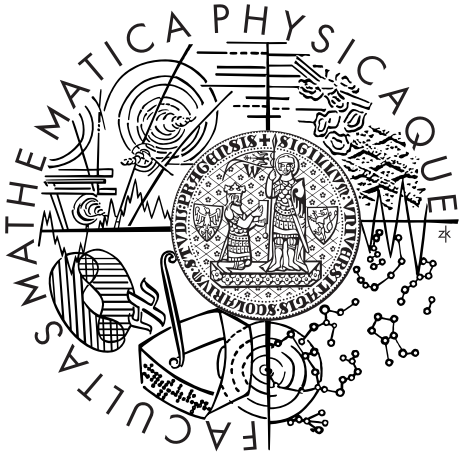
Univerzita Karlova v Praze

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 2013

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 24.7.2013 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 je použití myšlenky Kalmanových filtrů 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 paper focuses on object detection and tracking with the help of laser range finder sensor. The paper investigates other author's way of solving similar problems and analysis the problem as it is. The application of the task of robot competition is chosen as the reference problem which is solved as follows. The possible port solution for the non-PC platforms is emphasized. The basis of the solution is built on the idea of Kalman's filters and the identification of the objects acording to their position and speed.

Keywords: object detection and tracking, rangefinder, Kalman filter

Obsah

Úvod

Rozpoznávání a sledování objektů

Úloha rozpoznávání objektů si dává za cíl vyhledávat v senzorických datech (obrázky, videa, vzdálenostní informace atd.) objekty reálného světa. Sledování bere sérii takovýchto dat v čase a hledá množinu funkcí popisující pohyb jednotlivých objektů v čase. K vyřešení úlohy rozpoznávání a sledování objektů v její obecné poloze potřebujeme vyřešit mnoho velkých problémů umělé inteligence a to není snadné[1]. I přesto se ozývá potřeba sledovat objekty z mnoha odvětvích lidské činosti, jmenujme například v oblast autonomních vozidel, zabezpečovacích zařízení nebo dopravních systémů. Relaxací problému a jeho aplikací na konkrétní situaci lze dosáhnout alespoň parciálních výsledků. Toho se také často využívá a některé z výsledků jsou velmi přesvědčívé (blíže v kapitole 1).

Motivace a cíl

Za cíl si pokládáme prozkoumat úlohu rozpoznávání a sledování objektů. Vezmeme její konkrétní podobu aplikovanou na konkrétní úlohu a experimentálně ji naimplementujeme za použití běžně dostupných prostředků. Výstupem z práce bude jak konkrétní implementace systému řešící úlohu, tak podklady pro teoretické rozebraní problémů, které jsou spojené s obtížností tématu.
Jednou z výzev, které vidíme, je rozšíření a popularizovaní pokročilých odvětví robotiky. Je obtížné proniknout do některých teoretických oblastí pro osoby bez hlubších teoretických znalostí. V robotice lze najít mnoho teoreticky vyřešených úloh nebo implementací, ale pro člověka neznalého problémů jsou první krůčky velmi obtížné.

[1] Názorně demostrováno v sekci 1.2 v [14]

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 jsou některé úlohy velmi rozsáhlé a přechod od méně složitých úloh zahrnuje učení mnoha nových věcí najednou. To je velmi obtížné a může to člověka odradit. Budeme dbát důraz na to, aby naše práce byla přínosná člověka méně zkušeného v námi vybrané oblasti.
Budeme se inspirovat reálnou soutěží[2] a problém podřídíme tomu, abychom docílili řešení relevatnímu při konstrukci robota na tuto soutěž. Tím docílíme toho, abychom se drželi reálného problému. 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 byla soutěži SICK Robot Day 2012 řešena mnoha týmy. Drtivá většina řešení byla implementována naivními algoritmy a výsledky v řadě případů byly nedostatečné[3]. Námi vypracované řešení nabídne možnost zlepšení řídícího softwaru pro roboty konstruované na budoucí robotické soutěže.

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 na referečních situacích rozebereme v kapitole Diskuze a budoucí práce, zde také navrhneme možné zlepšení a rozšíření algoritmu nad rámec zadání.

[2]SICK Robot Day 2012, URL `http://www.sick.com/group/DE/home/pr/events/robot_day/Seiten/Robot_day_2012.aspx` [citováno 30. července 2013]
[3]Subjetivní pozorování autora.

1. Analýza

1.1 Související práce

Automatické sledování objektů je populární obor a existuje mnoho prací na dané téma[1][16][2][4][7]. Žádná práce nemůže tento problém vyřešit obecně (to souvisí s teoretickou hloubkou problému). Populární je zejména zpracování videa pro sledování osob[13], aut[7][2] a jiných objektů. Zpracování videa nám umožnuje pracovat s texturami objektů, ztrácíme však informaci o všech rozměrech jeho tvaru. Obecně můžou na vstup přicházet i jiná než obrazová data. Populární jsou laserové dálkoměrné systémy. Ty nám typicky dávají dobrou informaci o tvaru objektu, nikoliv o jeho textuře[1][9][16][7]. Z dat musíme rozpoznat jednotlivé objekty, téma je dobře rozepsané například v práci [14]. Pokud sledujeme více objektů a chceme udržovat historii pohybu pro každý objekt, musíme je po jejich detekci rozřadit k jejich historiím[5][19][7]. To děláme rozlišováním vlastností objektů. Například.: pozicí a rychlostí [1][9][16] nebo podle jejich textury[13]. Pokud objekty ze vstupních dat nedokážeme identifikovat jednoznačně používají se filtry, oblíbeným je pak Kalmanův filtr a jeho modifikace [8][2][4]. Často nastávají situace, kdy jsou měření ze senzorů nejednoznačná a je potřeba si vést více hypotéz o pozorovaném světě. Pro práci s hypotezámi se používají metody multi hypothesis tracking[10][13].

1.2 Definice řešeného problému

Náš problém je inspirován soutěží SICK Robot Day 2012. V soutěži roboti vozili míče do vlastních domečků. Soutěžilo se na kola a kolo vyhrál ten tým, jehož robot dovezl nejvíce míčů v barvě svého domečku. Každého kola se zúčastnili tři roboti, každý začínal v

domečku na okraji hřiště. Soutěžilo se na rovné ploše oválného tvaru o průměru přibližně 30 metrů. Plocha byla ohraničena 50cm vysokými krabicemi. Na ploše se nacházelo 30 míčů velikosti 22 cm těchto barev: bílá, zelená a žlutá. Míče na začátku každého kola stáli, pohybovali se pouze pokud do nich nějaký robot vrazil. Roboti jsou podle pravidel výrazně větší jak míče a jsou tedy snadno rozlišitelní.

Laserovým dálkovým skenerem budeme snímat okolí robota a ve snímcích rozpoznávat a sledovat míče. Rozpoznávání bere data senzoru a vrací pozice míčů, které v těchto datech nalezne. Sledováním myslíme to, že budeme schopni udržet informaci o tom, který míč je který v průběhu celého procesu. Tím bude schopni kromě jeho pozice učit i jeho rychlost a předvídat jeho pohyb[1]. Senzor pracuje pouze se vzdálenostními daty a z nich dokáže odvodit pouze tvar objektů, jejich barvu zjistit nedokážeme.

1.3 Popis senzoru

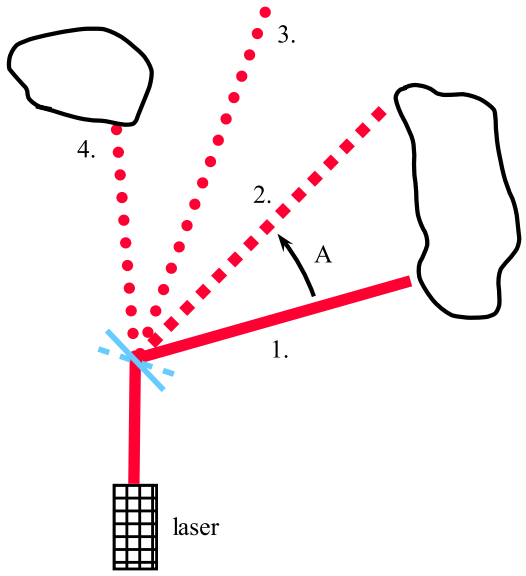
Jako referenční senzor používáme populární senzor SICK LMS 100[2]. Senzor vrhá sérii laserových paprsků, které měří vzdálenosti od senzoru (názorněji na obrázku 1.1). Tyto paprsky léží na jedné rovině a kromě informace o vzdálenosti vrací ještě informaci o odrazivosti porvchu, na který dopadly. My budeme používat informaci o vzdálenosti, pomocí které budeme pozorovat okolí senzoru.

Senzor komunikuje pomocí protokolu TCP/IP. Některé technické parametry relevantní našemu bádání můžeme najít v tabulce 1.1. Laserový senzor umístíme do výšky poloviny míče a budeme pořizovat vodorovné scany hřiště. Scany, které bude senzor produkovat, tedy budou zachycovat míče, stěny i ostatní roboty1.2.

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

[2] Specifikace senzoru SICK LMS 100, URL `https://www.mysick.com/PDF/Create.aspx?ProductID=33753&Culture=en-US` [citováno 30. července 2013]

Senzor velkou mírou ovlivňuje řešení problému, proto ho uvádíme již v analýze.



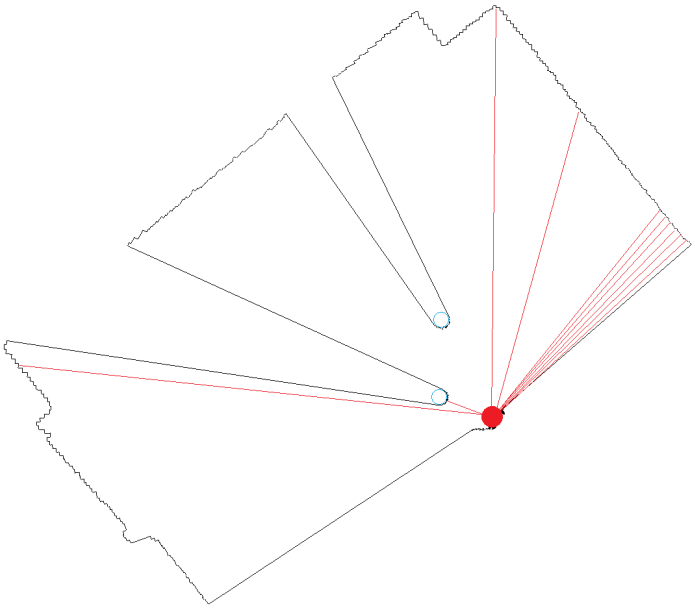
Obrázek 1.1: Schéma laserového dálkoměrného senzoru. Laser vypouští paprsek, který se odráží od rotujícího zrcadla (modrá plná čára), poté se odrazí od nejbližšího objektu a vrací se stejnou cestou zpátky k laseru, kde je zachycen. Senzor měří dobu letu paprku a pomocí této informace a rychlosti letu laserového paprku dopočítává délku paprsku. Zrcadlo se otáčí o úhel A (přerušovaná modrá čára) a proces se opakuje.

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.

Tabulka 1.1: Technické paramtery senzoru SICK LMS 100

1.4 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 tesné blízkosti bez kolize - zde nezjistíme, ke které ze situací došlo). V jednoduchých připadech,



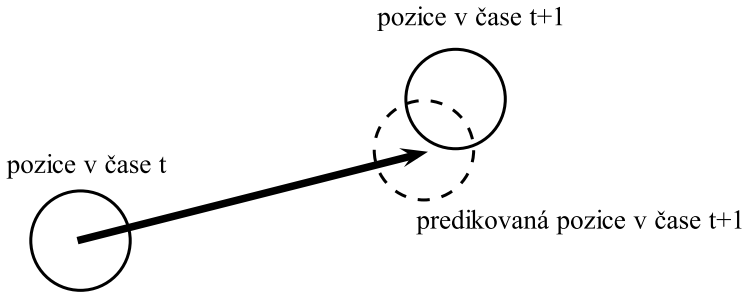
Obrázek 1.2: Vizualizace skenu místnosti s dvěma míčí. Červený plný kruh značí pozici senzoru. Pro názornost jsou naznačeny některé paprsky jako červené čáry. Paprsky jsou ve skutečnosti rozmístěny rovnoměrně po pár desetinách stupně. Modré kružnice naznačují pozice míčů. Obdélníkový útvar je půdorys místosti 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.

kdy to lze (mimochodem v těch v praxi nejčastějších) míč sledovat budeme. Ve vzdálenosti od senzorů klesá přesnost senzoru a také užitečnost informace o sledovaném objektu[3]. To zvyšuje nárok na rozpoznávání míče ze skenu. Budeme tolerovat, pokud informace o vzdálených objektech bude nepřesná (robot typicky potřebuje pracovat s blízkými míči, tak to tolik nevadí).

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 míče1.3. 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.

Nesmíme zapomínat ani na šum senzoru. Měření nejvíce zkreslují odlesky paprsků od meřeného objektu a srážky paprsků laserového

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



Obrázek 1.3: Za předpokladu, že naše predikce je správná, můžeme míče identifikovat, podle toho, že predikovaná a reálná pozice je blízká.

senzoru s nečistotami ve vzduchu. V našem případě pracujeme a v relativně čistém prostředí a míče paprsky soustavně neodrážejí. Z náhodných odlesků se můžeme zotavit, použijeme-li pozorování, že odlesků není mnoho a tedy paprsek není výrazně jinak dlouhý než oba jeho sousedi.

1.5 Prostředí práce

Abychom vytvořili řešení, které bude dostupné i pro mladé robotické týmy s malým rozpočtem, pokusíme se minimalizovat cenu. Na výpočet použijeme bežně dostupný levný notebook s ne příliš výkoným procesorem [4] a pamětí 3GB a operačním systémem Windows. Výhodou tohoto řešení je relativně vysoká výpočetní kapacita se zachováním komfortu PC platformy (TCP/IP, OS), snadnou dostupností a možnosti připojení na robota, pokud použijeme notebook menší velikosti. Nelze ingorovat ani komfort vzniklý pro vývoj řešení, který nám poskytuje operační systém.

[4]Specifikace procesoru, URL `http://ark.intel.com/products/42109/Intel-Core2-Duo-Processor-T6670-2M-Cache-2_20-GHz-800-MHz-FSB` [citováno 30. července 2013]

2. Návrh

2.1 Algoritmus

Zde si popíšeme algoritmus na vyšší úrovni abstrakce. Realizovanou implemenatci popíšeme v kapitole 3.

2.1.1 Vstupy

Na vstupu máme vzdálenostní data ze senzoru. Všechna měření nemáme k dispozici v jednom okamžiku, ale dostáváme je 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, tím blíže realitě jsou výstupy (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.

2.1.2 Jádro algoritmu

Sledování míčů provádíme pomocí Kalmanových filtrů. Celý proces vypadá následovně:

1. Načteme data ze senzoru.
2. Provedeme preprocessing dat (zbavíme šumu).
3. Vyhledáme pozice jednotlivých míčů.

[1] Online algoritmus, URL`http://en.wikipedia.org/wiki/Online_algorithm`[citováno 30. července 2013]

4. Jednotlivé nalezené pozice míčů, přiřadíme k sledovaným míčům Kalmanovy filtry.

5. Updatujeme Kalmanovy filtry o naměřené pozice míčů.

6. Informujeme o nových pozicích.

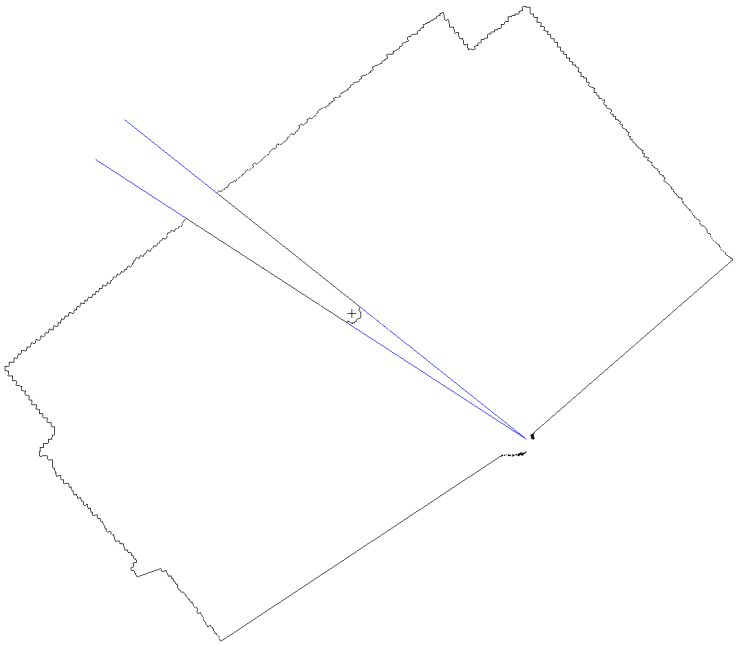
Následuje popis jednotlivých kroků trochu podrobněji.

2.1.2.1 Preprocessing scanů

Scany jsou zatíženy šumem. Šum zavisí 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. Nahodné odlesky od objektů v našem prostředí jsou charakteristické v tím, že nijak nezávisí na odlescích okolních paprsků. Slovy pravděpodobnosti: jevy, že se odlesknou dva paprsky v jednom scanu jsou nezávislé. Druhým typem šumu, se kterým se potýkáme, jsou soustavné nepřesnosti měření laser range finderu. Ty se více pojevují vzdáleně od senzoru, kde není potřeba mít přesné výsledky.
Potlačujeme pouze odlesky: Pro každou trojici sousedních paprsků sledujeme, zda prostřední není kratší nebo delší než oba krajní a pokud ano, tak jej nastavíme na velikost aritmetického průměru délky okrajových paprsků.

2.1.2.2 Vyhledávání míčů

Z definice problému jsou míče v prostoru převážně osamocené.
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[14]. To uděláme tak, že bereme sken jako



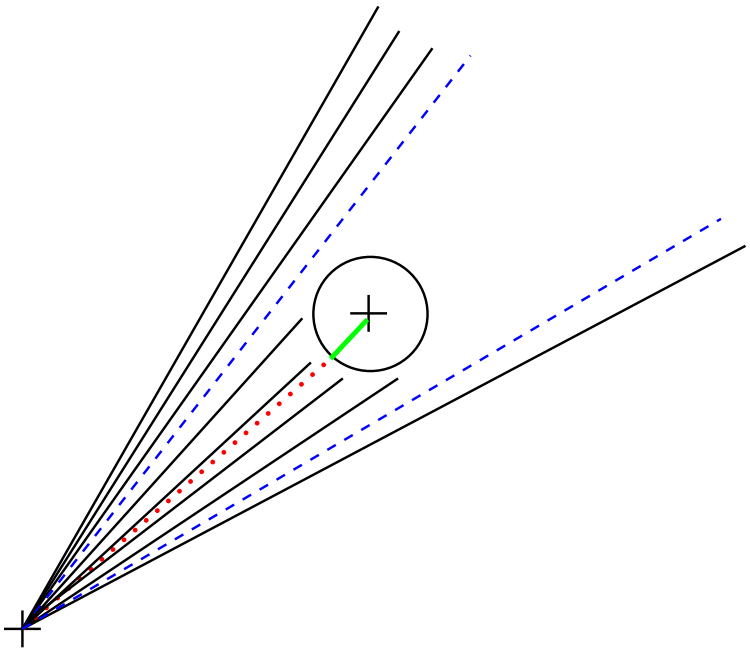
Obrázek 2.1: Hledání míčů ve scanu (modré čáry znázorňují hrany a černý křížek střed nalezeného míče)

posloupnost čísel a nalezáme velké (stanovíme konstantu dvou velikostí míče) hrany mezi sousedními čísly. Hrany jsou pro názornost zobrazeny na obrázku 2.1 jako modré paprsky. Dva po sobě jdoucí skoky 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 (například kulatost míče). Tyto způsoby by měly přinášet přesnější výsledky[3], avšak následně použité Kalmanovy filtry jsou dostatečně robustní, že výsledky jsou postačující.

Námi navržený algoritmus nedetekuje míče nalepené na sobě. Takto uspořádané míče detekovat nelze, protože detekujeme pouze míče a takovéto objekty se na scanu jeví jako objekty jiné třídy (v extrémním případě míče vyrovnané v řadě vedle sebe mohou působit jako

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

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



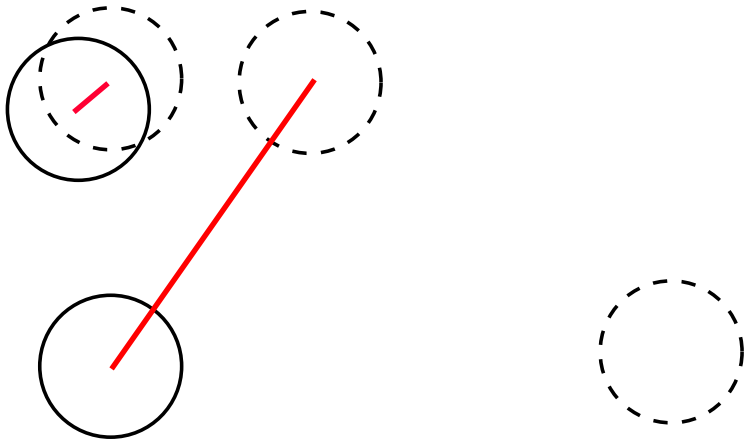
Obrázek 2.2: Dopočítání středu míčů ze stínů. Modré přerušované paprsky jsou detekované hrany scanu. Červený tečkovaný papsek je osou paprsků modrých a je dlouhý jako nejmenší z paprsků mezi hranami.

zeď). To je situace, která nenastane často (díky nízké hustotě míčů).

2.1.2.3 Párování měření k objetům

Na vstupu máme množinu měření a množinu sledovaných míčů. Výstupem toho kroku budou tři množiny: přiřazení měření k sledovaným hodnotám, lichá měření a liché sledované míče.

Přiřazení nově naměřených míčů k již sledovaným hodnotám převádíme na optimalizační úlohu. Vezmeme si vzdálenosti všech naměřených pozic míčů od všech sledovaných míčů. Hledáme přiřazení jednotlivých měření k sledovaným míčům takové, že součet vzdáleností mezi přiřazenými je co nejmenší. Vzdáleností myslíme Euklidovskou metriku mezi středy míčů. Toto je zadání optimalizační úlohy, kterou řešíme Maďarským algoritmem (podrobněji v 2.1.2.3). Předpoklad pro Maďarský algoritmus je, že měření bude stejný počet jako sledovaných objektů. My však můžeme mít rozdílný počet měření a sledovaných míčů. Předpokladu docílíme takto: pokud měření, resp. sledovaných míčů je méně, tak je doplníme o chybějící



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

počet virtuálních měření, resp. míčů. Vzdálenosti mezi virtuálními objekty a objekty měření/sledování nastavíme na konstantní hodnotu. Zvolíme 0. Námi přidané virtuální hodnoty výsledek nezmění. Ukažme si proč. Nechť máme množinu párů a lichý objekt. Přidáme virtuální objekt, který se napáruje na některý z párů (toto je chování, které nechceme). Poté součet vzdáleností nově napárovaných hodnot je větší jak součet v situaci, kdy se lichý objekt páruje s virtuálním. To je protože vzdálenost virtuálního objektu s jakýmkoliv je konstantní a vzdálenost lichého míče od míče z páru je větší jak vzdálenost míčů v páru. To plyne z toho jak jsme si definovali, že budeme míče k sobě přiřazovat (viz obrázek 1.3). 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 liché.
- Páry, které jsou od sebe vzdálené více než o zvolenou konstantu označíme za liché. Konstantu volíme tak, aby nebyla

neoznačila páry, které k sobě náleží a jsou díky šumu o trochu posunuté. Zvolme ji rovnou velikosti dvou průměrů míče. Názorně na obrázku 2.3.

- Všechny zbylé páry k sobě náleží.

Maďarský algoritmus Maďarský algoritmus (z anglického hungarian algorithm, někdy Kuhn–Munkres algorithm nebo Munkres assignment algorithm[4]) budeme používat v situaci, kdy získáme množinu nových měření a budeme je chtít přiřadit k sledovaným objektům.
Jedná se o optimalizační algoritmus. Minimalizujeme perfektní párování na úplném symetrickém bipartitním grafu se stejně velkými partitami. K nalezení takového párování používá metodu zlepšujících cest v grafu. Algoritmus nalezneme například v [6].

2.1.2.4 Sledování v čase

Měřené pozice míčů filtrujeme Kalmanovým filtrem. Kalmanův filtr budeme používat na redukci šumu při sledování objektu a získání hodnot skrytých proměnných z modelu (máme sérii měření pozic v čase a chceme z nich odvodit rychlosti při jednotlivých měřeních). Použijeme nemodifkovaný Kalmanův filtr a model volíme takto:
Stav modelu

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

[4]Maďarský algoritmus, URL `http://en.wikipedia.org/wiki/Hungarian_algorithm` [citováno 30. července 2013]

kde x, y jsou souřadnice pozice a vel_x, vel_y je rychlost. Předchodovou 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. Ukážeme si proč jsme tak zvolili: Rozepíšeme si přechod z času t na čas $t+1$ proměné 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 rozepsaní 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. Tyto rovnice popisují rovnoměrný přímočarý pohyb. To je pohyb, který míče v soutěži nejčastěji mají a tedy tento matematický model vystihuje naší situaci. V reálu však na míče působí různé malé síly (tření, valivý odpor, grativace na mírných nerovnostech podlahy atd.). Tyto 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

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

Nastavování modelu pro sledování objektů například v [11].
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 se však adaptuje na šum natolik dobře, že podává dobré výsledky i takto.

Kalmanův filtr Okolo Kalmanova filtru a jeho modifikací je toho popsáno mnoho (k hlubšímu prostudování můžeme doporučit například [15][18][3]).
Z pohledu naší úlohy je zajímavé, že Kalmanův filtr filtruje šum v datech. Pokud má šum normální rozdělení, filtr konverguje ke správné hodnotě. Filtru musíme dodat matematický model chování měřených proměnných.
Algoritmus pracuje v krocích. V každém kroku si drží vnitřní stav reprezentace světa a také směrodatnou odchylku těchto hodnot. Tato odchylka zachycuje to, jak filtr ,,věří“ aktualnímu vnitřnímu stavu filtru. Střídají se dva typy kroků: predikce a měření. Při predikci filtr odhadne budoucí vnitřní stav na základě matematického modelu dodaného filtru. Při měření se koriguje vnitřní stav filtru podle hodnot ze senzoru.

2.1.3 Výstupy

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

3. Implementace

3.1 Faktory ovlivňující implementaci

Při ladění řídícího softwaru autonomního robota je velmi užitečné ,,vidět“ co si robot aktuálně myslí. K našemu řešení přidáváme aplikaci schopnou vizualizace. Nazveme ji vizualizační server. K aplikaci se bude možné připojit po TCP/IP, aby bylo možné spustit aplikaci na vzdáleném počítačí v době, kdy algoritmus běží na počítači přidělaném 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.
Naše řešení je naimplementováno v jazyce C++. Jazyk jsme zvolili pro jeho rychlost, ruční správu paměti a pro možnost využít objektově orientované programování. Tím se nám zjednoduší návrh aplikace. Pro síťovou komunikaci používáme knihovnu Boost. Vizualizační server používá knihovny OpenGL a knihovny Qt pro práci s okny. Tyto volby nám dávájí možnost snadné přenositelnosti mezi různými systémy. Například systémem Windows a systémem Linux.

3.2 Algoritmus

Algoritmus je naimplementován přímo podle návrhu z kapitoly 2. Naimplementovali jsme původní verzi Kalmanova filtru. Pro prácí s maticemi používáme knihovnu Boost[1]. V původním Kalmanově filtru se střídají kroky měření a predikce modelu. V případě, že k danému sledovanému míči nenaleznu meření, vynechám krok měření a pouze predikuji. To má za následek to, že budeme schopni mode-

[1] `http://www.boost.org/doc/libs/1_53_0/libs/numeric/ublas/doc/index.htm` [citace 30. července 2013]

lovat 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 tyto měření velkou váhu.

3.3 Práce s daty

Připojení k senzoru SICK probíhá po TCP/IP pomocí definovaného protokolu[2]. 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 64bit. Tato volba umožnuje přesnější výpočty algoritmu za cenu zvětšení výpočetních nároků jednotlivých operací.
Přehrávaná data nezaznamenáváme rychlostí jakou senzor zvládá data načítat, ale zvolíme si frekvenci 10Hz. Tuto konstantu jsme zvolili tak, aby algoritmus na našem hardwaru stíhal data zpracovávat. Při real-time zpracování dat ze SICKu si algoritmus o další snímek říká ve chvíli, kdy předchozí data zpracoval. Pokud bychom data nahrávali stejnou rychlosti hrozilo by, že při zpracování naměřených dat bude aktuální algoritmus pomalejší než algoritmus, který běžel při zaznamenávání. Tím by mohla nastat situace, že rychlost zpracování dat nebude odpovídat realitě a výstup by byl opožděný, algoritmus by přestal být real-time. To obecně vyřešit nejde. Tím, že jsme zvolili rychlost přehrávání dat na dobu za kterou stihne běžný hardware zpracovat data, docílíme toho, že čas se kterým pracuje algoritmus při přehrávání bude přibližně odpovídat času našeho světa.

[2] Operating Instructions LMS100/111/120, URL `http://www.sick-automation.ru/images/File/pdf/DIV05/LMS100_manual.pdf`[citováno 30. července 2013]

3.4 Vizualizace

Vizualizační server je schopný přijímat instrukce od aplikace řešící algoritmus. Instrukce popisují, jaká data má vizualizovat.

3.4.1 Síťový protokol vizualizace

Jako základní vysokoúrovňové prostředí pro komunikaci volíme TCP /IP. Pro práci s touto rodinou protokolů používáme knihovnu Boost.asio[3]. Komunikace se senzorem probíhá pomocí protokolu specifikovaném v manuálu senzoru[12].
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[4]. Odvozené zprávy musí obsahovat v proměnné mLayerName jméno vrstvy v kontextu, které se budou vykonávat a musí mít přetíženou metodu ExecuteCommand, ve které specifikují akci, která se vykoná po jejich zaslání na server.
Samotné zaslání a vykonání zprávy probíhá tak, že na straně klienta vytvoříme instanci třídy odvozené od Command. Tuto instaci serializujeme, pošleme na server, kde ji deserializujeme a zavoláme na ni metodu ExecuteCommand.

3.4.2 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

[3]Boost.Asio, URL `http://www.boost.org/doc/libs/1_54_0/doc/html/boost_asio.html`[citováno 30. července 2013]
[4]Boost.serialization, URL`http://www.boost.org/doc/libs/1_54_0/libs/serialization/doc/index.html`[citováno 30. července 2013]

řetezcem. Tím zajištujeme identifikaci v případě, že aplikace klienta neočekávaně havaruje a je restartována. Vrstvám nastavujeme barvu. V případě vizualizace scanů 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.
- Nastva barvu vrstvy.
- Nakresli úsečku z bodu A do bodu B v dané vrstvě.
- Nakresli sken. (Tato zpráva lze simulovat pomocí přechozího typu zprávy, avšak pokud pokud vykreslujeme čáry po jedné, režie je příliš vysoká.)
- Ukonči vykreslování snímku.

3.5 Výstup

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

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 v chová při těchto specifických situacích 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 miče

Testovací data jsou pojmenovány „jeden_mic.log“.
Jedná se o nejzákladnější situaci. Sledujeme pouze jeden míč v přímočarém pohybu. Můžeme pozorovat, jak se Kalmanův filtr po začátku adaptuje na šum a stabilizuje rychlost a sledovanou pozici na reálných hodnotách. To je dáno tím, že v době, kdy máme pouze jedno měření nemůžeme nijak efektivně odhadnout rychlost, tak ji nastavíme na 0. Ve chvíli, kdy se míč dostává daleko od senzoru, algoritmus na rozpoznávání míčů začíná mít problém rozeznat v deformovaném scanu míč. 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“.
Situace, kdy se míče míjejí, je velmi zajímavá. Míč blíže ke kameře zakryje vzdálenější a algoritmus na rozpoznávání míčů nemůže sprá-

vně fungovat a nacházet všechny míče. Zde pomůže Kalmanův filtr, který odhaduje pohyb míčů jako stále přímočarý. Odhad je přesný, což vídíme v situaci, kdy se míče dostanou ze zákrytu a nově rozpoznávané hodnoty se shodují s odhady.
V době, kdy byl míč v zákrytu a probíhal pouze predikční krok Kalmanova filtru, vzrůstal rozptyl měřených proměnných. Poté co k filtru bylo přiřazeno první měření, mělo toto měření velkou váhu a rychle adaptovalo model na nový stav. To je příjemný důsledek námi zvoleného postupu.

4.1.3 Srážky míčů

Testovací data jsou pojmenována ,,dva_mice_srazka.log“.
Ve chvíli kdy do sebe míče narazí, 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 mohli hodnoty Kalmanových filtrů vyměnit pozice s míči. Ve chvíli, kdy se stíny od sebe oddělí, 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.
Důvodem, proč se sledování nepokazilo tím, že by sledované hodnoty mezi míči přeskočili, je ten, že míče nejely velikou rychlostí a snímkujeme je rychlostí 10Hz (viz sekce 3.3). 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“.
Zde demostrujeme případy, kdy rozpoznávací algoritmus selhává. Míče opírající se jeden o druhý vytvařejí takový obraz na scanu, že

nelze rozlišit, zda se jedná o dva míče nebo objekt úplně jiné povahy. Konkrétně na těchto konkrétních datech by šel problém vyřešit přidáním specifických podmínek, ale náš algoritmus byl navrhnut, tak aby hledal pouze míče samotné.

4.2 Budoucí práce

Z analýzy chování algoritmu v modelových situacích plyne, že námi navrhnutý a naimplementovaný algoritmus funguje i v nestandartních situacích jako je například srážka míčů. Tyto situace v soutěžním porstředí nastávají velmi zřídka. 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, v budoucnu chceme zapojit i jejich rychlost. Domníváme se, že to vylepší výsledky v modelový situacích jako je 4.1.3, kde hrozilo, že se sledované míče vymění.

V situacích, kdy míč sledujeme, ale nemáme žádné nové měření chceme zapojit více informaní ze skenu. Například, pokud sledovaný míč projíždí oblastí, která je pokryta paprsky, logicky tam míč být nemůže a můžeme tedy snížit věrohodnost vzorku.

Softwarová část lze rozšířit do obecnějšího frameworku. Algoritmus na rozpoznávání objektů by měl jít měnit, tak jako model sledování. To zajistí větší rozšířitelnost do nových aplikací. Framework může podporovat více komunikačních protokolů. Celkově je zde spousta místa pro optimalizaci rychlosti aplikace. Senzor produkuje velké množství objemných dat, které v současné verzi nestíháme zpracovávat. Současné vysledky jsou pro námi definouvanou úlohu plně

dostatečné, avšak od zvýšení frekvence si slibujeme větší robustnost v okrajových případech.

Závěr

Za cíl jsme si dali řešit jednu z težkých úloh robotiky. Rozhodli jsme se vzít specifické podmínky známé robotické soutěže a navrhnout a naimplemetovat 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 splnily.

Vybudovali jsme řešení pro sledování míčů stojících nebo pohybujících se v blízkosti senzoru. Využili jsme toho, že podmínky úlohy jsou přesně dané konkrétní aplikací. Tím jsme docíli toho, že jsme mohli vyřešit velmi složité problémy bežně spojené s úlohou sledování objektů. Byli jsme dokonce schopni krátkodobě sledovat objekty, které jsou mimo dosah senzoru. Námi navrhnuté řešení se ukázalo robustní pro sledování míčů i při jejich přímé interakci. Byla vytvořena přímá implementace a otestována na sadě testovacích dat. Byly identifikovány okrajové případy, které nelze z dat produkováných použitým senzorem obecně vyřešit a tyto případy byly demostrovany na naší implemetaci. U některých z těchto případu byly navrhnuty náhradní postupy jak dosáhnout alespoň částečných úspěchů.

Dodatek

Obsah přiloženého média\emph{

K práci je přiloženo CD s následujícím obsahem:

- kořenový adresář: Text práce a průvodní dokument k CD.
- adresář source: Zdrojové soubory a projekt Microsoft Visual Studia 2010.
- 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. 4.
- adresář video: Videa vizualizací jednotlivých modelových situací. Videa zachycující výstupy vizualizačního serveru. Videa byla natočena pomocí softwaru CamStudio[1].
- adresář manual: Dokumenty popisující instalaci, používaní a strukturu aplikace.

[1] CamStudio, URL `http://camstudio.org/` [citováno 30. července 2013]

Seznam použité literatury

[1] 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: Algorithms and 405 Applied Systems,Advances in Pattern Recognition. Springer-Verlag London Limited 2009. DOI: 10.1007/978-1-84800-277-717.

[2] GUERRERO-GÓMEZ-OLMEDO, Ricardo, Roberto J. LÓPEZ-SASTRE, Saturnino MALDONADO-BASCÓN a Antonio FERNÁNDEZ-CABALLERO. *Vehicle Tracking by Simultaneous Detection and Viewpoint Estimation.* IWINAC 2013: Part II. Springer-Verlag Berlin Heidelberg 2013, LNCS 7931.

[3] HU, Congwei, Wu CHEN, Yongqi CHEN a Dajie LIU. *Adaptive Kalman Filtering for Vehicle Navigation*. Journal of Global Positioning Systems. 2003, č. 2.

[4] KARAVASILIS, Vasileios, Christophoros NIKOU a Aristidis LIKAS. *Visual Tracking by Adaptive Kalman Filtering and Mean Shift*. SETN 2010, LNAI 6040. Springer-Verlag Berlin Heidelberg 2010.

[5] MA, Weizhang, Bo MA a Xueliang ZHAN. *Kalman Particle PHD Filter for Multi-target Visual Tracking*. IScIDE 2011, LNCS 7202. Springer-Verlag Berlin Heidelberg 2012.

[6] X-RAY. *Assignment Problem and Hungarian Algorithm* [online]. [cit. 30. července 2013]. Dostupné z: `http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=hungarianAlgorithm`

[7] PETROVSKAYA, Anna a Sebastian THRUN. *Model based vehicle detection and tracking for autonomous urban driving.* Autonomous Robots. 2009, vol. 26, 2-3, s. 123-139. DOI: 10.1007/s10514-009-9115-1. Dostupné z: http://link.springer.com/10.1007/s10514-009-9115-1

[8] PNEVMATIKAKIS, Aristodemos a Lazaros POLYMENAKOS. *Kalman Tracking with Target Feedback on Adaptive Background Learning.* MLMI 2006, LNCS 4299. Springer-Verlag Berlin Heidelberg 2006.

[9] DE PONTE MÜLLER, Fabian, Luis Martín NAVAJAS a Thomas STRANG. *Characterization of a Laser Scanner Sensor for the Use as a Reference System in Vehicular Relative Positioning.* Nets4Cars/Nets4Trains 2013, LNCS 7865. Springer-Verlag Berlin Heidelberg 2013.

[10] REZATOFIGHI, Seyed Hamid, Stephen GOULD, Ba-Ngu VO, Katarina MELE, William E. HUGHES a Richard HARTLEY.*A Multiple Model Probability Hypothesis Density Tracker for Time-Lapse Cell Microscopy Sequences.* IPMI 2013, LNCS 7917. Springer-Verlag Berlin Heidelberg 2013.

[11] SEONG, Chi-Young, Byung-Du KANG, Jong-Ho KIM a Sang-Kyun KIM. *Effective Detector and Kalman Filter Based Robust Face Tracking System.* PSIVT 2006, LNCS 4319. Springer-Verlag Berlin Heidelberg 2006.

[12] SICK AG WALDKIRCH. *Operating Instructions LMS100/LMS111/LMS120 Laser Measurement Systems with Double-pulse Technology.* [online]. [cit. 30.července 2013]. Dostupné z: `http://www.sick-automation.ru/images/File/pdf/DIV05/LMS100_manual.pdf` 2008.

[13] SIEBEL, Nils T. a Steve MAYBANK. *Fusion of Multiple Tracking Algorithms for Robust People Tracking*. ECCV 2002, LNCS 2353. 2002.

[14] ŠONKA, Milan, Václav HLAVÁČ a Roger BOYLE. *Image processing, analysis, and machine vision*. 3rd ed. Toronto: Thomson, 2008, xxv, 829 s. ISBN 978-0-495-08252-1.

[15] THRUN, Sebastian. *Probabilistic robotics*. Massachusetts: MIT Press, c2006, xx, 647 s. ISBN 02-622-0162-3.

[16] TSOKAS, Nicolas A. a Kostas J. KYRIAKOPOULOS. *Multi-robot multiple hypothesis tracking for pedestrian tracking*. Autonomous Robots. 2012, vol. 32, issue 1, s. 63-79. DOI: 10.1007/s10514-011-9259-7. Dostupné z: `http://link.springer.com/10.1007/s10514-011-9259-7`

[17] WANG, Yuxia a Qingjie ZHAO. *Robust object tracking via online Principal Component–Canonical Correlation Analysis (P3CA)*. Signal, Image and Video Processing. DOI: 10.1007/s11760-013-0430-9. Dostupné z: http://link.springer.com/10.1007/s11760-013-0430-9

[18] 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.

[19] ZAMIR, Amir Roshan, Afshin DEHGHAN a Mubarak SHAH. GMCP-Tracker: Global Multi-object Tracking Using Generalized Minimum Clique Graphs. ECCV 2012, Part II, LNCS 7573. Springer-Verlag Berlin Heidelberg 2012.

Seznam tabulek