videonahrávky snadno získáme informace o texturách objektů, problém je získaní informace o tvaru objektů. Kvůli špatné optice a špatným viditelnostním podmínkám bývají záznamy velmi zašuměné.

Problematikou sledování a rozpoznávání objektů se zabývá mnoho prací. Z množství prací vybereme typické zástupce a popíšeme jejich řešení. Cheriyadat, Bhaduri, Radke[1] řeší problém sledování pohybujících se objektů. Objekty se vyhledávají podle jejich pohybu mezi snímky a jejich předem zadané přibližné velikosti. Autoři navrhují následující algoritmus. Snímek videa chápeme jako matici barev a barvy jako reálné číslo. Po sobě následující snímky od sebe odečteme a tím získáme matici znázorňující pohyb v daném snímku. Na okrajích pohybujících se objektů jsou hodnoty těchto matic nejvyšší. V těchto maticích vyhledáme rohy pomocí Shi-Tomasi-Kanade detektoru[7] a Rosten-Drummond detektoru[9]. Kanade-Lucas-Tomasi algoritmus[10], nalezne trasu rohů pohybujících se v čase. Autoři definují metriku "podobnosti pohybu" rohů a hledají shluky rohů, které se podle této metriky pohybují podobně. Ty označí za pohybující se objekty.

Sledování objektů zpracováním videa je poměrně oblíbené téma, proto vzniklo mnoho přístupů jak problém řešit. Algoritmy sledování objektů ovšem pracují jen za jistých podmínek a mají různé slabiny. Siebel, Maybank[2] navrhují sledovat objekty pomocí více

algoritmů a výsledky sjednotit. Přístup demonstrují na problému vyhledávání lidí ve videu. Používají následující přístupy: hledání pohybujících se oblastí, hledání siluet lidských postav v obrazu a hledání siluet hlav. Následné hypotézy o pohybujících se lidech vzniklé v jednotlivých algoritmech následně slučují a filtrují (např. dvojice konverzujících chodců, typicky vypadá jako jedna velká pohybující se oblast, algoritmus hledající siluety postav však správně nalezne dvě postavy). Sjednocením více algoritmů autoři vytvořili robustní algoritmus.

Dalším často používaným senzorem je dálkoměrný senzor. Ten nám dává přesnou informaci o tvaru, avšak téměř žádnou informaci o textuře. Tvar objektu pozorujeme vzorkováním vzdáleností od senzoru. Dálkoměrné senzory potřebují přímý výhled na sledovaný objekt. Narozdíl od kamery máme větší rozteč mezi paprsky, které vzorkují povrch objektu, a tedy musíme být se senzorem blíže sledovaným objektům. Při nevhodném umístění senzoru blízké objekty "zastiňují" jiné objekty a v těchto situacích nemáme žádnou informaci o objektu ve "stínu". Proto musíme volit umístnění senzoru pečlivě.

Song, Zhao, Zha, Shibasaki[3] používají dálkoměrné senzory, které pozorují plochu ve výšce lidských kotníků. Akumulují pozorování více senzorů rozmístěných po místnosti a vyhledávají konce jednotlivých paprsků senzorů. Tam, kde paprsek končí, se nachází objekt. Body na konci paprsků se filtrují pomocí algoritmu Parzen window[11] a tam, kde jsou jejich lokální maxima, jsou lidské nohy. Autoři pozorují nohu, která při chůzi stojí, a párují dvojice nohou, které patří jedné osobě. Z informací o nohou odvozují pozici sledované osoby.

V jiné práci[4] autoři systém postavený z dálkoměrných senzorů rozšiřují o videokameru. Tímto přístupem jsou schopni modelovat situace, kdy je dálkoměrný senzor zakryt a neposkytuje relevantní data o objektech, zatímco videokamera zakryté objekty vidí. Toho

dosahují vhodným postavením kamery.

1.2 Definice řešeného problému

Náš problém je inspirován soutěží SICK Robot Day 2012. Soutěžilo se v kryté hale, kde byl padesát centimetrů vysokými krabicemi ohraničen oválný prostor o průměru přibližně 15 metrů. Na okrajích byly mezi bednami tři metrové brány barev tří týmů: bílá, zelená a žlutá. Na ploše se nacházelo 30 míčů o velikosti 22 cm, v bílé, zelené a žluté barvě. Míče na začátku každého kola stály nehybně.

Obrázek 1.1: Na levém obrázku je jeden ze zástupců soutěžících robotů. Vpravo vidíme pohled na celé hřiště. V popředí je brána bílého týmu, v pozadí jsou vidět brány zbylých týmů. Zdroj obrázků `http://static.sick.com/webgallery_robotday_2012/` [citováno 7. května 2014]

Soutěže se zúčastnily evropské týmy složené převážně ze studentů. Každý tým soutěžil s robotem vlastní konstrukce a použitým senzorem SICK LMS 100 (více o senzoru v sekci Senzory). Roboti byli podle pravidel výrazně větší než míče a byli tedy snadno rozlišitelní. V soutěži roboti vozili míče do vlastních domečků. Soutěžilo se na kola, která trvala 10 minut. Kolo vyhrál tým, jehož robot dovezl nejvíce míčů v barvě svého domečku. Každého kola se zúčastnili tři roboti, každý začínal v domečku na okraji hřiště.

1.3 Zvolený přístup

Využijeme faktu, že míče leží v jedné rovině a použijeme laserové dálkoměrné senzory, kterými budeme skenovat plochu v okolí robota. Ze skenů získáme pozice viditelných míčů a jejich pohyb budeme dále modelovat pravděpodobnostními filtry. To nám umožní kromě jejich pozice určit i jejich rychlost a předvídat jejich budoucí pohyb[1]. Senzor pracuje pouze se vzdálenostními daty a z nich dokáže odvodit pouze tvar objektů, jejich barvu zjistit nedokážeme.

1.4 Dekompozice úlohy a terminologie

Naši úlohu lze rozdělit na dva odlišné úkoly. Prvním úkolem je zpracovat senzorická data ze senzoru a určit z nich relativní pozici míče vůči senzoru. Druhým úkolem je tyto jednotlivé pozice zpracovávat a s jejich pomocí určit jednotlivé trasy jednotlivých míčů. První pojmenujeme rozpoznávání objektů a druhý sledování objektů.
Rozpoznávání objektů má na vstupu data získaná při pozorování světa senzory (jednotlivé snímky videa, informace z dálkoměrného senzoru, rentgenový snímek, atd.). Na výstupu produkuje pozice jednotlivých objektů. Obecně může generovat i jiné výstupy: například zda je objekt přítomen, jakou má objekt barvu, atd. Rozpoznávání probíhá pro jednotlivé snímky odděleně a nemůže tedy pracovat s veličinami závislými na čase. Sledování oproti tomu pracuje s více snímky najednou a má informace o vzájemné závislosti snímků (například čas, poloha, orientace). Při sledování objektů tuto informaci využijeme k určení více vlastností objektů. Typicky se jedná o rychlost a trajektorii objektů.

[1] Vezmeme v potaz setrvačnost objektů a jejich rychlost. Rychlost dopočítána ze záznamů o jednotlivých pozicích v průběhu času $v = \delta x / \delta t$

1.5 Rozpoznávání objektů

Rozpoznávané objekty musí mít určité vlastnosti, které je odlišují od okolního světa. Bez toho není možné objekty najít[2]. Pro stavbu algoritmu musíme vědět, jaké vlastnosti objekt reprezentují. Bez této znalosti nevíme, co v senzorických datech hledat. Typicky použité vlastnosti k nalezení objektů jsou: objekt má jinou barvu než pozadí, objekt se pohybuje oproti pozadí, objekt je určitého tvaru, atd. V našem případě budeme pracovat s informací o vzdálenosti částí objektů od senzoru. K detekci použijeme tvar, který je možný z této informace odvodit.

1.5.1 Odstranění šumu

Prvním krokem je příprava dat k následnému zpracování. Naší přípravou bude potlačení šumu, který naše data obsahují. Šum je náhodný signál přidaný k původnímu signálu. Původci šumu jsou například nedokonalost senzorů nebo nepřesný způsob uložení dat. Jedná se o případy, kdy je optická část senzoru znečištěna nebo laserové paprsky jsou rozptylovány kapkami deště. Šum může způsobit i nepřesná reprezentace dat jako binární hodnoty[3].

Přítomnost šumu ztěžuje práci algoritmům, které použijeme na samotné rozpoznávání objektů. Budeme se zabývat pouze takovým šumem, který připomíná „sněhové vločky“ nebo „zrníčka prachu“ na fotografiích. Tento druh se v praxi vyskytuje často, protože výskyt tohoto šumu je na jednotlivých bodech dat (pixelech, paprscích) nezávislý. Toto odpovídá vlastnostem běžně používaných senzorů. Tento šum také můžeme efektivně potlačit algoritmy, které vycházejí z myšlenky, že obraz tolik „neskáče“. Formálněji vysloveno, de-

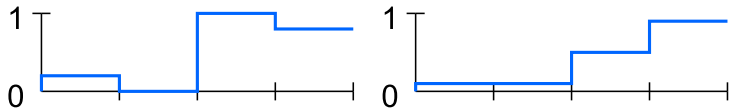
[2]Pro ilustraci: na fotce pořízené v absolutní tmě, nenajdeme černý papír

[3]Senzor produkuje vzdálenostní data jako 16 bitové hodnoty. Tato reprezentace je konečná a tedy není schopná zachytit reálný stav světa. Část informace se tedy ztratí a to způsobuje šum.

rivace funkce popisující jeho barvu je malá, či obraz po Fourierově transformaci neobsahuje vysoké frekvence. Tedy sousední body dat nejsou příliš rozdílné[4]. Stačí tyto frekvence odstranit. Problém nastává v místech obrazu, kde neplatí předpoklad, že v daném místě má vysoké frekvence pouze šum - například hrany. Hrany jsou charakterizovány také vysokými frekvencemi a odstraněním šumu je poškozujeme. Blíže v sekci Hrany. Redukci šumu musíme tedy provést jen po určitou úroveň.

1.5.1.1 Průměrování

Základním algoritmem pro redukci šumu je výpočet nových hodnot jako plovoucí průměr okolí bodu. Je potřeba projít data bod po bodu a nahradit aritmetickým průměrem jejich okolí. Algoritmus lze rozšířit tím, že aritmetický průměr nahradíme váženým průměrem. Typicky body blíže zpracovávanému bodu budou mít vyšší váhu. Nevýhodou tohoto algoritmu je jeho chování na hranách (viz obrázek 1.2).



Obrázek 1.2: Odstranění šumu pomocí průměrování. Na obrázku vlevo vidíme funkci obrazu před průměrováním. Vpravo vidíme funkci po průměrování. V blízkosti ostrých hran algoritmus dopočítá hodnotu nového bodu na přibližně polovinu hodnoty bodu před a po hraně a tím hranu výrazně vyhladí. Vzniknou tak dvě malé hrany místo jedné velké a to práci algoritmu na detekci hran činí složitější.

1.5.1.2 Mediánový filtr

Mediánový filtr oproti průměrování nenahrazuje body průměry svého okolí, ale mediánem okolí. Výhoda tohoto přístupu spočívá v tom,

[4]Blíže v kapitole 5.3 Image Processing. Analysis. and Machine Vision[8]

že lépe zachovává hrany.

1.5.2 Detekce zajímavých oblastí

Data na vstupu bývají velmi složitě interpretovatelná (například barvy fotografie nám sice poskytnou informaci, kde se nachází hledaný automobil, avšak zjištění polohy automobilu není pro počítače lehká úloha). Existuje proto mnoho různých transformací dat, která jejich interpretaci zjednoduší. Aplikováním správných transformací můžeme úlohu počítači značně zjednodušit. Výběr těchto transformací zavisí na expertních znalostech o prostředí problému, který řešíme.
Obrázky jsou běžně reprezentovány jako dvourozměrné pole reálných čísel vyjadřující barvu v jednotlivých pixelech. Skeny jsou reprezentovány jako jednorozměrné pole reálných čísel reprezentující délky paprsků. Tyto reprezentace převedeme na takové, které nám poví více o objektech, které sledujeme. V našem případě jde hlavně o hrany, které se vyskytují ve skenech na okrajích míčů.

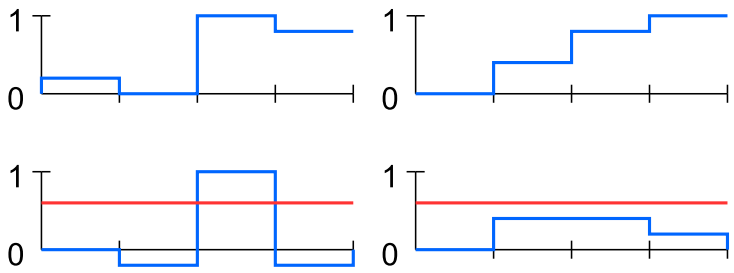
1.5.2.1 Prahování

Prahování je základní technikou, jak odlišit zajímavou oblast. Principem techniky je, že z obrázku vezme všechny body překračující určitou hodnotu (práh).

1.5.2.2 Hrany

Vezmeme si objekt, který se oproti pozadí liší barvou. Okraje takového objektu jsou charakteristické tím, že hodnota barvy se dramaticky změní. V matematické terminologii: obrázek reprezentujeme jako funkci $f : pozice\_pixelu \rightarrow hodnota\_barvy$, pak hrana je místo, kde je absolutní hodnota parciální derivace f vysoká. Obrázky jsou diskrétní, a proto k detekci místo derivace používáme

diferenciál sousedních hodnot. V takto transformovaném obraze se k odlišení hran od zbytku používá prahování. Tento přístup je však náchylný vůči šumu (viz obrázek 1.3).



Obrázek 1.3: Detekce hran pomocí prahování diferenciálu. V horní řadě jsou barevné funkce jednorozměrných obrázků. Pro zjednodušení 0 znamená černou, 1 znamená bílou barvu. Hodnoty mezi jsou odstíny šedi. Oba obrázky reprezentují skok funkce z nuly na jedna na krátkém úseku, tedy hranu. Levý obrázek má hranu výraznou, pravý obrázek má na hraně šum, který hranu rozmělní. Aplikujeme diferenciál mezi sousedními body obrázku a dostaneme funkci znázorněnou na spodním řádku. Prahování na úrovni 0,6 v levém případě hranu odhalí, v pravém případě neodhalí.

1.6 Sledování objektů

Sledování objektů v čase řeší těžší úlohu než je rozpoznávání. Přidává si za cíl objekty nejenom najít, ale také jim přiřadit identitu. Díky tomu můžeme určit více informací o sledovaném objektu. Například pokud máme informace o pozici objektu v čase, umíme určit jeho rychlost. Na vstupu oproti rozpoznávání nemá pouze jeden obrázek, ale jejich množinu s časovými razítky. Pro každý tento obrázek provede rozpoznávání objektů a získá množinu pozic objektů pro každý obrázek. K těmto objektům přiřadí identifikátory tak, aby každý objekt z reálného světa měl jedinečné id. Abychom toto mohli udělat, potřebujeme další informace o reálném světě. Tou může být barva objektu nebo obecně nějaká vlastnost jeho vzhledu, která ho odlišuje od ostatních. Pokud známe pozici a rychlost objektu, tak víme, že pokud další snímek byl pořízen krátce po snímku prvním, bude objekt jen o kousek vzdálen oproti snímku prvnímu.

1.6.1 Kalmanův filtr

Pro sledování objektů, které se pohybují, se často využívá Kalmanova filtru. Kalmanův filtr dokáže zpracovat normálně rozdělený šum na vstupních datech. Dokáže se také vypořádat s neúplnými daty na vstupu (vzniklými zastíněním nebo nedokonalostí senzoru). Za pomoci matematického modelu přímočarého nebo zrychleného pohybu, umí odvodit skryté proměnné modelu (například z měřených pozic v čase odvodí rychlost a zrychlení). To vyhovuje naší úloze - budeme schopni predikovat budoucí pohyb míče (pomocí jeho pozice a rychlosti), pracovat s nepřesnými daty ze senzoru a sledovat míče, i když o nich nedostáváme aktuální informace (například protože jsou v zákrytu).
Algoritmus pracuje v krocích. V každém kroku si drží vnitřní stav reprezentace světa. Stav si drží jako normálně rozdělené veličiny, se střední hodnotou (vektor x_t) a kovarianční matici směrodatné odchylky P_t reprezentující odhad nepřesnosti měření. Tato odchylka zachycuje to, jak filtr „věří“ aktuálnímu vnitřnímu stavu filtru. Střídají se dva typy kroků: predikce a korekce. Při predikci filtr odhadne budoucí vnitřní stav na základě matematického modelu dodaného filtru. Zde se typicky důvěra ve vnitřní stav sníží. Při korekci se upravuje vnitřní stav filtru podle hodnot naměřených pomocí senzoru. Korekce odhad naopak upřesní.
Model závislosti stavu v čase t na stavu v čase $t-1$ je

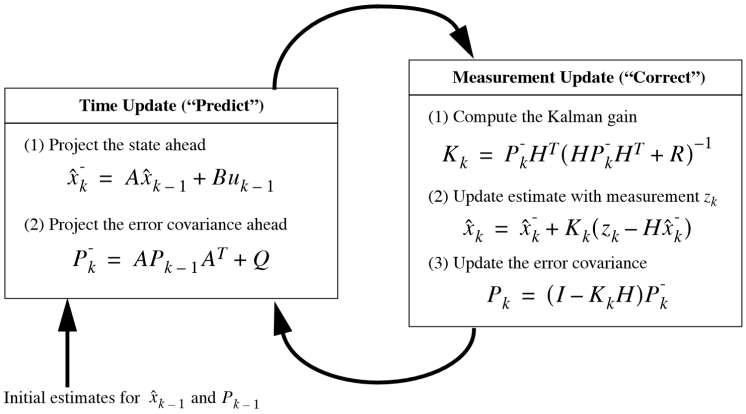
$$x_t = Ax_{t-1} + Bu_{t-1} + w_t$$

kde x_t je stav v čase t, A je matice přechodu mezi stavy. u_t je řídící signál v čase t, B je matice zobrazení řídícího signálu do stavu a w_t je šum s rozdělením $w_t \sim N(0, Q_k)$. Přechod mezi stavy je lineární funkce, což omezuje použití Kalmanova filtru pouze na lineární

systémy.
Model měření je

$$z_t = H_t x_t + v_k$$

kde z_t je měření v čase t, H_t je matice zobrazení ze stavu do měření, v_t je šum s rozdělením $v_t \sim N(0, R_k)$.
Z modelu odvodíme predikční krok a korekční krok (viz obr. 1.4). Pokud má šum normální rozdělení a reálný svět tedy odpovídá mo-

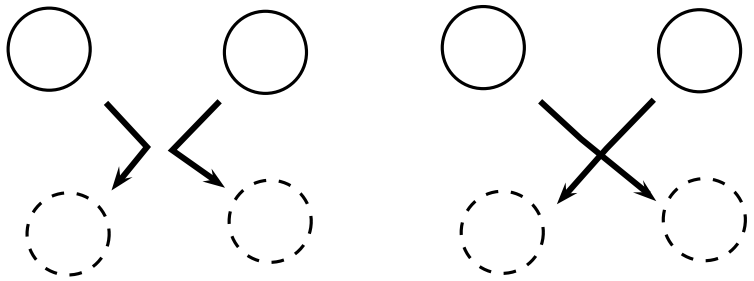


Obrázek 1.4: Životní cyklus Kalmanova filtru. Obrázek převzat ze článku An Introduction to the Kalman Filter[5]

delu, filtr konverguje ke správné hodnotě. To v praxi neplatí, avšak filtr se ukazuje být natolik robustní, že nesplnění těchto předpokladů nevadí.

1.7 Práce s hypotézami

Kalmanův filtr ze své podstaty pracuje pouze s jednou hypotézou o modelovaném světě. V reálných situacích může nastat případ, kdy kvůli nepřesnostem jedna hypotéza nemůže efektivně popisovat naší informaci o světě. (viz obrázek 1.5) Jedná se o situace, kdy nemáme dostatek informací, abychom jednoznačně určili identifikátory objektů.
Využijeme toho, že v budoucnosti budeme mít informací více. Bu-



Obrázek 1.5: Vytvoření více hypotéz. Situace vlevo ukazuje míče, které se srazily. Míče tedy zůstaly každý na své straně. Situace vpravo vypadá stejně jako první situace. Míče se ale nesrazily - pozice si prohodily.

deme si pamatovat všechny možnosti a podle dat, které dostaneme z budoucích snímků, budeme přiřazovat hypotézám věrohodnost. Hypotézy s příliš malou věrohodností zahazujeme.

1.8 Maďarský algoritmus

Maďarský algoritmus[12] (z anglického hungarian algorithm nebo také Kuhn–Munkres algorithm nebo Munkres assignment algorithm) je optimalizační algoritmus na úplném symetrickém bipartitním grafu. Algoritmus budeme používat v situaci, kdy získáme množinu nových měření a budeme je chtít přiřadit k sledovaným objektům. Minimalizujeme perfektní párování na úplném symetrickém bipartitním grafu se stejně velkými partitami, jehož ohodnocení hran reprezentuje vzdálenosti mezi míči. K nalezení takového párování používáme metodu zlepšujících cest v grafu.

Algorithm 1.1 Grafová verze Maďarského algoritmu.[6]

Máme bipartitní graf $G = ((A \cup B), E)$, $A \cap B = \emptyset$

1. krok Pro všechny vrcholy partity A naleznu nejmenší hranu a odečtu od všech hran vedoucích z vrcholu. To samé pro všechny vrcholy z partity B.

2. krok Naleznu maximální párování. Pokud je perfektní, máme vyhráno, pokud ne, pokračuji dalším krokem.

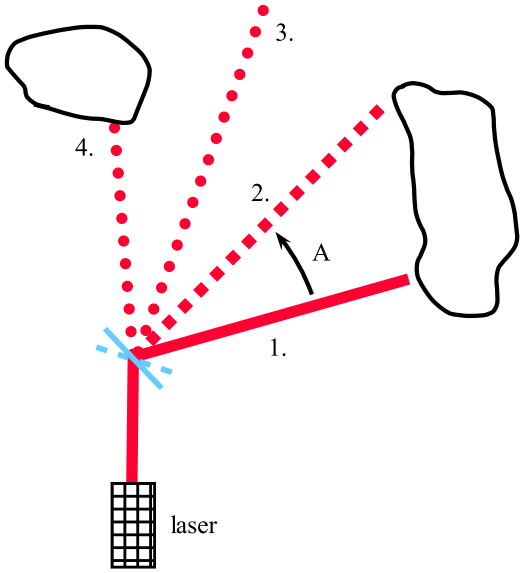
3. krok Naleznu minimální pokrytí V a nastavím $\delta = min_{i \notin V, j \notin V}(c_{ij})$

4. krok Nastavíme hrany na $c_{ij} = \begin{cases} c_{ij} + \delta & , i \in V \wedge j \in V \\ c_{ij} & , i \in V \vee j \in V \\ c_{ij} - \delta & , i \notin V \wedge j \notin V \end{cases}$

5. krok Vrať se na krok 2

1.9 Senzory

Použijeme populární senzor SICK LMS 100[5]. Senzor vrhá sérii laserových paprsků, které měří vzdálenosti od senzoru (názorněji na obrázku 1.6). Tyto paprsky leží na jedné rovině a kromě informace o vzdálenosti vrací ještě informaci o odrazivosti povrchu, na který dopadly. My budeme používat informaci o vzdálenosti, pomocí které budeme pozorovat okolí senzoru.
Senzor komunikuje pomocí protokolu TCP/IP. Vybrané technické parametry, relevantní našemu použití senzoru, můžeme najít v tabulce 1.1. Laserový senzor umístíme do výšky poloviny míče a budeme pořizovat vodorovné skeny hřiště. Skeny, které bude senzor produkovat, budou zachycovat míče, stěny i ostatní roboty (obrázek 1.7).



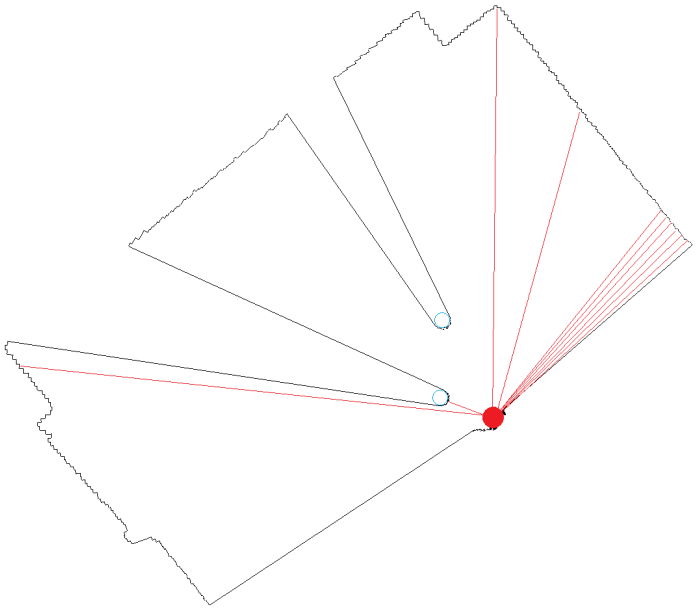
Obrázek 1.6: Schéma laserového dálkoměrného senzoru. Laser vypouští paprsek, který se odráží od rotujícího zrcadla (modrá plná čára), poté se odrazí od nejbližšího objektu a vrací se stejnou cestou zpátky k laseru, kde je zachycen. Senzor měří dobu letu paprsku a pomocí této informace a rychlosti letu laserového paprsku dopočítává délku paprsku. Zrcadlo se otáčí o úhel A (přerušovaná modrá čára) a proces se opakuje.

Dalším možným použitelným, levným a dostupným senzorem jsou

[5]Specifikace senzoru SICK LMS 100, URL `https://www.mysick.com/PDF/Create.aspx?ProductID=33753&Culture=en-US` [citováno 7. května 2014]

velikost úhlu měřené výseče	270
rozteč mezi paprsky skenu	0,25 nebo 0,5
frekvence skenování	25Hz nebo 50Hz
rozsah měřených paprsků laserem	0,5m až 20m.
šířka datového typu reprezentující vzdálenost	16 bitů

Tabulka 1.1: Technické parametry senzoru SICK LMS 100



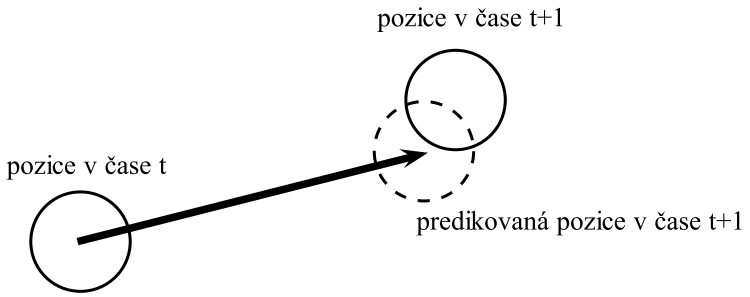
Obrázek 1.7: Vizualizace skenu místnosti s dvěma míči. Červený plný kruh značí pozici senzoru. Pro názornost jsou naznačeny některé paprsky jako červené čáry. Paprsky jsou ve skutečnosti rozmístěny rovnoměrně po pár desetinách stupně. Modré kružnice naznačují pozice míčů. Obdélníkový útvar je půdorys místnosti S9 v budově MFF UK na Malé Straně a je to jediná informace, kterou ze skenu použijeme. Dva výřezy v obdélníku jsou stíny, které vrhají míče.

videokamery. Video nám dá informaci o textuře objektů a tedy můžeme rozpoznávat barvu míčů. Videokamery jsou však velmi závislé na kalibraci na dané světelné podmínky, a tedy méně robustní než laserové dálkoměné senzory. Z tohoto důvodu jsme se rozhodli kamery nepoužít.

1.10 Možné problémy

Díky vlastnostem senzoru nastanou situace, kdy senzor nevidí míč přímo (je ve stínu za jiným objektem). V těchto situacích nemůžeme míč v obecném případě sledovat, (např.: míč mohl ve stínu s jiným

objektem kolidovat nebo projít v jeho těsné blízkosti bez kolize - zde nezjistíme, ke které ze situací došlo). V jednoduchých případech, kdy to lze (v praxi v těch nejčastějších), míč sledovat budeme.
Ve vzdálenosti od senzorů klesá přesnost senzoru a hustota paprsků dopadajících na objekty[6]. Informace o vzdálených objektech tedy bude nepřesná a velmi zašuměná. Objekty vzdálené od robota jsou ovšem málo důležité, protože s nimi robot nemůže interagovat.
Další problém je, že senzor nedokáže míče rozlišit. Senzor pracuje pouze s tvarem míčů a ten je u všech míčů stejný. Míče budeme rozlišovat podle přidané informace o poloze a rychlosti míče (obrázek 1.8). Kvůli nepřesnostem v měření může nastat, že měřená rychlost a pozice jednoho míče bude odpovídat rychlosti a pozici jiného míče. Poté může nastat, že míč zaměníme za jiný.



Obrázek 1.8: Za předpokladu, že naše predikce je správná, můžeme míče identifikovat podle toho, že predikovaná a reálná pozice je blízká.

Nesmíme zapomínat ani na šum senzoru. Měření nejvíce zkreslují odlesky paprsků od měřeného objektu a srážky paprsků laserového senzoru s nečistotami ve vzduchu. Námi sledované míče jsou matné a robot se pohybuje v zastřešeném prostředí, proto můžeme vyloučit soustavné odrážení paprsků od míčů nebo vliv deště a jiných povětrnostních podmínek na měření senzoru. Z náhodných odlesků se můžeme zotavit, použijeme-li při pozorování předpoklad, že odlesků není mnoho a tedy paprsek není výrazně jiné délky než oba jeho sousedi.

[6]Pro ilustraci pokud je míč vzdálen cca 25 m od senzoru trefí ho průměrně pouze jeden paprsek senzoru. Při vzálenosti cca 5m jsou to cca 3 paprsky.

1.11 Prostředí práce

V rámci sledování objektů musíme zpracovat velké množství dat v krátkém čase. Zároveň nemůžeme použít velmi výkonný HW, protože vše musí být součástí mobilního robota. Pokusíme se vytvořit řešení, které bude dostupné i pro mladé robotické týmy s nízkým rozpočtem. Jako referenční HW volíme běžně dostupný levný notebook s ne příliš výkonným procesorem[7], 3 GB pamětí a operačním systémem Windows. Výhodou tohoto řešení je relativně vysoký výpočetní výkon, snadná dostupnost HW a možnost připojení na robota, pokud použijeme notebook menší velikosti. Budeme podporovat i port na jiné než PC platformy.

[7] Specifikace procesoru, URL `http://ark.intel.com/products/42109/Intel-Core2-Duo-Processor-T6670-2M-Cache-2_20-GHz-800-MHz-FSB` [citováno 7. května 2014]

2. Návrh

2.1 Algoritmus

V této kapitole navrhneme konkrétní algoritmus, který aplikujeme na úlohu definovanou v sekci Definice řešeného problému. Konkrétní implementační detaily rozebereme v kapitole Implementace.

2.2 Jádro algoritmu

Kostru algoritmu lze popsat jako opakující se smyčku obsahující následující kroky.:

1. Načteme data ze senzoru.
2. Provedeme preprocessing dat (zbavíme šumu).
3. Vyhledáme pozice jednotlivých míčů.
4. Jednotlivé nalezené pozice míčů přiřadíme k sledovaným míčům Kalmanovy filtry.
5. Aktualizujeme Kalmanovy filtry o naměřené pozice míčů.
6. Informujeme o nových pozicích.

2.3 Vstup

Na vstupu máme vzdálenostní data z dálkoměrného laserového systému. Výsledky měření dostáváme postupně po určitých časových intervalech. Vstup, který dostaneme při každém takovém intervalu, nazývejme parciálním vstupem. Ze své podstaty se jedná o online algoritmus[1]. Velikost časových intervalů by měla být co nejmenší možná. Čím kratší jsou intervaly, tím blíže realitě jsou výstupy

[1]Online algoritmus, URL `http://en.wikipedia.org/wiki/Online_algorithm` [citováno 7. května 2014]

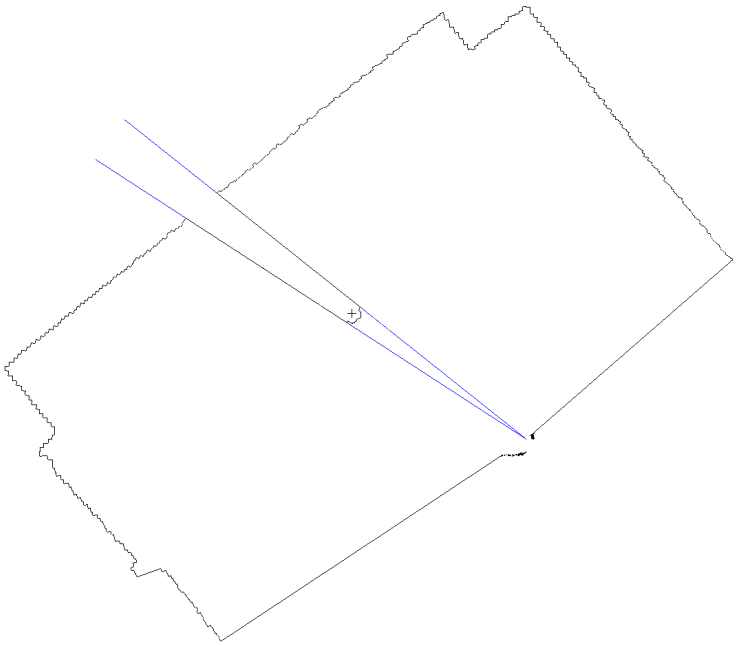
(pozice, rychlosti, sledování). Doba zpracování závisí na složitosti scény, počtu sledovaných míčů, velikosti měřených skenů a výkonu počítače.
Na vstupu je pozice senzoru, jeho natočení v rovině a jeho konfigurace (rozteč paprsků, atd.). Parciální vstupy se skládají z uspořádané množiny čísel, které udávají vzdálenosti pro jednotlivé paprsky senzoru.

2.4 Preprocessing skenů

Skeny jsou zatíženy šumem. Šum závisí na vzdálenosti od senzoru a odrazivosti povrchu objektů. Pokud se paprsek odrazí od lesklé plochy nebo narazí na zrnko prachu ve vzduchu, tak se může měření prodloužit/zkrátit. Objekty v našem prostředí jsou matné a měření tímto způsobem soustavně nezkreslují. Náhodné odlesky od objektů v našem prostředí jsou charakteristické tím, že nijak nezávisí na odlescích okolních paprsků. Druhým typem šumu, se kterým se potýkáme, jsou soustavné nepřesnosti měření laser range finderu.
Na redukci prvního typu šumu použijeme mediánový filtr popsaný v kapitole Analýza. Velikost okénka pro ně stanovíme na tři. To vyhovuje, protože nepředpokládáme, že by byly zašuměny dva sousední paprsky a v případě šumu tedy vyhraje nezašuměný paprsek.
Druhý typ šumu budeme redukovat průměrováním. Průměrování však funguje pro paprsky měřící stále stejnou vzdálenost. To jsou typicky paprsky mířící na zeď. Ty neovlivní hledání míčů a tak je můžeme ignorovat. Tato chyba roste se vzdáleností od senzoru a pro blízké objekty není velká.

2.5 Vyhledávání míčů

Z definice problému jsou míče v prostoru převážně osamocené. Toto bereme jako předpoklad.

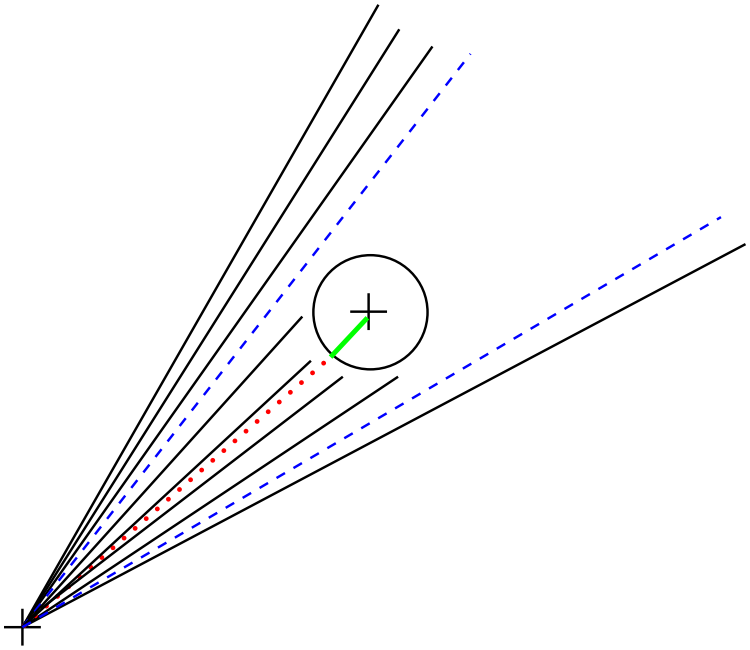


Obrázek 2.1: Hledání míčů ve skenu (modré čáry znázorňují hrany a černý křížek střed nalezeného míče)

V této fázi vezmeme sken ze senzoru a snažíme se najít míče. Míč na skenu (viz kapitola 2.1) vrhá ,,stín“. Pokusíme se detekovat tyto stíny a podle nich hledat míče. Stíny budeme detekovat algoritmem na zjištění hrany popsaným v kapitole analýza. To provedeme tak, že sken považujeme za jednorozměrný obrázek a hrany jako rozdíl mezi sousedními hodnotami. Nalezené hrany prahujeme konstantou zvolenou jako dvě velikosti míče. Prahování hrany nalezne, protože na skenech jsou hrany ostré, nejsou zašuměné. Hrany jsou pro názornost zobrazeny na obrázku 2.1 jako modré paprsky. Dvě po sobě jdoucí hrany s opačnými znaménky ukazují na stín. Nyní vezmeme všechny nalezené stíny a vyřadíme všechny, do kterých se míč nevejde nebo je moc velký na to, aby to byl míč. Ze stínů dopočítáme střed míčů tak, že vezmeme nejkratší paprsek[2] umístíme ho doprostřed úseku mezi hrany definující stín a prodloužíme ho o poloměr míče (názorněji viz obrázek 2.2). Hledání středu lze řešit i způsoby, které zohlední více informací o kulatosti míče. Tyto způ-

[2]Nemůže nastat situace, kdy bude paprsek kvůli šumu výrazně kratší, to jsme vyloučili algoritmem na redukci odrazů při preprocessingu dat.

soby by mohly přinášet přesnější výsledky[3]. Následně použité Kalmanovy filtry jsou dostatečně robustní, proto jsou výsledky postačující.



Obrázek 2.2: Dopočítání středu míčů ze stínů. Modré přerušované paprsky jsou detekované hrany skenu. Červený tečkovaný papsek je osou paprsků modrých a je dlouhý jako nejmenší z paprsků mezi hranami.

Námi navržený algoritmus nedetekuje míče nalepené na sebe. Takto uspořádané míče detekovat nelze, protože detekujeme pouze míče a takovéto objekty se na skenu jeví jako objekty jiné třídy (v extrémním případě mohou míče vyrovnané v řadě vedle sebe působit jako zeď). To je situace, která nenastane často (díky předpokladu o nízké hustotě míčů).

2.6 Párování měření k objektům

Na vstupu máme množinu měření a množinu sledovaných míčů. Výstupem tohoto kroku budou tři množiny: dvojice měření a souvisejících sledovaných hodnot, osamělá měření a osamělé sledované míče.

[3]Domněnka autora. Experimentálně, ani teoreticky nepotvrzeno.

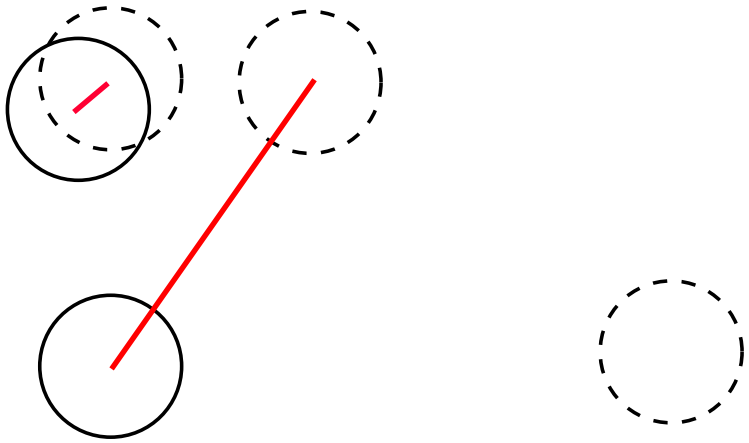
Přiřazení nově naměřených míčů k již sledovaným hodnotám převádíme na optimalizační úlohu. Vezmeme si vzdálenosti všech naměřených pozic míčů od všech sledovaných míčů. Hledáme takové přiřazení jednotlivých měření ke sledovaným míčům, aby součet vzdáleností mezi přiřazenými hodnotami byl co nejmenší. Vzdáleností myslíme Euklidovskou metriku mezi středy míčů. Toto je zadání optimalizační úlohy, kterou řešíme Maďarským algoritmem (podrobněji v 1.8). Předpoklad pro Maďarský algoritmus je, že počet měření bude stejný jako počet sledovaných objektů. My však můžeme mít rozdílný počet měření a sledovaných míčů. Předpokladu docílíme takto: pokud je měření, resp. sledovaných míčů méně, tak je doplníme o chybějící počet virtuálních měření, resp. míčů. Vzdálenosti mezi virtuálními objekty a objekty měření/sledování zvolíme na konstantní hodnotu. Vybereme 0.

Pozorování Námi přidané virtuální hodnoty výsledek nezmění.

Důkaz Nechť máme množinu párů a osamělý objekt. Přidáme virtuální objekt, který se napáruje na některý z párů (toto je chování, které nechceme). Poté součet vzdáleností nově napárovaných hodnot je větší než součet v situaci, kdy se osamělý objekt spáruje s virtuálním. To nastává, protože vzdálenost virtuálního objektu s jakýmkoliv je konstantní a vzdálenost lichého míče od míče z páru je větší jak vzdálenost míčů v páru. Což plyne ze způsobu, kterým jsme si definovali, že budeme míče k sobě přiřazovat (viz obrázek 1.8). Zde jsme dostali spor s optimalitou řešení. □

Formálně hledáme minimální vážené párování na úplném symetrickém bipartním grafu. Výstupní párování (viz 2.3) z algoritmu interpretujeme takto:

- Měření nebo sledované míče spárované s virtualními objekty označíme za osamělé.



Obrázek 2.3: Ukázka nalezených párování Maďarským algoritmem. Čárkované kružnice značí nová měření, ty nakreslené plnou čarou predikované hodnoty filtrů. Červená úsečka mezi středy značí pár. Pár s delší úsečkou je osamělé měření spojené s lichým sledovaným míčem. Takovéto situace je třeba detekovat.

- Páry, které jsou od sebe vzdálené více než o zvolenou konstantu, označíme za osamělé. Konstantu volíme tak, aby neoznačila páry, které k sobě náleží a jsou díky šumu o trochu posunuté. Zvolme ji rovnou ve velikosti dvou průměrů míče. Názorně na obrázku 2.3.
- Všechny zbylé páry k sobě náleží.

2.7 Sledování v čase

Měřené pozice míčů modelujeme Kalmanovým filtrem. Kalmanův filtr používáme na redukci šumu při sledování objektu a k získání hodnot skrytých proměnných z modelu (máme sérii měření pozic v čase a chceme z nich odvodit rychlosti při jednotlivých měřeních). Použijeme původní podobu Kalmanova filtru a model volíme následovně.
Stav modelu

$$X_t = \begin{pmatrix} x \\ y \\ vel_x \\ vel_y \end{pmatrix}$$

kde x, y jsou souřadnice pozice a vel_x, vel_y je rychlost. Přechodovou matici nastavíme na

$$F = \begin{pmatrix} 1 & 0 & time & 0 \\ 0 & 1 & 0 & time \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

kde $time$ je doba časového úseku kroku filtru.
Tyto hodnoty odvodíme z fyzikálního modelu pohybu přímočarého rovnoměrného pohybu. Rozepíšeme si přechod z času t na čas $t+1$ proměnné x_t a $vel_{x,t}$. Proměnnými x_{t+1} a $vel_{x,t+1}$ budeme značit nový stav. Přechod mezi stavy je $X_{t+1} = F \cdot X_t$. Po rozepsání podle rozměru x, dostaneme rovnice $x_{t+1} = x_t + vel_x \cdot time$ a $vel_{t+1} = vel_t$. Substituujeme-li x za y, dostaneme identické rovnice pro druhý rozměr.
Tento model je aproximací reálného světa. V reálném světě na míče působí různé malé síly (tření, valivý odpor, gravitace na mírných nerovnostech podlahy, atd.). Zmíněné síly vzdalují míč od předpokládaného rovnoměrného přímočarého pohybu. Tyto odchylky jsou zpracovány Kalmanovým filtrem jako šum a Kalmanův filtr tedy reflektuje realitu.
Matici měření nastavíme na

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Kalmanův filtr s takto nastaveným modelem popisuje pohyb míčů, odvozuje rychlost míčů a potlačuje jak náhodný, tak systematický šum. Je dobré dodat, že Kalmanův filtr potřebuje data zatížená pouze šumem s normálním rozdělením. Tento předpoklad na data nemáme. Kalmanův filtr je natolik robustní algoritmus, že v práci podá dobré výsledky, i když teorie nezaručí, že filtrované hodnoty budou konvergovat k reálným hodnotám.[15]

V původním Kalmanově filtru se střídají kroky měření a predikce modelu. V případě, že k danému sledovanému míči nenaleznu měření, vynechám krok měření a pouze predikuji. Výsledkem je, že budeme schopni modelovat chování míče v situacích, kdy ho senzor nevidí. Další následek je ten, že filtr bude zvětšovat nedůvěru ke svému vnitřnímu stavu a ve chvíli, kdy se k filtru začnou přiřazovat nová měření, budou mít tato měření velkou váhu.

2.8 Výstup

Ke každému parciálnímu vstupu, který zpracujeme, máme parciální výstup. Parciální výstup je množina informací o sledovaných míčích. O míčích evidujeme identifikátor, aktuální pozici a rychlost.

3. Implementace

3.1 Volba technologií

Jádrem problému je zpracování dat senzorem. Senzor produkuje velké množství dat, které potřebujeme zpracovávat co nejrychleji. Naše řešení je naimplementováno v jazyce C++. Jazyk jsme zvolili pro jeho rychlost a ruční správu paměti. Pokud alokujeme paměť na vstup, vstup zpracujeme a ihned poté můžeme paměť odalokovat. Díky tomu se budeme pohybovat ve stejné části paměti a je zde velká šance, že data pojmou cache. Tohoto bychom nemohli docílit v jazycích spravujících paměť pomocí garbage-collection. Zavrhli jsme jazyk C, protože cílem práce není low-level optimalizace algoritmu a C++ nám umožní využít objektově orientované programování. Tím se nám zjednoduší návrh aplikace. Jako pomocnou knihovnu k jazyku C++ používáme knihovnu Boost. Boost používáme pro práci s vlákny, vstupem a výstupem, síťovou komunikací a matematickými strukturami.

Pro síťovou komunikaci použijeme protokolu TCP/IP. Samotný senzor po TCP/IP komunikuje, proto je využití tohoto protokolu nutné. Výstup algoritmu zpřístupňujeme také po TCP/IP. To umožní umístit běžící algoritmus na jiný hardware než poběží aplikace, která naše vstupy zpracovává. Pokud budeme chtít průběh algoritmu zobrazovat na jiném zařízení, budeme posílat větší objemy dat (více v sekci Vizualizace). K tomuto účelu TCP/IP také dostačuje svojí rychlostí.

Abychom mohli algoritmus ladit, naimplementujeme jeho vizualizaci. Vizualizaci chceme používat z operačních systémů Windows a Linux. Proto jsme použili přenositelné knihovny OpenGL a knihovny Qt pro práci s grafickým uživatelským rozhraním.

3.2 Podporované operační systémy

Řešení lze vyvíjet i spouštět na operačních systémech Windows a Linux. Windows je zvolen hlavně pro jeho rozšířenost a oblíbenost mezi vývojáři. Jak na Windows, tak na Linuxu lze spouštět jak samotný algoritmus, tak vizualizace. Vizualizace potřebuje podporu okenního manažeru. Samotný algoritmus je vázán technologiemi C++, podporou Windows vláken nebo pthreads[1] a implementací BSD socketů.

V robotice jsou populární systémy ROS a real-time systémy, jmenujme FreeRTOS. Portování algoritmu na ROS lze přímočarou implementaci ROS node. Pro portování na ROS communication infrastructure, musíme přepsat nebo implementovat třídy NetworkClient a NetworkServer. Port na FreeRTOS je obtížnější než port na ROS. Zde nemáme k dispozici BSD sockety ani vlákna. Musíme použít vlastní implementaci TCP/IP stacku.

3.3 Algoritmus

Naimplementovali jsme původní verzi Kalmanova filtru. V této verzi probíhá mnoho operací s maticemi. Pro práci s maticemi používáme knihovnu Boost[2].

Výstup realizujeme po TCP/IP. Ve chvílích, kdy zapracujeme měření senzoru, posíláme zprávu obsahující popis množiny všech aktuálně sledovaných míčů. Ke každému míči posíláme identifikátor, jeho pozici a rychlost.

[1] POSIX threads, URL `http://en.wikipedia.org/wiki/POSIX_Threads` [citace 7. května 2014]

[2] `http://www.boost.org/doc/libs/1_53_0/libs/numeric/ublas/doc/index.htm` [citace 7. května 2014]

3.4 Práce s daty

Připojení k senzoru SICK probíhá po TCP/IP pomocí definovaného protokolu[14]. Data senzor produkuje jako vektor 16bitových celých čísel, znamenajících délku jednotlivých paprsků v mm. My si je převádíme na čísla s plouvoucí desetinnou čárkou o velikosti 64 bitů. Tato volba umožňuje přesnější výpočty algoritmu, za cenu zvětšení výpočetních nároků jednotlivých operací. Při použití hardware, který má FPU[3], výpočet znatelně nezpomalíme.

3.5 Přehrávání dat z logů

Pro účely ladění umožňujeme zpracovávat místo dat ze senzoru data uložená na pevném disku. Data ukládáme jako posloupnosti skenů v jejich původní binární podobě[4]. Tak neukládáme redudantní informace a tím šetříme místo, které data na pevném disku zabírají. Data ukládáme ve formátu, který obsahuje vzdálenostní data i dobu mezi pořízením jednotlivých skenů.

Při real-time zpracování dat ze SICKu si algoritmus o další snímek říká ve chvíli, kdy předchozí data zpracoval. Při přehrání naměřených dat může být aktuální algoritmus[5] pomalejší, než algoritmus, který běžel při zaznamenávání. Tím by mohla nastat situace, že nastal čas zpracovat další sken, ale algoritmus ještě počítá předchozí sken - výstup tedy bude opožděný. Algoritmus by přestal přehrávat data v reálném čase. Možným řešením je zahazovat skeny, které algoritmus nestihne zpracovat včas a tím docílit toho, aby běžel realtime. Stinnou stránkou tohoto řešení je chování různých běhů algoritmu (přehrání dat) - ty mohou dát různé výsledky, což znesnadňuje ladění programu.. Z tohoto důvodu poskytujeme dva způsoby jak

[3]Jednotka pro počítání s plovoucí desetinnou čárkou, URL `http://en.wikipedia.org/wiki/Floating-point_unit` [citováno 7. května 2014]

[4]Skeny jsou senzorem reprezentovány jako posloupnosti 16 bitových hodnot.

[5]V praxi se spíše setkáme s tím, že použijeme stejně rychlý algoritmus na pomalejším HW.

data přehrávat. Přehrání v reálném čase světa a přehrání všech skenů. První způsob zahazuje skeny, které nestíhá zpracovat včas. Druhý způsob zpracuje všechny skeny i za cenu toho, že čas přehrávání nebude odpovídat realitě.

3.6 Vizualizace

Při ladění řídícího softwaru je pro operátora užitečné vidět aktuální vnitřní stav robota. K našemu řešení přidáváme aplikaci schopnou vizualizovat průběh algoritmu. Nazveme ji vizualizační server. K aplikaci je možné se připojit po TCP/IP, aby bylo možné aplikaci spouštět na vzdáleném počítači v době, kdy algoritmus běží na počítači umístěném na robotovi. Další výhodou tohoto přístupu je, že poměrně výpočetně náročná vizualizace nezabírá prostředky programu na sledování míčů. Pro ladění je umožněno nahrávat senzorická data z lokálního souboru.

3.6.1 Síťový protokol vizualizace

Jako základní vysokoúrovňové prostředí pro komunikaci použijeme TCP/IP. Pro práci s touto rodinou protokolů používáme knihovnu Boost.asio[6]. Komunikace se senzorem probíhá pomocí protokolu specifikovaném v manuálu senzoru[14].
Komunikace s vizualizačním serverem probíhá vyměňováním řídících zpráv. Tyto zprávy jsou naimplementovány jako odvozené třídy od třídy Command. Třída Command a odvozené třídy jsou serializovatelné pomocí knihovny Boost.serialization[7]. Odvozené zprávy musí obsahovat v proměnné mLayerName jméno vrstvy určující, v jakém kontextu se zprávy vykonávají. Dále musí mít přetíženou metodu ExecuteCommand, ve které specifikují akci, která se vykoná

[6]Boost.Asio, `http://www.boost.org/doc/libs/1_54_0/doc/html/boost_asio.html` [citováno 7. května 2014]
[7]Boost.serialization, `http://www.boost.org/doc/libs/1_54_0/libs/serialization/doc/index.html` [citováno 7. května 2014]

po jejich zaslání na server.
Samotné zaslání a vykonání zprávy probíhá tak, že na straně klienta vytvoříme instanci třídy odvozené od třídy Command. Tuto instanci serializujeme, pošleme na server, kde ji deserializujeme a zavoláme na ni metodu ExecuteCommand.

3.6.2 Grafický protokol

Vizualizační server otevírá okno v operačním systému. Základním primitivem, které zobrazujeme, jsou úsečky. Z úseček sestrojíme všechny vykreslované objekty. Připojení klienti zasílají zprávy s daty k vykreslení. Kreslí se do vrstev, které jsou identifikovány řetězcem. Tím zajišťujeme identifikaci v případě, že aplikace klienta neočekávaně havaruje a je restartována. Vrstvám nastavujeme barvu. V případě vizualizace skenů senzoru se posílají relativně velké objemy dat. Je proto nutné optimalizovat a dát velký pozor na to, aby se data přenesla právě jednou. Kreslíme pomocí techniky double buffering, abychom předešli problikávání obrazu.
Typy zpráv jsou:

- Započni vykreslování snímku.
- Nastav barvu vrstvy.
- Nakresli úsečku z bodu A do bodu B v dané vrstvě.
- Nakresli sken.[8]
- Ukonči vykreslování snímku.

[8]Tato zpráva lze simulovat pomocí přechozího typu zprávy. Při posílání velkého počtu úseček se zvyšuje režie. Při velkém počtu paprsků skenu je režie již příliš vysoká, jde přibližně o čtyřnásobný nárůst velikosti přenášených dat oproti vykreslování pomocí úseček.

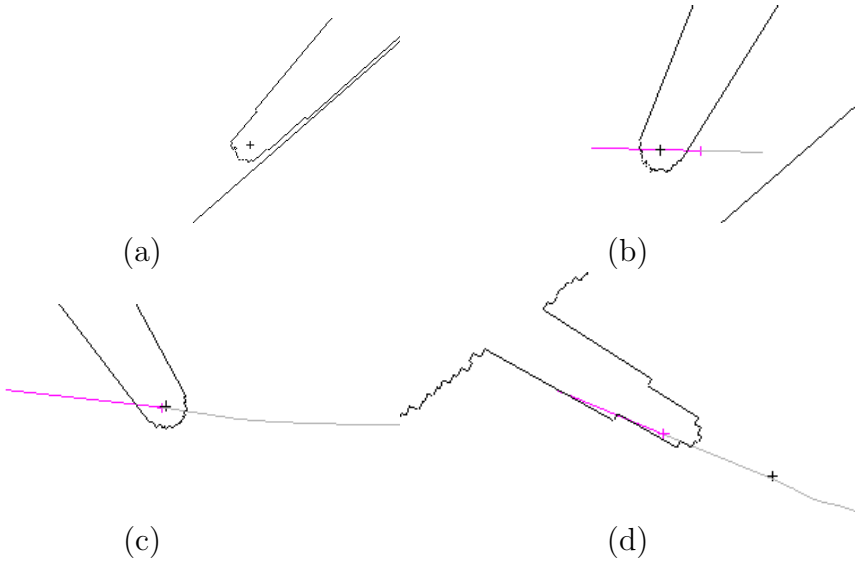
4. Diskuze a budoucí práce

4.1 Modelové situace

Úspěšnost našeho řešení validujeme na reálných situacích. Zaznamenali jsme data typických situací, které nastávají v soutěžním prostředí. V následujících sekcích rozebereme, jak se při těchto specifických situacích chová námi naimplementovaný algoritmus. Testovací data a videa jsou přiloženy na CD (viz Dodatek). Na videu vidíme data ze senzoru znázorněná černou barvou, sledované míče purpurovou barvou (křížek značí pozici a čára vedoucí z křížku vektor rychlosti), hrany ve skenovaných datech modrou barvou a dráhy míčů šedou barvou.

4.1.1 Sledování samostatného míče

Testovací data jsou pojmenována `jeden_mic.log`.



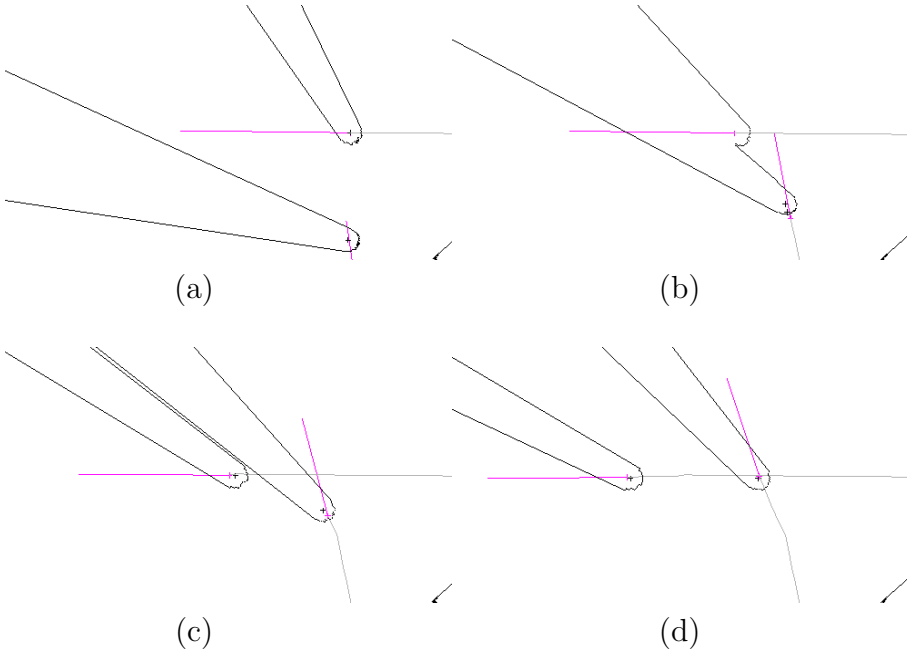
Obrázek 4.1: Průběh situace Sledování samostatného míče.

Jedná se o nejzákladnější situaci. Sledujeme pouze jeden míč v přímočarém pohybu. Můžeme pozorovat, jak se Kalmanův filtr na začátku adaptuje na šum a stabilizuje rychlost a sledovanou pozici na reálných hodnotách (b). To je dáno tím, že v době, kdy

máme pouze jedno měření, nemůžeme nijak efektivně odhadnout rychlost, proto ji nastavíme na 0. Ve chvíli, kdy se míč dostává daleko od senzoru, algoritmus na rozpoznávání míčů začíná mít problém rozeznat v deformovaném skenu míč (d). To kompenzuje Kalmanův filtr, který míč nadále sleduje.

4.1.2 Sledování více objektů beze srážek s překrytím

Testovací data jsou pojmenována `dva_mice_bez_srazky.log`.



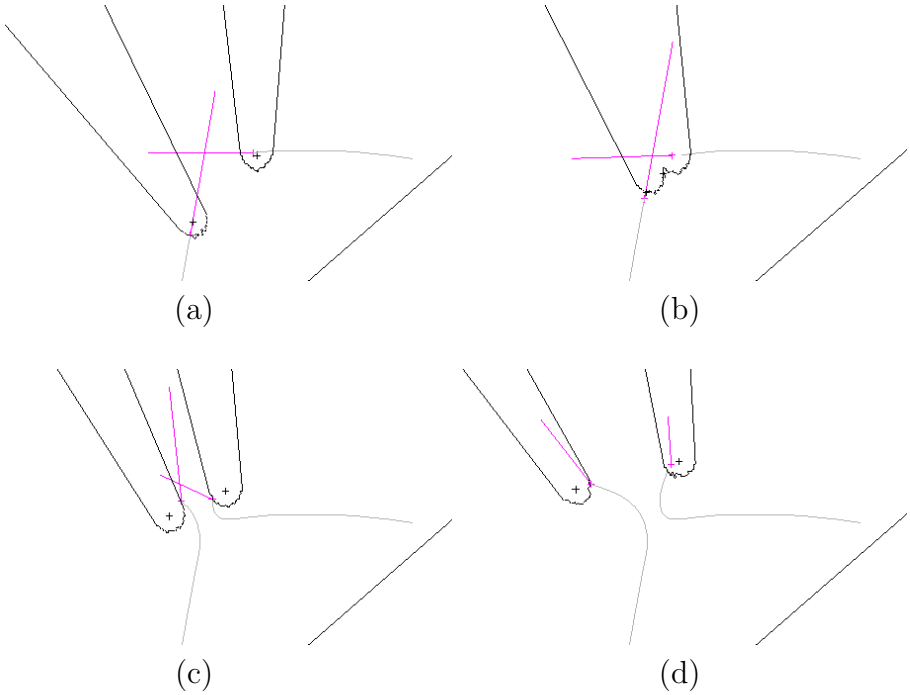
Obrázek 4.2: Průběh situace Sledování více objektů beze srážek s překrytím.

Situace, kdy se míče míjejí, je velmi zajímavá. Míč blíže ke kameře zakryje vzdálenější a algoritmus na rozpoznávání míčů nemůže správně fungovat a nacházet všechny míče (b). Zde pomůže Kalmanův filtr, který odhaduje pohyb míčů jako stále přímočarý. Odhad je přesný, což vidíme v situaci, kdy se míče dostanou ze zákrytu a nově rozpoznávané hodnoty se shodují s odhady (c).
V době, kdy byl míč v zákrytu a probíhal pouze predikční krok Kalmanova filtru, vzrůstal rozptyl měřených proměnných. Poté co míč opustil zákryt, mělo první měření velkou váhu. Tím se model rychle

adaptoval na nový stav. To je důsledek námi zvoleného postupu.

4.1.3 Srážky míčů

Testovací data jsou pojmenována `dva_mice_srazka.log`.

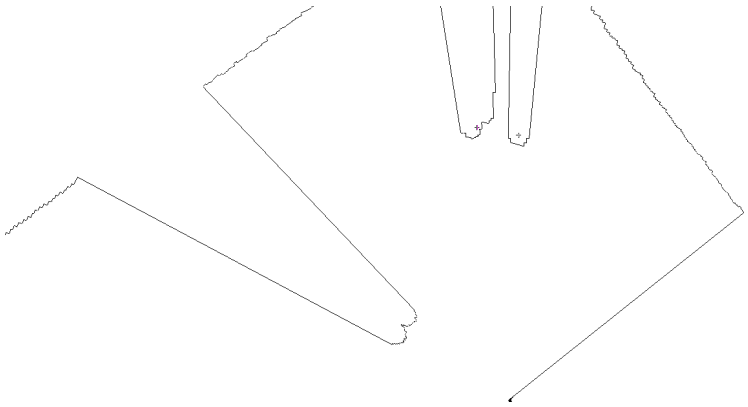


Obrázek 4.3: Průběh situace Srážky míčů.

Ve chvíli kdy do sebe míče narazí (b), stanou se z pohledu senzoru jedním objektem a není je možné rozeznat jako míče. V tu chvíli Kalmanův filtr predikuje pohyb stále ve stejném směru. Zde hrozí, že by si tímto způsobem mohly hodnoty Kalmanových filtrů vyměnit pozice s míči. Ve chvíli, kdy se stíny od sebe oddělí (c), je už detekujeme, nicméně jejich rychlost byla výrazně zdeformována srážkou a vůbec neodpovídá hodnotám v Kalmanově filtru. Pozorujeme, že v příštích pár snímcích se Kalmanův filtr na nové hodnoty adaptuje a míče jsou nadále úspěšně sledovány (d).
Úspěšnost sledování je důsledkem nízké rychlosti míčů a vysoké frekvence snímkování (viz sekce 3.5). Tyto podmínky jsou v námi řešené úloze běžné a často jsme schopni snímkovat daleko rychleji.

4.1.4 Shluky míčů

Testovací data jsou pojmenována `shluky_micu.log`.



Obrázek 4.4: Průběh situace Shluky míčů.

Zde demostrujeme případy, kdy rozpoznávací algoritmus selhává. Míče, opírající se jeden o druhý, vytvářejí shluk. Tento shluk je tak velký, že nelze rozlišit, zda se jedná o dva míče nebo objekt úplně jiné povahy. Konkrétně na těchto datech by šel problém vyřešit tak, že bychom analyzovali tvar objektu. Náš algoritmus byl však navržen, aby hledal pouze míče samotné a proto selhává.

4.2 Budoucí práce

Z analýzy chování algoritmu v modelových situacích plyne, že námi navrhnutý a naimplementovaný algoritmus funguje i v situacích jako je například srážka míčů nebo dočasné zakrytí sledovaného míče. Dovolujeme si navrhnout několik zlepšení, od kterých si slibujeme zlepšení v situacích nad rámec zadání.

Rozpoznávání objektů může být rozšířeno o sofistikovanější algoritmy než je detekce hran. Při detekci můžeme více zohlednit tvar míče. Od toho si slibujeme lepší výsledky při detekování vzdálených míčů a míčů ve shlucích.

Při párování objektů v současnosti používáme pouze informaci o pozici. Použitím informace o jejich rychlosti a rozšířením algoritmu

o zpracování této informace bychom mohli vylepšit výsledky v modelových situacích jako je 4.1.3, kde hrozilo, že se sledované míče vymění.

Softwarová část lze rozšířit do obecnějšího frameworku. Můžeme se zaměřit na optimalizaci rychlosti aplikace. Senzor produkuje velké množství objemných dat, které v současné verzi nestíháme zpracovávat. Současné výsledky jsou pro námi definovanou úlohu plně dostatečné, avšak zvýšením frekvence zpracovaných dat si zvýšíme robustnost v okrajových případech.

Závěr

Za cíl jsme si dali řešit jednu z těžkých úloh robotiky, rozpoznávání a sledování objektů. Rozhodli jsme se vzít specifické podmínky známé robotické soutěže a navrhnout a naimplementovat systém, který by byl aplikovatelný v soutěžním prostředí. Výsledky z naší práce měly také poskytnout podklad pro další rozšíření a zrobustnění algoritmu. Domníváme se, že jsme uspěli a vytyčené cíle jsme splnili.

Vybudovali jsme řešení pro sledování míčů stojících nebo se pohybujících v blízkosti senzoru. Využili jsme faktu, že podmínky úlohy jsou přesně dané konkrétní aplikací. Tím jsme docílili toho, že jsme mohli vyřešit velmi složité problémy běžně spojené s úlohou sledování objektů. Byli jsme dokonce schopni krátkodobě sledovat objekty, které jsou mimo dosah senzoru. Námi navrhnuté řešení se ukázalo jako robustní pro sledování míčů i při jejich přímé interakci. Byla vytvořena implementace a otestována na sadě testovacích dat. Byly identifikovány okrajové případy, které nelze z dat produkovaných použitým senzorem modelovat. Na naší implementaci bylo demonstrováno chování algoritmu v těchto okrajových situacích. U některých z těchto případu byly navrhnuty náhradní postupy jak dosáhnout alespoň částečných úspěchů.

Dodatek

Obsah přiloženého média

K práci je přiloženo CD s následujícím obsahem:

- kořenový adresář: Text práce a průvodní dokument k CD.
- adresář source: Zdrojové soubory,projekt Microsoft Visual Studia 2010 a Makefile.
- adresář binary: Binární spustitelné soubory s naším řešením.
- adresář test_data: Testovací data. Jde o nahrané sekvence dat senzoru SICK, které je možné přehrát v naší aplikaci. Více o datech v kapitole Diskuze a budoucí práce.
- adresář video: Videa vizualizací jednotlivých modelových situací. Videa zachycující výstupy vizualizačního serveru.[1]
- adresář manual: Dokumenty popisující instalaci, používaní a strukturu aplikace.

[1] Videa byla natočena pomocí softwaru CamStudio, URL `http://camstudio.org/` [citováno 7. května 2014]

Literatura

[1] CHERIYADAT, Anil M., Budhendra L. BHADURI a Richard J. RADKE. *Detecting multiple moving objects in crowded environments with coherent motion regions.* 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops. IEEE, s. 1-8. DOI: 10.1109/CVPRW.2008.4562983.

[2] SIEBEL, Nils T. a Stephen J. MAYBANK. *Fusion of Multiple Tracking Algorithms for Robust People Tracking.* 2002 Proceeding ECCV '02 Proceedings of the 7th European Conference on Computer Vision-Part IV, strany 373-387 Springer-Verlag London, UK ISBN:3-540-43748-7

[3] CUI, J., X. SONG, H. ZHAO, H. ZHA a R. SHIBASAKI. *Real-Time Detection and Tracking of Multiple People in Laser Scan Frames.* Augmented Vision Perception in Infrared. London: Springer London, 2009, s. 405. DOI: 10.1007/978-1-84800-277-7_17.

[4] CUI, Jinshi, Hongbin ZHA, Huijing ZHAO a Ryosuke SHIBASAKI. *Multi-modal tracking of people using laser scanners and video camera.* Image and Vision Computing. 2008, svazek. 26, číslo 2, strany 240-252. DOI: 10.1016/j.imavis.2007.05.005.

[5] WELCH, Greg a Gary BISHOP. *An Introduction to the Kalman Filter.* University of North Carolina at Chapel Hill, Department of Computer Science, Chapel Hill, NC, USA. TR95-041. 1995.

[6] X-RAY. *Assignment Problem and Hungarian Algorithm.* [online]. [cit. 7. května 2014]. Dostupné z: `http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=hungarianAlgorithm`

[7] JIANBO, SHI a TOMASI. *Good features to track.* 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94. IEEE Comput. Soc. Press, strany 593-600. DOI: 10.1109/CVPR.1994.323794.

[8] SONKA, Milan, Vaclav HLAVAC a Roger BOYLE. *Image processing, analysis, and machine vision.* 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94. IEEE Comput. Soc. Press, strany 593-600. DOI: 978-0-412-45570-4.

[9] ROSTEN, Edward, Tom DRUMMOND a Roger BOYLE. *Machine Learning for High-Speed Corner Detection.* 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94. IEEE Comput. Soc. Press, strana 430. DOI: 10.1007/11744023_34.

[10] LUCAS, B. D. a T. KANADE. *An iterative image registration technique with an application to stereo vision.* 1981 Proc. of 7th International Joint Conference on Artificial Intelligence, strany 674–679

[11] PARZEN, E. *On estimation of a probability density function and mode.* 1962 Ann. Math. Statistics. 33, s. 1065–1076

[12] KUHN, H. W.. *The hungarian method for the assignment problem.* 1955 Naval Research Logistic Quarterly, IEEE Comput. Soc. Press, svazek 2, 1-2. ISSN 00281441.

[13] THRUN, Sebastian. *Probabilistic robotics.* 2006 Massachusetts: MIT Press, ISBN 02-622-0162-3.

[14] SICK AG WALDKIRCH. *Operating Instructions LMS100/LMS111/LMS120 Laser Measurement Systems with Double-pulse Technology.* 2008 [online]. [cit. 7. května

2014]. Dostupné z: http://www.sick-automation.ru/images/File/pdf/DIV05/LMS100_manual.pdf.

[15] SEONG, Chi-Young, Byung-Du KANG, Jong-Ho KIM a Sang-Kyun KIM. *Effective Detector and Kalman Filter Based Robust Face Tracking System.* Image and Vision Computing. 2008, svazek 26, číslo 2, straba 453. DOI: 10.1007/11949534_45.