VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

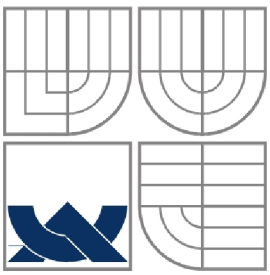
IMPLEMENTACE ROBOTICKÉHO VYSAVAČE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

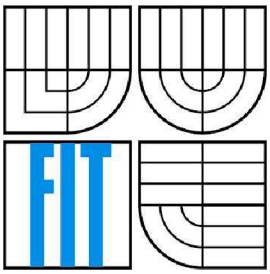
AUTOR PRÁCE
AUTHOR

STANISLAV SOJKA

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

IMPLEMENTACE ROBOTICKÉHO VYSAVAČE

IMPLEMENTATION OF THE ROBOTIC VACUUM CLEANER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — STANISLAV SOJKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAROSLAV ROZMAN
SUPERVISOR

BRNO 2011

Abstrakt

Tato bakalářská práce popisuje algoritmy a způsoby řízení robotických vysavačů v neznámém prostředí. Jsou popsány principy jednotlivých metod a jejich omezení. Práce také popisuje způsob implementace důležitých úseků řídících algoritmů. V nemalé míře se práce zabývá způsobem řešení problémů s lokalizací, které se vyskytly při vytváření řídícího algoritmu pro reálného robota. Výsledkem práce jsou dvě aplikace, jedna pro vzdálené řízení robota podle popsaných algoritmů, druhá pro jejich demonstraci bez nutnosti vlastnit robota.

Abstract

This bachelor thesis describes the algorithms and methods of control of robotic vacuum cleaners in an unknown environment. It describes the principles of methods and their limitations. The thesis also describes the implementation of important sections of control algorithms. The work examines problems with localization that have occurred in the control algorithm on creating a real robot. The results of the work are two applications, one for remote control robot according to these algorithms, the second for their demonstration without owning a robot.

Klíčová slova

Robotický vysavač, pokrytí prostoru, buňka, dekompozice na buňky, Boustrophedon dekompozice, Lichoběžníková dekompozice, Dekompozice řezem, lokalizace, detekce překážek.

Keywords

Robotic vacuum cleaner, coverage area, cell, cell decomposition, Boustrophedon decomposition, Trapezoidal decomposition, Slice decomposition, location, detection of obstacles.

Citace

Sojka Stanislav: Implementace robotického vysavače, bakalářská práce, Brno, FIT VUT v Brně, 2011

Implementace robotického vysavače

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jaroslava Rozmana. Další informace mi poskytli Ing. Tomáš Novotný a Ing. Petr Dittrich. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Stanislav Sojka
18. května 2011

Poděkování

Chtěl bych poděkovat Ing. Jaroslavu Rozmanovi za jeho odborné vedení, zapůjčení literatury a pomoc při řešení problémů. Dále bych chtěl poděkovat Ing. Tomáši Novotnému za přípravu robota, pomoc při jeho zprovoznění a rady při implementaci algoritmů pro reálné odzkoušení. Poděkování patří i Ing. Petru Dittrichovi za rady při řešení problémů s lokalizací.

Obsah

1 Úvod

1.1 Cíl a podmínky práce

Cílem této práce je navrhnout a implementovat algoritmus, který dokáže řídit robotický vysavač tak, aby projel vymezený prostor místo vedle místa a zanechával co možná nejmenší nevyčištěné úseky. Prostor, ve kterém se robotický vysavač bude pohybovat, bude běžná místnost. Je tedy důležité, aby se při své práci dokázal vyhnout všem překážkám. Toto prostředí bude pro robota neznámé, nebude mít předem žádné informace o předmětech v místnosti, ani o jeho počáteční poloze.

1.2 Struktura práce

V následujících kapitolách bude popsáno několik algoritmů, které se zabývají problémem pokrytí prostoru. Budou rozebrány jednodušší (2.1, 2.2), méně přesné algoritmy, které jsou pro implementaci jednoduché, pro vykonání robotem nenáročné, ale do neznámého prostředí se příliš nehodí. Popíši i sofistikovanější algoritmy (2.3, 2.4), které se nezaměřují pouze na to, aby robot navštívil všechna místa, ale zohledňují i délku trasy, kterou při své práci urazí. Také se snaží o minimalizaci míst, které robot navštíví vícekrát. Tyto algoritmy jsou již schopné pracovat v neznámém prostředí.

V praktické části (kapitola 4 a 5) se budu zabývat strukturou hlavní části aplikace. Vysvětlím, jak jednotlivé moduly pracují a co je jejich úkolem. Bude ukázáno grafické uživatelské rozhraní a vysvětleno ovládání demonstračního appletu (kapitola 5.2). Dále v této části práce popíši, jakým způsobem byly teoretické znalosti převedeny do praxe a jaké odlišnosti je možné ve zdrojovém kódu nalézt oproti teoretickému popisu.

V nemalé míře se budu zabývat i problémy, se kterými jsem se při vytváření programu setkal. Bude se jednat především o problémy, které vznikly při vytváření aplikace pro reálného robota. Týkaly se zejména jeho lokalizace a nepřesného měření některých zařízení. Popíši, jak se jednotlivé problémy a chyby projevovaly, co je způsobovalo, a jaký měly efekt na práci robota. Vysvětlím i způsob, jakým jsem tyto problémy řešil. Pokud to charakter chyby dovolil, navrhnul jsem i jiná možná řešení.

1.3 Stav na trhu

V dnešní době je na trhu již poměrně velké množství robotických vysavačů od různých výrobců, které se od sebe na první pohled liší pouze cenou. Můžeme nalézt robotické vysavače s cenou začínající již na 3000 Kč. Tyto vysavače ovšem nedisponují žádnou umělou inteligencí, jejich pohyb je pouze náhodný. Na druhé straně pomyslné cenové řady robotických vysavačů můžeme nalézt modely s cenou blížící se k 30 000 Kč. Tyto modely již mají inteligentní způsob řízení a oplývají celou řadou funkcí a režimů činnosti. Za jejich přednost by se mohla označit i dokovací stanice, která mimo nabíjení slouží i jako shromaždiště smetí, které robotický vysavač při své práci posbírá. Vždy, když robot ukončí svoji práci a připojí se k dokovací stanici, přečerpá smetí ze svého zásobníku do zásobníku dokovací stanice. Není tedy nutné po každém čištění vyprazdňovat zásobník robota, který vzhledem k rozměrům robotických vysavačů není nikterak velký. Nevýhodou této dokovací stanice je její velikost, která až několikanásobně převyšuje ostatní typy. Je tedy pouze na potřebách a možnostech kupujícího, který model si vybere.

2 Plánovací algoritmy

V této kapitole budou popsány a vysvětleny algoritmy pro řízení robotického vysavače. V první podkapitole se budu zabývat velice jednoduchým způsobem řízení, který je založen pouze na náhodné změně směru jízdy po nárazu. V druhé části popíši algoritmus, který umožňuje řízení robota a plánování jeho cesty v předem známém prostoru. V dalších podkapitolách si ukážeme algoritmy, které již zvládají řídit robota i v neznámém prostředí. Poslední z nich je za jistých podmínek, které budou diskutovány v kapitole Implementace, již plně použitelný v reálném prostředí.

2.1 Náhodná změna směru

Robotické vysavače, které využívají tohoto typu řízení, nemají implementovanou žádnou umělou inteligenci. Nemají tedy naprogramovaný žádný algoritmus, který by řídil jejich pohyb. Pohyb je řízen heuristikami, které využívají jednoduchého souboru chování a pohybů při různých situacích, do kterých se robot může dostat. Použití této metody s sebou přináší problém ukončení čistící procedury a zaručení úklidu všech míst.

Výhodou těchto robotů je, že si nemusí pamatovat mapu prostoru, ve kterém se pohybují. Náročnost na senzory také není vysoká, zpravidla se používají pouze mechanická čidla. I když jsou tato čidla citlivá a nárazy nejsou nikterak silné, hrozí převrhnutí méně stabilních předmětů umístěných na podlaze nebo postupné poškozování nábytku.

Dalším typem robota, s řízením bez umělé inteligence je ten, který nepoužívá žádnou heuristiku, ale jeho pohyb je zcela náhodný. Narazí-li na překážku, náhodně změní směr a pokračuje v jízdě. Problémem tohoto typu řízení i toho založeného na heuristikách je, že se robot může zacyklit v nějakém těžko přístupném místě. Může neustále narážet a měnit směr, ale z místa, odkud vede jedna cesta jen o něco málo širší, než je robot sám, se nemusí bez cizí pomoci dostat.

Velkou výhodou těchto robotů je nízká náročnost na senzory a na řídící jednotku. Použitím takto zjednodušeného přístupu výrobci mohou nabízet robotické vysavače i s pětkrát nižší cenou, oproti ceně robotů s umělou inteligencí.

2.2 Přibližný rozklad na buňky

V této kapitole bude vysvětlen způsob řízení a plánovaní cesty algoritmů, které využívají přibližný rozklad na buňky

Přibližný rozklad na buňky je první stupeň na cestě za inteligentním řízením robota. Tento způsob řízení předpokládá, že před započetím úklidové procedury je v paměti uložena mapa prostředí, ve kterém se robot bude pohybovat. Pokud je tato podmínka splněna, je tento algoritmus efektivní z hlediska ujeté vzdálenosti, přesnosti a preciznosti úklidu. Minimalizace ujeté vzdálenosti lze dosáhnout díky tomu, že řídící algoritmus zná rozmístění a pozici překážek. Může tedy efektivně naplánovat cestu úklidu.

2.2.1 Rozdělení na buňky

V literatuře [4] se lze také setkat s pojmem *Plánování na mřížce*, který je ekvivalentní názvu této kapitoly. Princip metody je vcelku jednoduchý. Pokrývaný prostor se rozloží na buňky o stejné velikosti – pokryje se sítí. Velikost buňky může být například šířka stopy robota, kterou je schopen vysát. Z praktického hlediska bych volil spíše menší šířku stopy, neboť robotický vysavač bývá většinou kruhového tvaru a může tedy u stěn a překážek zanechávat nevyčištěná místa. Zvolení menší stopy by sice vedlo k prodloužení ujeté trasy, ale díky překrývání stop by nevznikala tak velká nevyčištěná místa (obrázek číslo 3). Z těchto důvodů, je třeba velikost buněk řádně zvážit. Problematikou velikosti buněk se budeme podrobněji zabývat v kapitole 2.2.3.

Po rozdělení místnosti na buňky se každá ohodnotí podle algoritmu *záplavového vyplňování* [7] tak, jak je ukázáno na obrázku číslo 1. Algoritmus pracuje v těchto krocích:

1) Zvolí se cílové místo *G*, které má ohodnocení 0.
2) Aktuální buňka je ta, která byla právě ohodnocena.
3) Všem volným neohodnoceným sousedním buňkám přiřadí hodnotu o jedna vyšší, než je ohodnocení aktuální buňky.
4) Pro každou ze sousedních buněk provede bod 3.
5) Pokud jsou ohodnoceny všechny buňky celého prostoru, tak algoritmus končí, jinak se vrátí na bod 2.

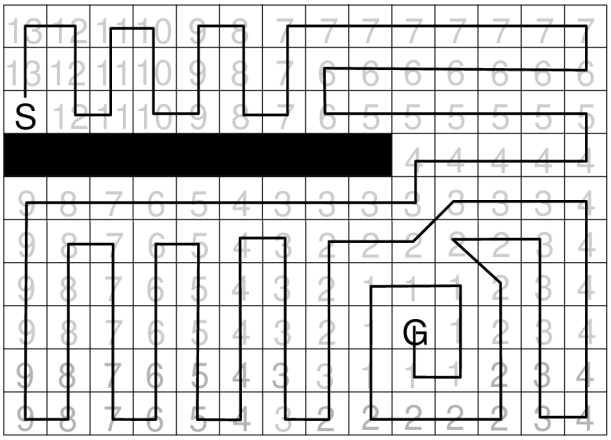
13	12	11	10	9	8	7	7	7	7	7	7	7	7
13	12	11	10	9	8	7	6	6	6	6	6	6	6
S	12	11	10	9	8	7	6	5	5	5	5	5	5
									4	4	4	4	4
9	8	7	6	5	4	3	3	3	3	3	3	3	4
9	8	7	6	5	4	3	2	2	2	2	2	3	4
9	8	7	6	5	4	3	2	1	1	1	2	3	4
9	8	7	6	5	4	3	2	1	**G**	1	2	3	4
9	8	7	6	5	4	3	3	1	1	1	2	3	4
9	8	7	6	5	4	3	2	2	2	2	2	3	4

Obrázek 1: Ohodnocené buňky, buňka *S* představuje startovací pole, buňka *G* představuje cílové pole

2.2.2 Cesta úklidu

V tento moment je celá místnost rozdělena do buněk s určitým ohodnocením. Samotný algoritmus pro nalezení cesty, která by zajistila vyčištění celé místnosti, vychází z algoritmu pro vyhledávání nejkratší cesty. Ten se při pohybu buňkami snaží jít „z kopce dolů“ (od nejvyšších čísel na nulu). Při rozhodování, na kterou sousední buňku pokračovat vybere tu s nejnižším ohodnocením.

Pokud tento algoritmus zmodifikujeme tak, aby se robot při rozhodování nevydal na buňku s nejnižším ohodnocením, ale naopak na buňku s nejvyšším ohodnocením, docílíme toho, že robot při pohybu místností postupně projde všechny buňky, tedy i celou místnost.



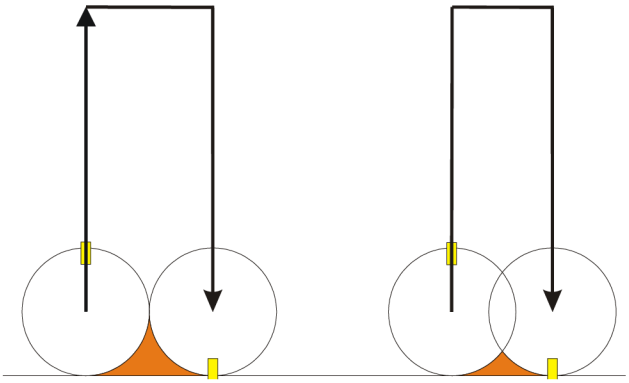
Obrázek 2: Zobrazená cesta ze startovacího políčka *S* do cílového *G*

Na obrázku číslo 2 je vidět, jakou cestou robot prochází. Je zřejmé, že se cesta nikde nekříží a ani se žádné části neuklízí vícekrát. Ovšem jak již bylo řečeno, algoritmus musí znát přesnou mapu prostředí, ve kterém se bude pohybovat. Díky této vlastnosti tedy není vhodný jako řídící algoritmus běžných, v dnešní době prodávaných robotických vysavačů.

Pokud bychom tohoto algoritmu chtěli využít v neznámém prostředí, bylo by nutné, aby si robotický vysavač zapamatoval mapu uklízeného pokoje. Při příštím uklízecím cyklu by se tedy tento algoritmus mohl využít. Problém by ale nastal, pokud by se v prostoru jen nepatrně přemístily překážky, což je v běžné domácnosti téměř jisté. Musel by se opět spoléhat na senzory, kterými by detekoval změny v pozici a množství překážek. Tyto změny by musel zaznamenávat do své mapy prostoru a tu znovu přepočítávat při každé odlišnosti. Tím by podstata tohoto algoritmu, kterou je předpočítání cesty před počátkem práce, odcházela do ústraní.

2.2.3 Velikost buňky

Jak již bylo naznačeno v kapitole 2.2.1, velikost jednotlivých buněk je třeba dobře zvážit. Pokud jako velikost buňky zvolíme šířku stopy robotického vysavače, budou u stěn zůstávat nevyčištěná místa. Snížením velikosti kroku dosáhneme i snížení nevysáté plochy (viz. obrázek číslo 3). Robot nebude zanechávat volná místa, pokud bude šířka kroku rovna jeho poloměru.



Obrázek 3: Rozdíl nepokryté plochy při velikosti kroku rovno šířce stopy vysavače (vlevo) a 75 % z šířky stopy (vpravo)

Dalším faktorem, který by při rozhodování o šířce stopy hrál podstatnou roli, by byla velikost překážek. Pokud by překážka byla menší než šířka stopy, vznikalo by kolem ní nevyčištěné místo. Pokud by takto malých překážek bylo větší množství, mohla by nepokrytá část mít poměrně nezanedbatelnou velikost. Stejný problém by nastal, pokud by se při vytváření buněk nevešla do zbylého prostoru celá buňka. V místech, kam by se celá buňka nevydala, by vznikala hluchá místa, do kterých by vysavač nezajel, a zůstávala by tedy nevyčištěna. Řešením této situace by bylo zavedení drobnějších vyplňovacích buněk, kterými by se daná místa doplnila.

2.3 Přesný rozklad na buňky

V anglické literatuře ([1], [2],[3] apod.) se tento pojem nazývá Exact cell decomposition. Základní myšlenkou těchto algoritmů je rozložení prostoru do jednoduchých tvarů, pro které nebude problém naplánovat a provést pokrytí. Takovým tvarem může být trojúhelník, nebo lichoběžník. V následujících podkapitolách se budeme zabývat dvěma způsoby rozkladu prostoru na buňky. A to konkrétně Lichoběžníkovou dekompozicí a algoritmem Boustrophedon.

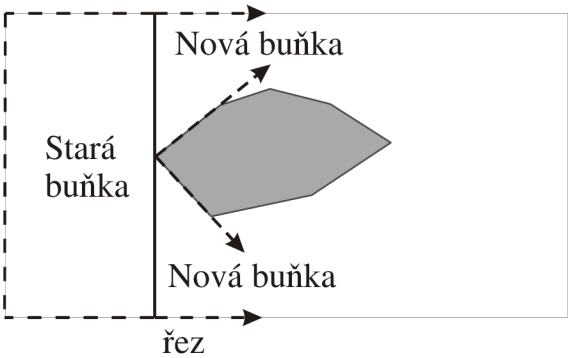
2.3.1 Lichoběžníková dekompozice

Tento přístup využívá přesného buněčného rozkladu takovým způsobem, že vzniknou nepřekrývající se buňky. Spojením buněk vznikne opět původní prostor, který je naprosto totožný se skutečným vzorem. Pokrytí v jednotlivých buňkách se docílí jednoduchým pohybem tam a zpět, který je ukázán na obrázku 9. Pokrytí celého prostoru se dosáhne tak, že robot navštíví a pokryje každou buňku. Jsou-li všechny buňky pokryty, je pokryt i celý prostor.

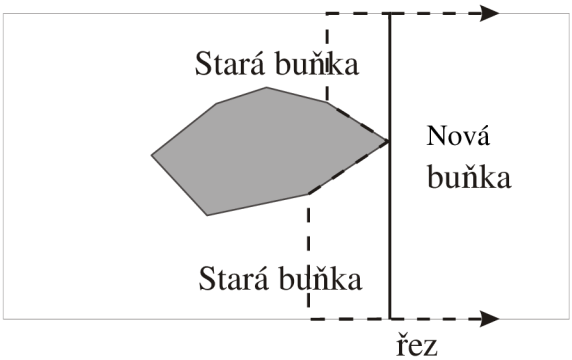
Každá buňka může reprezentovat uzel v grafu, ve kterém mají sousední uzly společnou hranu. Tento graf se nazývá *graf sousednosti*. Protože je tvar buňky jednoduchý a nemusí se tedy uvnitř ní plánovat sofistikovanější pohyb, tak se problém redukuje pouze na projití každého uzlu v grafu sousednosti právě jedenkrát. Tento problém se nazývá problém *obchodního cestujícího*. Problém obchodního cestujícího lze řešit jedním z algoritmů pro prohledávání stavového prostoru ([6]).

Lichoběžníková dekompozice se vytváří průchodem svislé čáry napříč celým prostorem. Tato čára se nazývá řez. Při průchodu řezu se udržuje seznam protnutých buněk. Všechny buňky, které již seznamem prošly, vytváří výsledný rozklad. Nová buňka vzniká ve chvíli, kdy řez narazí na vrchol překážky.

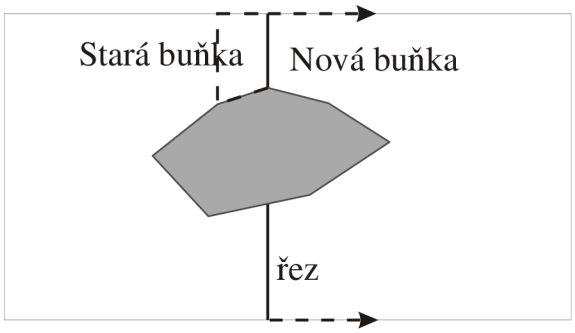
V následujícím textu budu používat terminologii z [2]. Jak bylo řečeno v předchozím odstavci, nová buňka vznikne, když řez narazí na vrchol překážky. Tato situace se nazývá IN. Během dekompozice nastávají ještě další operace. Tyto operace jsou OUT a MIDDLE. Při operaci IN se uzavírá aktuální buňka (dokončí se její konstrukce) a otevřou se dvě nové buňky. Situace je zobrazena na obrázku 4. Operace OUT znamená, že se uzavřou dvě buňky a jedna nová se vytvoří (obrázek 5). MIDDLE je situace, kdy se jedna buňka uzavře a nová se otevře (obrázek číslo 6).



Obrázek 4: IN operace

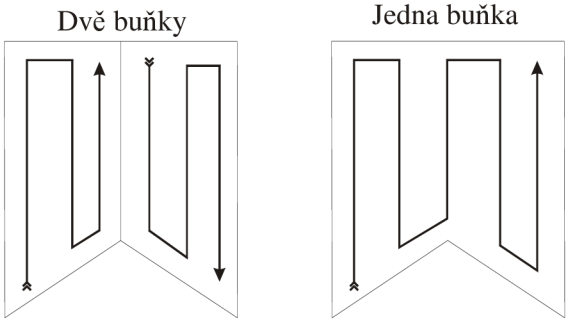


Obrázek 5: OUT operce

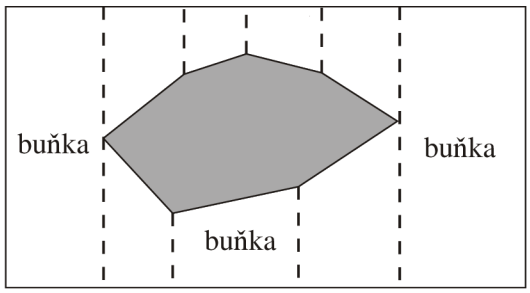


Obrázek 6: MIDDLE operace

Nevýhodou lichoběžníkové metody je, že generuje zbytečné buňky. Tyto buňky vznikají při operaci MIDDLE. Pokrytí prostoru nemusí být kvůli počtu vygenerovaných buněk efektivní. Na levé části obrázku číslo 7 je vidět, že při pokrývání každé buňky zvlášť je potřeba jedna cesta navíc než v pravé části obrázku, přitom složitost buňky na pravé straně je z hlediska jejího pokrývání téměř totožná s buňkou na levé straně. Pokud by nastala situace, že vzdálenost mezi překážkou a koncem pokrývaného prostoru (stěnou) by byla velká, mohl by se čas potřebný k pokrytí znatelně prodloužit.



Obrázek 7: Pokrytí dvou buněk VS. pokrytí jedné buňky



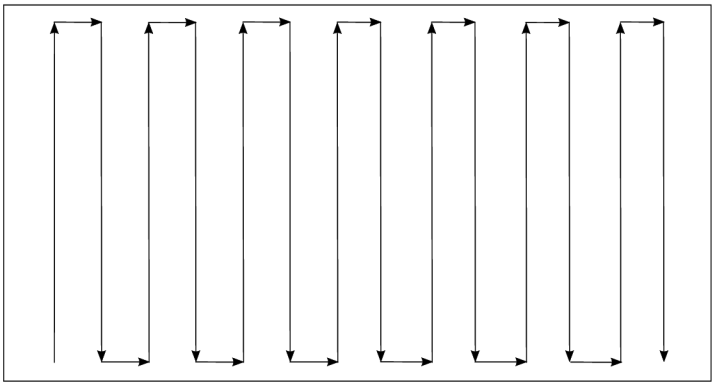
Obrázek 8: Lichoběžníková dekompozice

Algoritmus

Algoritmus při svoji práci využívá seznam x-ových souřadnic vrcholů překážek. Z toho tedy vyplývá, že pro svoji činnost musí předem znát rozmístění překážek. Pokud bychom chtěli algoritmus používat v neznámém prostředí, musel by být tento seznam vytvářen v průběhu činnosti algoritmu a nově nalezené vrcholy překážek by musely být zařazovány do seznamu na správné místo.

x_L = *pozice řezu*
$\{x_1, x_2 \dots x_n\}$ = *seznam x − souřadnic vrcholů*
$D = (\dots, c_{i-1}\ c_i, c_{i+1}, \dots)$ = *seznam buněk*
for $x_L = x_1$ *to* x_n **do**
  if *vrchol rozděluje buňku* c_i *na dvě*
    then c_i *nahraď* c_d, c_{d+1}
  else if *vrchol slučuje dvě buňky* c_i *a* c_{i+1}
    then c_i *a* c_{i+1} *nahraď za* c_d
  else if *vrchol nahrazuje buňku ci novou buňkou*
    then c_i *nahraď* c_f
  end if
end for

Algoritmus 1: Algoritmus lichoběžníkové dekompozice

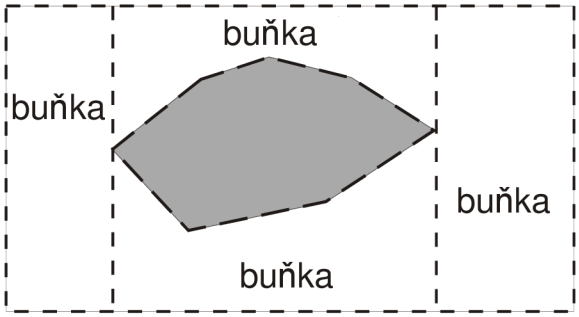


Obrázek 9: Jednoduchý pohyb, který se používá při pokrývání jednotlivých buněk

2.3.2 Boustrophedon dekompozice

Slovo Boustrophedon[1] pochází z řeckého slova, jehož význam je *„otáčení jako vůl v orbě“*. Tento název tedy napovídá, jakým způsobem bude algoritmus pracovat. Při pokrývání jednotlivých buněk bude pohyb robota připomínat vola, který oře na poli. Tento pohyb byl již využíván v lichoběžníkové dekompozici pro pokrývání jednotlivý buněk.

Boustrophedon dekompozice je modifikace výše uvedené lichoběžníkové dekompozice. Odstraňuje problém se zbytečně vzniklými buňkami při operaci MIDDLE. Výsledkem je menší počet buněk, z čehož vyplývá i menší počet pohybu tam a zpět při pokrývání buňky na hranici se sousední buňkou. Výsledná dekompozice je znázorněna na obrázku 10. Z obrázku je patrné, že všechny buňky mezi operacemi IN a OUT jsou sloučeny do jediné.



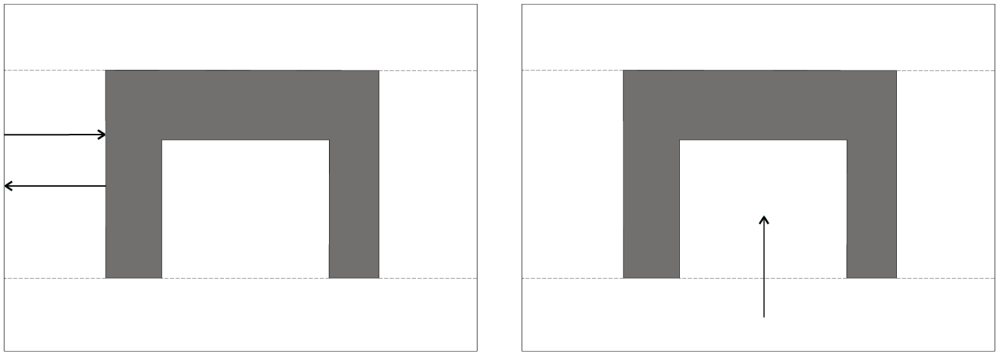
Obrázek 10: Boustrophedon dekompozice

Při dekompozici řez opět prochází skrz prostředí s překážkami. Pokud řez narazí na překážku, uzavře se aktuální buňka a dvě nové buňky se otevřou. Tato operace se nazývá, stejně jako v Lichoběžníkové dekompozici, operace IN. Operace OUT uzavírá dvě buňky a otevírá novou buňku. Rozdíl mezi Boustrophedon dekompozicí a Lichoběžníkovou dekompozicí je v MIDDLE operaci. V Boustrophedon dekompozici se při MIDDLE operaci neotevírají, ani nezavírají žádné buňky.

[1] Vysvětlení slova Boustrophedon převzato z Wikipedie: http://en.wikipedia.org/wiki/Boustrophedon

Nekonvexní překážky

Omezení algoritmu Boustrophedon je, že dokáže zpracovat pouze konvexní překážky. Tedy překážky, které mají všechny vnitřní úhly menší než 180°. V běžné domácnosti se ale zcela jistě vyskytují i překážky nekonvexní – překážky ve tvaru U apod. Na obrázku číslo 11 je znázorněna ukázka nekonvexní překážky. Při použití nemodifikovaného algoritmu Boustrophedon by část uvnitř překážky zůstala nepokryta. Řešení této situace je vést řez ve dvou směrech, nejprve jedním směrem a po dokončení dekompozice směrem opačným. Tím by se docílilo kompletního pokrytí prostoru, ve kterém se nachází nekonvexní překážky. Pokud bychom ale toto dvojité čistění použili, čas potřebný k vyčištění místnosti by se zdvojnásobil, což může být nežádoucí jev či neřešitelný problém s ohledem na omezenou kapacitu a výdrž baterie, kterou je robot poháněn.



Obrázek 11: Nekonvexní překážka

Neznámé prostředí

Tak jak byl Boustrophedon algoritmus popsán v předchozí části tohoto textu ho lze využít pouze pro známé prostředí, kde je pozice a velikost překážek známá a přesně určená. Robot by předem prostředí rozdělil na buňky a poté naplánoval cestu mezi nimi a jejich samotné pokrytí. V neznámém prostředí ovšem nelze řez aplikovat na prostor tak, jak bylo ukázáno výše. V reálné situaci se řezem stává sám robot.

Neznámé prostředí zavádí řadu problémů, které je nutno řešit. Pokud se například detailněji podíváme na operaci IN, zjišťujeme, že konstrukce buněk, které vznikly při této operaci, by měla probíhat současně a současně by se také měly tyto dvě buňky uzavřít. Toho v reálné situaci nelze s jedním robotem dosáhnout, protože robot nemůže být na dvou místech zároveň (z jedné i z druhé strany překážky). Proto je nutné, aby byla nejprve pokryta jedna strana překážky a poté při operaci OUT by se robot vrátil na pozici, kde překážka rozdělila volný prostor (na pozici OUT). Následně by začal vytvářet a čistit buňku z druhé strany překážky. Ve chvíli, když by dokončil čištění i druhé buňky, by teprve mohl provést operaci OUT.

Algoritmus

$x_L = pozice\ řezu$
$\{x_1, x_2 \dots x_n\} = seznam\ x - souřadnic\ vrcholů$
$D = (\dots, c_{i-1}\ c_i, c_{i+1}, \dots) = seznam\ buněk$
$\boldsymbol{for}\ x_L = x_1\ to\ x_n\ \boldsymbol{do}$
    $\boldsymbol{if}\ vrchol\ rozděluje\ buňku\ c_i\ na\ dvě$
        $\boldsymbol{then}\ c_i\ nahraď\ c_d, c_{d+1}$
    $\boldsymbol{else\ if}\ vrchol\ slučuje\ dvě\ buňky\ c_i\ a\ c_{i+1}$
        $\boldsymbol{then} c_i\ a\ c_{i+1}\ nahraď\ za\ c_d$
    $\boldsymbol{end\ if}$
$\boldsymbol{end\ for}$

Algoritmus 2: Algoritmus Boustrophedon dekompozice

2.4 Dekompozice řezem

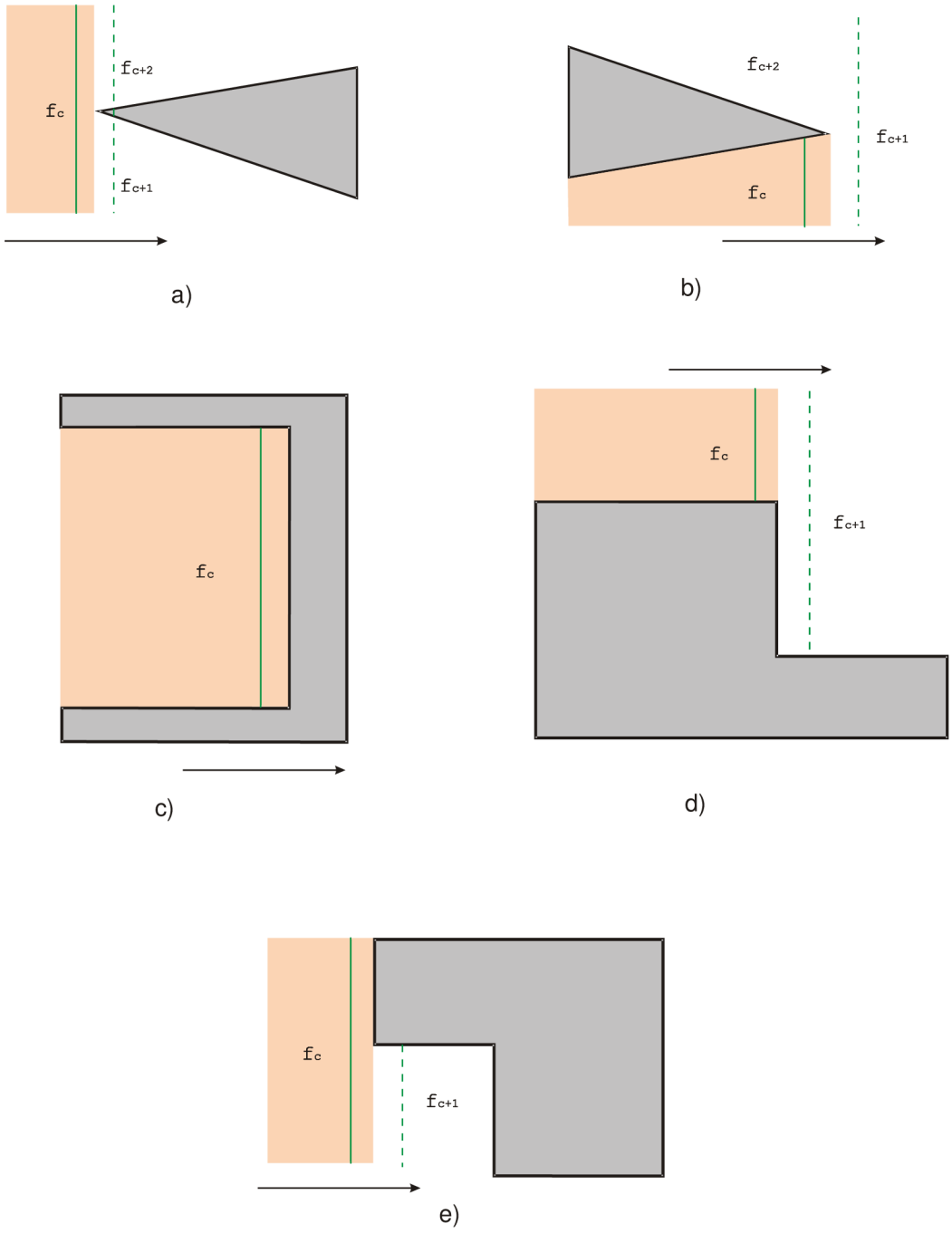
V této kapitole bude vysvětlena funkčnost metody a způsob, jakým se při dekompozici řezem vytváří jednotlivé buňky. Dekompozice řezem je další metoda, která dělí prostor na buňky, která využívá řez. Tato metoda se ovšem od předchozích dvou liší tím, že už její první teoretický popis vychází z toho, že se robot bude pohybovat v neznámém prostředí, které obsahuje překážky různých tvarů. Počítá tedy i s nekonvexními překážkami, se kterými si předchozí metody bez opakování činnosti v opačném směru neuměly poradit.

2.4.1 Události

Díky novým možnostem algoritmu vznikají i nové události, které jsou oproti předchozím algoritmům mírně rozšířeny. Jedná se zejména o událost END, která se v předchozích dvou metodách nevyskytuje. Je to událost, která vzniká v momentě, kdy se robot nachází ve slepé uličce a nemůže dále pokračovat. Tato slepá ulička je část prostředí, která se nachází v nekonvexní části překážky. Z tohoto důvodu událost END nebyla v předchozích dvou metodách (předchozí dvě metody uměly pouze konvexní překážky). Na obrázku 12 můžeme vidět změny tvaru překážky, které se v dekompozici řezem uvádí jako události. S ohledem na to, že se řez může pohybovat i vzad (kvůli pokrytí nekonvexních částí překážky) je u každého obrázku uveden směr pohybu řezu. Události, které mohou při dekompozici řezem nastat, jsou:

a) SPLIT – Jedná se o ekvivalentní událost IN v Lichoběžníkové dekompozici nebo Boustrophedon dekompozici. Volný prostor se dělí na dvě části. Obrázek 12 a).
b) MERGE – U této události by se mohlo zdát, že se opět jedná o ekvivalent k události OUT. Opak je pravdou, neboť událost MERGE, na rozdíl od operace OUT vytváří dvě nové buňky. Neslučuje tedy dva volné prostory. Obrázek 12 b).
c) END – Situace kdy pokračování daným směrem není možné. Robot se nachází v uzavřené oblasti. Obrázek 12 c).
d) LENGTHEN – Událost, při které se délka řezu prodlouží. Obrázek 12 d).
e) SHORTEN – Událost, kdy se délka řezu volným prostorem zmenší Obrázek 12 e).

Podle vysvětlení v literatuře [5] události LENGTHEN a SHORTEN odpovídají operaci MIDDLE Lichoběžníkové dekompozice. I když Boustrophedon odstraňuje tuto operaci a tím minimalizuje počet buněk, v literatuře zabývající se dekompozicí řezem [5] tyto dvě události zůstávají. Proto byly i v tomto textu operace zachovány.



Obrázek 12: Události při dekompozici řezem – neznámé prostředí a) SPLIT, b) MERGE, c) END d) LENGTHEN, e) SHORTEN

Na obrázku 12 jsou vidět události, které mohou nastat při dekompozici řezem. Zelená plná čára je střed robota a světle oranžové pozadí představuje část volného prostoru, která již byla zpracována. Čárkovaná zelená čára je následující řez volným prostorem. Detekce vrcholů a překážek bude probrána v další kapitole.

2.4.2 Algoritmus

Algoritmus ke své činnosti potřebuje dva seznamy O (*open*) a F (*finish*). Seznam O obsahuje všechna volná místa (buňky), které byly při procházení prostorem objeveny. Seznam F si pamatuje všechny buňky, které byly navštíveny a vyčištěny. Algoritmus vybírá ze seznamu O uzly, dokud není seznam

prázdný. Ve chvíli, kdy k tomu dojde, algoritmus končí svoji činnost, jelikož všechna místa byla navštívena. Každý vybraný uzel ze seznamu *O* představuje buňku, jež má být pokryta. Na každý uzel se tedy aplikuje jednoduchý pohyb tam a zpět od jednoho okraje buňky k druhému. Když nastane událost ze seznamu *O*, odebere se aktuálně zpracovávaný uzel (na obrázku 12 a v algoritmu číslo 3 je označen $f_{c)}$ a přidá se do seznamu *F*, tím se dokončí jeho pokrývání. Pokud nastane událost MERGE nebo SPLIT, do seznamu otevřených uzlů *O* se přidají všechny uzly, které sdílejí stejnou hranici (na obrázku 12 a v algoritmu číslo 3 jsou označeny f_{c+1},f_{c+2}).

$O \leftarrow prvni\ bunka$
$F \leftarrow \emptyset$
while $O \neq \emptyset$ **do**
$f_c \leftarrow f \in O$
$presun\ k\ jednomu\ okraji\ aktualni\ bunky\ f_c$
repeat
$presun\ o\ vzdalenost\ \Delta y\ a\ jed\ k\ druhemu\ okraji\ aktualní\ bunky\ f_c$
if $nastala\ udalost$ **then**
$F \leftarrow F + f_c$
$O \leftarrow O - f_c$
if $udalost = slouceni\ nebo\ rozdeleni$ **then**
$O \leftarrow O + f_{c+1}, f_{c+2}$ $\quad f_{c+1}, f_{c+2} \notin O$
end if
if $udalost = prodlouzeni\ nebo\ zkraceni$ **then**
$O \leftarrow O + f_{c+1}$ $\quad f_{c+1} \notin O$
end if
end if
until $nastala\ udalost$
end while

Algoritmus 3: Algoritmus dekompozice řezem

3 Robotický vysavač

Na našem trhu se v současné době pohybuje celá řada výrobců robotických vysavačů. Jako zástupce bych jmenoval *CleanMate, Goddess, iClebo, iRobot, RobZone* nebo *Samsung*. Všechny firmy nabízí hned několik modelů vysavačů jak na vysávání, tak i na vytírání. V následujících kapitolách budou popsány jejich senzory, vybavení a systém jejich práce, který se ovšem může model od modelu lišit. Proto následující kapitoly budou brány spíše jako ukázka možných dovedností robotických vysavačů.

3.1 Algoritmus

Jelikož přesný algoritmus činnosti robotických vysavačů je firemním tajemstvím, nelze zjistit, jakým algoritmem vysavače pracují. Některé firmy uvádějí alespoň základní informace o činnosti.

3.2 Režimy činnosti

Někteří výrobci uvádějí, že jejich roboti mají naprogramováno několik algoritmů pro řízení. Střídáním režimů se robotický vysavač dostane vícekrát na různá místa a tím se tedy zlepší výsledek jeho práce. Tímto způsobem uklízí vysavač firmy *RobZone* (i další), který se nejprve pohybuje různě po místnosti ve spirálách. Poté se přepne do režimu, ve kterém jeho pohyb připomíná algoritmus boustrophedon. Tento i ostatní výrobci umožňují přepnutí vysavačů do režimu, ve kterém neuklízí celou plochu, ale soustředí svoji činnost pouze na rohy a kouty. Vysavač tedy kopíruje tvar překážek a tím zajistí uklizení nečistot, které zanechal například při otáčení.

Firma *iRobot* osazuje své vysavače senzory pro detekci velice znečištěných míst. Pokud vysavač této firmy při své práci nalezne místo, které je hodně znečištěné, přepne se do režimu, ve kterém intenzivně uklízí blízké okolí nalezených nečistot. V tomto režimu robotický vysavač jezdí spirálovitě do té doby, dokud senzor detekuje výrazné nečistoty.

3.3 Senzory a snímače

Senzory robotickým vysavačů slouží k orientaci v prostoru, ale i k ochraně proti pádu např. ze schodů. Vysavače jsou osazeny různými druhy senzorů od ultrazvukových sonarů přes infračervené až po širokoúhlé kamery.

- PSD (Position Sensitive Device) senzory – těmito senzory je osazen robot od firmy *iClebo*, díky nim dokáže odhadnout výšku překážky pod sebou a tím zabránit pádu ze schodů. Senzory pro detekci schodů je vybavena většina vysavačů.
- Infračervené dálkoměry – slouží k měření vzdálenosti od překážky. Pracují na principu triangulace [17].
- Ultrazvukové sonary – sonary slouží k detekci překážek v okolí robotického vysavače, při měření vzdálenosti se počítá doba od vyslání ultrazvukového signálu po jeho přijetí.
- Širokoúhlé kamery – je to poslední technologie, kterou nasadila firma *Samsung* na svého robota. Tento systém snímání okolí se nazývá *Visionary Mapping*^{TM} a je schopen pořídit až 30 snímků za sekundu. Díky tomuto systému si robot vytváří mapu okolí, kterou může využít

v režimu, ve kterém přesně objíždí překážky, aby odstranil nečistoty, které zanechal při otáčení.

- Senzory nečistot – při svém pohybu robotický vysavač snímá podlahu pod ním a detekuje více znečištěná místa. V případě, že se nad takovým místem nachází, změní svůj režim činnosti a nečistoty odstraní.

3.4 Jiné vybavení

- *Silence technology* – je technologie robotů *iClebo* zajišťující co možná nejtišší chod. Na tomto místě si dovolím tvrdit, že se jedná pouze o reklamní trik, protože po zhlédnutí srovnávací tabulky[2] se tito roboti *iClebo* se svými 60 dB (odpovídá hluku kytary ze vzdálenosti 40 cm)[3] řadí mezi průměrné. Podle výše uvedeného přehledu jsou nejtišší robotické vysavače od firem *Goddess, Bralko Galaxy* nebo *Fun Beat*. Tyto modely dosahují hlučnosti 55 dB. Na opačném konci tabulky jsou robotické vysavače značky *CleanMate* s 80 dB, což přibližně odpovídá hluku automobilu.
- *Virtuální zeď* – se využívá pro vymezení plochy úklidu, je tedy možné vysavači omezit vstup do další místnosti, kterou nelze uzavřít dveřmi. Virtuální zeď vysílá infračervené paprsky.
- *Dálkové ovládání* – pomocí dálkového ovládání lze vysavač řídit manuálně, takto lze přivolat vysavač např. na rozsypané smetí.

3.4.1 Virtuální stěna

Většina robotických vysavačů umožňuje omezení jejich pracovního prostoru pomocí virtuálních stěn. Tyto stěny pro robota představují hranici, za kterou se nesmí dostat. U některých modelů robotických vysavačů je možno nastavit virtuální stěny do režimu, ve kterém robota nechají projet až poté, kdy má uklizený celý pokoj. Tímto lze posílit jeho systematický úklid domu.

3.4.2 Detekce schodů

Mezi důležité schopnosti bych zařadil detekci schodů. Robot při své práci neustále kontroluje, jestli se jeho část nedostala za hranu schodu. Pokud tato situace nastane, robot okamžitě zastaví a kousek se vrátí, aby mohl změnit směr své jízdy a nespadl přitom ze schodů.

[2] Srovnávací tabulku lze nalézt na http://www.robotforum.cz/srovnani/
[3] Zdroj informací o hluku http://cs.wikipedia.org/wiki/Decibel

Obrázek 13: Ukázka práce robotického vysavače *iRobot Roomba*. Modrá část představuje vyčištěnou plochu po ukončení úklidového procesu.

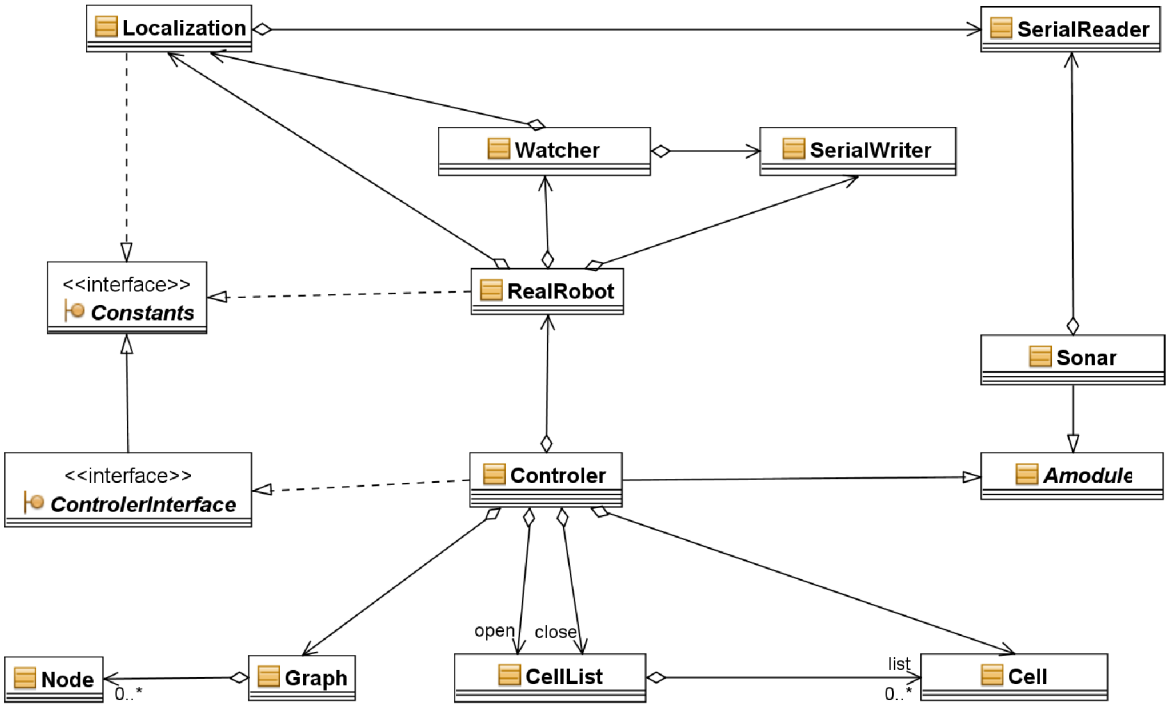
4 Návrh

Protože výsledkem mé bakalářské práce má být algoritmus pro řízení robotického vysavače a demonstrační applet, který bude ukazovat způsob práce jednotlivých algoritmů. V návrhu se budu zaměřovat na to, aby struktura byla použitelná pro demonstrační applet i pro skutečného robota. Na základě teoretických znalostí o jednotlivých metodách bude navrženo praktické řešení těch z nich, které se umí řídit robota v neznámém prostředí. Jedná se tedy o lichoběžníkovou dekompozici, Boustrophedon dekompozici a dekompozici řezem.

Demonstrační applet bude umístěn na webové adrese www.stanislavsojka.cz/bakalarska-prace.

4.1 Diagram tříd

Na obrázku 14 je diagram tříd, podle kterého byla aplikace vytvořena. Diagram zobrazuje hlavní část aplikace pro ovládání reálného robota. Pro demonstrační applety je digram tříd téměř totožný, jen neobsahuje třídy pro komunikace po sériové lince.



Obrázek 14: Diagram tříd zobrazující hlavní část aplikace pro ovládání reálného robota

4.2 Uživatelské rozhraní

Uživatelské rozhraní se bude vyskytovat pouze na demonstračním appletu. Hlavní částí bude vytyčený prostor pro pohyb vysavače. Při návrhu velikosti a tvaru vysavače jsem se držel komerčních modelů. Jejich tvar je kruhový s průměrem přibližně 40 cm. Do demonstračního appletu jsem proto zvolil kruhový tvar s průměrem 40 px. Velikost prostoru, ve kterém se bude robotický vysavač pohybovat, jsem zvolil 450 x 900 px. Tento rozměr už pro běžné domácnosti tolik typický není. Rozhodl jsem se ovšem, že pro lepší demonstraci algoritmů bude názornější velký prostor.

Vedle panelu s místností se bude nacházet panel s údaji o stavu robota. V tomto panelu uživatel vidí, co detekují jednotlivé sonary, jakou má robot pozici a další informace spojené s pohybem robota.

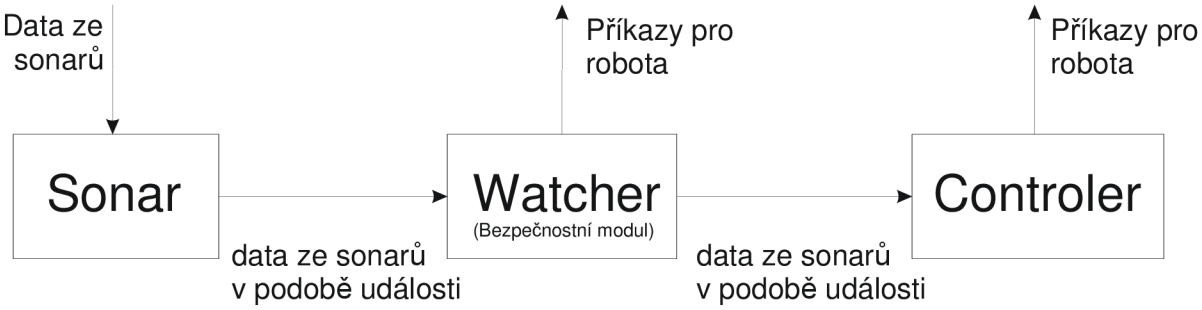
Pod hlavní panelem se bude nacházet ovládací panel. Ten bude uživateli dovolovat měnit rychlost vysavače, pouštět jednotlivé metody a umožňovat výběr místností (rozmístění překážek v místnosti). Celkový rozměr okna aplikace bude 1000 x 600 px. S ohledem na statistiky nejpoužívanějších rozlišení monitorů[4] nebude většině uživatelů tato velikost překážkou v bezproblémovém zobrazení. Výsledný vzhled si je možné prohlédnout na obrázku 20.

4.3 Struktura aplikace

Pokud pomineme pomocné části, jakými jsou třídy zachytávající události tlačítek, zajišťující vykreslení uživatelského rozhraní apod., aplikace bude rozdělena do tří základních modulů (tříd). První z nich je třída zapouzdřující vlastnosti sonarů. V demonstračním appletu bude zjišťovat, jestli se vykreslený sonar neprotíná s některou z překážek. Zjištěný stav bude posílat jako událost, kterou si bude zachytávat jiný modul. V aplikaci pro ovládání reálného robota bude modul sonaru posílat události s hodnotou, kterou přečte ze sériové linky.

Modul, který události vyvolané sonarem zachytává, slouží pro kontrolu, jestli robot nenarazí. Jeho funkce se využívá v době, kdy robot přejíždí mezi dvěma buňkami a hrozila by kolize s některou z překážek z důvodu nepřesně naměřených údajů. Protože v demonstračním appletu chyby v měření nevznikají, je tento modul vypnut, resp. pouze přeposílá přijatá data. V aplikaci pro řízení reálného robota se tento modul již využívá. U každých přijatých dat se zkontroluje, jestli nehrozí náraz. Pokud náraz hrozí, je vyslán příkaz pro robota k zatočení směrem od překážky. Jeho jízda se přitom nepřeruší, ale robot pokračuje dál. Změna směru jízdy ale musí být co možná nejmenší, aby se robot příliš nevychýlil ze svého směru a neminul cíl.

Dalším modulem je řídící jednotka. Ta přijímá přeposlaná data od předchozího modulu. Podle přijatých hodnot rozhoduje, jakým směrem robot pojede a ukládá si důležité pozice pro pozdější orientaci v prostoru. Na následujícím obrázku 15 je zobrazena struktura hlavní části aplikace.



Obrázek 15: Struktura hlavní části aplikace

4.4 Reprezentace buněk

Jednotlivé buňky budou reprezentovány vlastní třídou, která bude zapouzdřovat jejich vlastnosti. Pro správnou funkci jednotlivých metod je důležité, aby bylo možné nastavit přesnou pozici buňky. Každá buňka bude mít uloženou počáteční pozici, tedy pozici některé z událostí IN, OUT nebo MIDDLE.

[4] Statistiky lze nalézt na http://marketshare.hitslink.com/report.aspx?qprid=17

Dále ve třídě bude udržován seznam, nebo jiná datová struktura bodů, které tato buňka obsahuje. Při konstrukci buňky se budou do tohoto seznamu ukládat pozice robota. Není nutné ukládat všechny body, nad kterými se robot pohybuje, ale bude stačit, když se do seznamu uloží pouze okrajové body buňky. Tedy body, ve kterých robot mění směr po nárazu.

Další vlastností buňky, která je pro správnou funkci důležitá, je směr, kterým se bude robot v buňce pohybovat. U lichoběžníkové dekompozice a Boustrophedon dekompozice tento směr nebude hrát žádnou roli, protože robot při své práci směr nemění. V tomto kontextu není směr pohybu robotovo aktuální natočení, ale jedná se o směr, kterým se pohybuje oproti jeho výchozí pozici např. zleva doprava. Směr je velmi důležitý u dekompozice řezem, protože ho robot při své práci mění a je tedy nutné, aby každá buňka měla směr uložený.

4.5 Graf sousednosti

Protože při události IN vznikají dvě buňky, ale robot nejdříve zpracuje jednu a až poté druhou, musí existovat způsob, kterým robot najde cestu do této druhé buňky. Z tohoto důvodu, musí být při vytváření buněk zároveň tvořen i graf sousednosti, pomocí kterého se najde cesta do cílové buňky. Cesta v grafu sousednosti bude počítána algoritmem BFS tak, jak je definován v [6]. Protože algoritmus BFS je slepá metoda[5], nalezená cesta nemusí být vždy nejlepší. Pokud by byl požadavek na nalezení nejkratší cesty, bylo by vhodnější použít některé z informovaných metod (BestFS, GS, A* Search apod.). Díky objektově orientovanému návrhu by tato změna nečinila žádné potíže. Postačuje pouze zdědit třídu `Graph` a přepsat metodu `searchPath()` na vyhledávání cesty.

4.6 Události

4.6.1 Události lichoběžníkové a Boustrophedon dekompozice

Pokud se zaměříme na událost IN, tak z teoretického popisu vyplývá, že tato situace nastává ve chvíli, kdy řez narazí na vrchol překážky. Protože řez je v reálné situaci reprezentován samotným vysavačem, tak událost IN vznikne ve chvíli, kdy robot pojede těsně (lze počítat s odchylkou řádově v jednotkách centimetrů) vedle překážky. V tuto chvíli vznikají dvě možnosti, jakou pozici zvolit pro nově vzniklé buňky. První možností je, že pozice, na kterých obě buňky začínají, budou stejné. Odlišnost bude pouze ve směru, kterým se vysavač vydá, až začne buňky zpracovávat (první směr doleva od bodu vzniku, druhý doprava od bodu vzniku). Tato situace by sice korespondovala s teoretickým popisem události, ale vysavač by zbytečně znovu projížděl poslední stopu z předchozí buňky. Proto byl zvolen způsob jiný, výchozí pozice buněk bude z jedné a z druhé strany překážky. Tato situace je detailněji ukázána s příkladem v části Implementace v kapitole 5.3.1.

Při události OUT jsou opět dvě možnosti, jaký bod vybrat pro novou buňku. Může se zvolit bod, ve kterém je detekována OUT událost, ale opět by robot zbytečně čistil již hotovou část z předchozí buňky. Proto byl zvolen bod jiný, který se nachází na konci místnosti. Detailní vysvětlení s obrázky lze nalézt v kapitole 5.3.2.

Událost MIDDLE lichoběžníkové metody se od teoretického popisu nikterak neliší. Pozice nové buňky bude v místě, kde se mění tvar překážky.

[5] Slepé prohledávání do šířky

4.6.2 Události při dekompozici řezem

Událost IN v tomto typu dekompozice koresponduje s událostí IN z předchozí kapitoly. Při události OUT vzniknou dvě nové buňky. Jedna se nachází vedle překážky a druhá pod ní. Směr robota při pokrývání první buňky bude opačný než v době jejího vytvoření. Na pozici druhé buňky bude robot pokračovat po vyčištění předchozích buněk (směr bude stejný jako v době vytvoření buňky).

4.7 Robot

Je vhodné, aby výsledná aplikace byla bez větších úprav použitelná jak pro demonstrační applety, tak pro reálného robota. Tím bude, školní robot ovládaný FITkitem [8]. Ideálním testovacím objektem by byl již vyrobený a používaný robotický vysavač. Snadno by se tak srovnávaly výsledky algoritmů diskutovaných v této práci a algoritmu, který je využíván komerčními robotickými vysavači. Takový robot ovšem nebyl k dispozici.

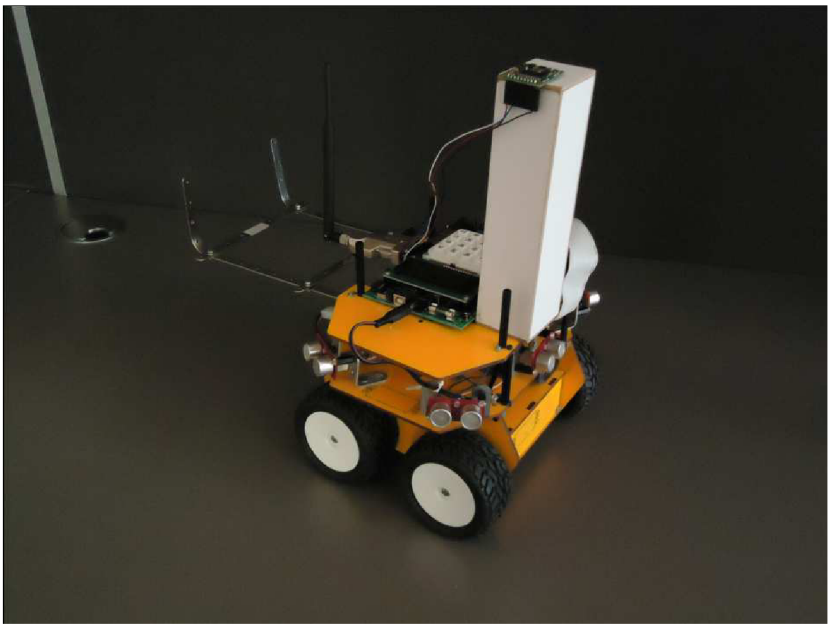
Hlavní část aplikace, samotný řídící algoritmus, bude pro obě části této práce (pro robota i applet) téměř totožný, tedy až na rozšíření, která boudou dodělána do jednoho nebo druhého řídícího algoritmu. Rozdíl bude patrný až v těch částech kódu, ve kterých se budou provádět jednotlivé příkazy pro robota. Na straně demonstračního appletu se bude jednat o vykreslovací funkce, na straně reálného robota o komunikaci s ním samotným.

5 Implementace

Nedílnou součástí této práce je implementace výše uvedených metod. Podle teoretických znalostí a návrhu řešení byla provedena implementace boustrophedon dekompozice a dekompozice řezem. Výsledkem jsou dvě aplikace. Jedna pro demonstrační applety a druhá pro skutečného robota.

5.1 Testovací robot

Jako testovací robot byl použit školní robot, který je postaven na kitu 4WD1 Robot Kit od firmy Lynxmotion [12]. Robot je rozšířen o dvě desky, na kterých je umístěna elektronika, FITkit a o sloupek, na kterém je připevněn kompas (musí být dále od motorů kvůli rušení). Na obrázku 16 je zobrazen robot, na kterém jsou algoritmy testovány.



Obrázek 16: Obrázek robota

5.1.1 Řídící jednotka

Řídící jednotkou testovacího robota je FITkit [8]. Jedná se o výukový kit pro praktické zkoušení hardwarových úkolů na Fakultě informačních technologií Vysokého učení technického v Brně. Samotné ovládání robota FITkitem není součástí této práce. Byla využita aplikace, která byla vytvořena v rámci jiné bakalářské práce [9]. Aplikace vytvořená v rámci moji práce pouze zasílá na FITkit příkazy, které má robot vykonat a přijímá data ze sonarů. Tato aplikace byla rozšířena o odometrii, která byla vytvořena v rámci projektu do předmětu Robotika na FIT VUT [11].

Komunikace s FITkitem probíhá po sériové lince, při využívání knihovny *RXTX* [10]. Využívá se textový komunikační protokol.

5.1.2 Sonary

Robot je osazen pěti ultrazvukovými sonary. Tři jsou umístěny vpředu (jeden směřuje vpřed a dva směřují na bok) a dva vzadu (oba směřují šikmo vzad). Sonary, kterými je robot osazen jsou sonary

SRF08 od firmy Devantech [15]. Počet sonarů je šest a jsou pravidelně rozmístěny po obvodu celého robota.

5.1.3 Vybavení

Dalším vybavením, kterým je robot osazen a které je využíváno, je kompas. Ten slouží pro snazší orientaci v pokrývané buňce a pro nastavení směru v nově vytvořené buňce. Robot je osazen kompasem CMPS03 od firmy Devantech [13].

Dalším využívaným zařízením je optický enkodér. Ten je používán pro měření ujeté vzdálenosti. Robot je osazen čtveřicí optických dekodérů QME-01 (Quadrature Motor Encoder) se senzorem firmy US Digital (senzor E4P-120-079-HT) [14].

Tyto dvě zařízení jsou využívány pro lokalizaci robota v prostoru. Problémem pro tento typ lokalizace jsou chyby při měření a jejich kumulace.

5.2 Ovládání demonstračního appletu

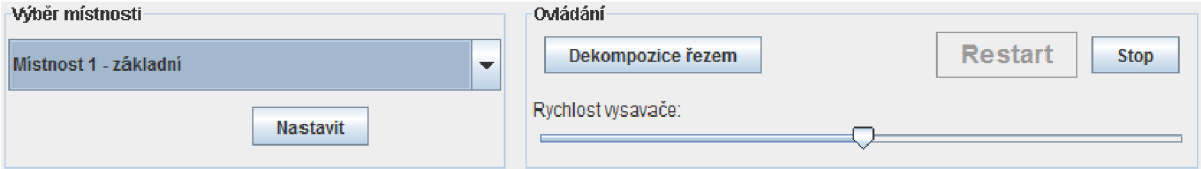
V následující kapitole bude vysvětleno ovládání demonstračního appletu. Applet byl vytvořen na základě návrhu, který byl diskutován v kapitole 4.1. Samotné GUI (grafické uživatelské rozhraní) je napsáno s využitím knihovny uživatelských prvků jazyka Java - Swingu.

5.2.1 Řídicí prvky

Řídící panel obsahuje tlačítka a další prvky pro ovládání celého demonstračního appletu. V levé části je řídící panel a combo box s nabídkou místností, tedy s nabídkou různých druhů překážek. Combo box dává na výběr ze dvou typů překážek. První druh překážek je pouze konvexní, druhý typ místnosti obsahuje i nekonvexní překážky. Každý druh obsahuje dvě úrovně překážek. Jednu jednoduchou pro snadnou demonstraci způsobu tvorby buněk a druhou náročnější pro ukázku, že si aplikace poradí i se složitějším prostředím.

V pravé části ovládacího panelu je tlačítko pro spuštění algoritmu. V demonstračních appletech je to tlačítko pro spuštění dekompozice řezem nebo Boustrophedon metody. Tato tlačítka spouští čistící proces podle jedné z výše uvedené metody. Dále pravá část panelu obsahuje tlačítko pro restartování metody (tlačítko Restart) a pro její zastavení (Stop).

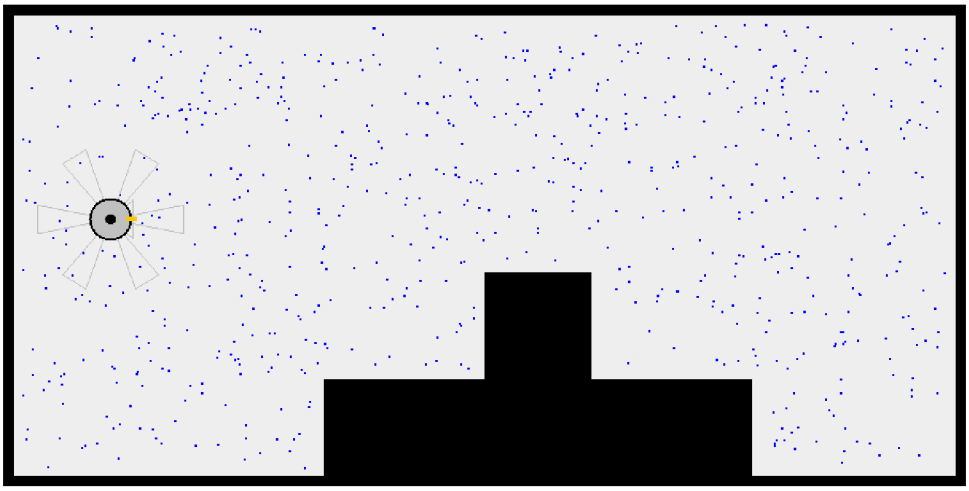
Pod těmito tlačítky nalezneme posuvník pro ovládání rychlosti vysavače. Nastavování vyšší rychlosti vysavače se nedoporučuje. Protože pokud by byla nastavená rychlost vysavače příliš vysoká, mohlo by se stát, že sonary špatně detekují, nebo vůbec nedetekují překážku. Další vykonávání algoritmu by již bylo chybné. Zvýšení rychlosti se dá využít ve chvíli, kdy chceme urychlit přejezd po dlouhé rovince. Pokud nastavíme rychlost nad určitou hranici, robot ji při příjezdu k překážce automaticky sníží na takovou, při které nebudou nastávat chyby.



Obrázek 17: Ovládací prvky demonstračního appletu

5.2.2 Místnost

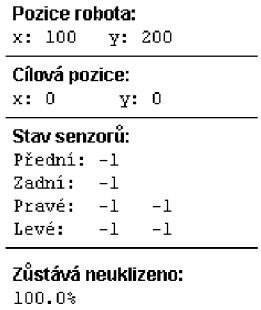
Podle zvoleného typu místnosti v levé části řídícího panelu se na panelu určeného pro zobrazení místnosti objeví překážky (obrázek 18). Na tomto panelu je také zobrazen robotický vysavač se svými sonary. Tečky, které jsou rozmístěny po celé ploše, představují smetí, které má robotický vysavač za úkol uklidit.



Obrázek 18: Místnost s překážkou a robotickým vysavačem

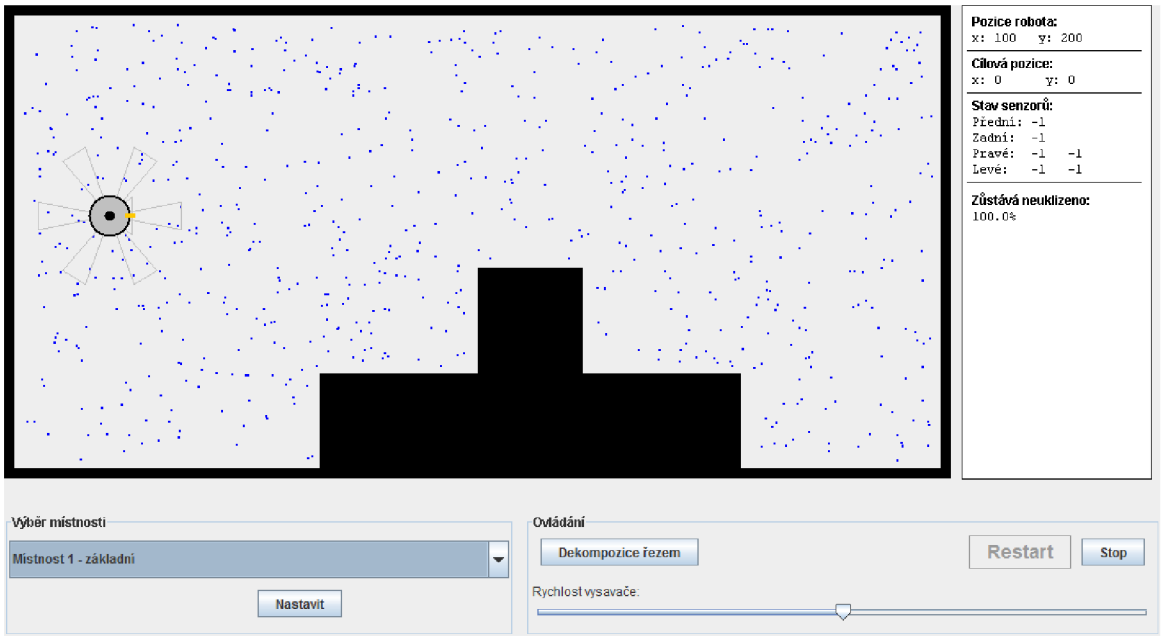
5.2.3 Stav robota

V pravém panelu jsou zobrazeny souhrnné informace o stavu robota. Lze zde nalézt stav sonarů, zda něco detekují a v jaké vzdálenosti. Pokud sonary ukazují hodnotu -1, nic nedetekují. Pokud ukazují jinou hodnotu, jedná se o vzdálenost (v pixelech) překážky od okraje robotického vysavače. Pod informacemi o sonarech je umístěn ukazatel poměru vysátého smetí (v procentech).



Obrázek 19: Panel, na kterém je v textové podobě zobrazen stav robota

Jak vypadá výsledná aplikace je ukázáno na obrázku 19. Vidíme uspořádání jednotlivých panelů, které byly popisovány v předchozím textu.



Obrázek 20: Výsledný vzhled demonstračního appletu

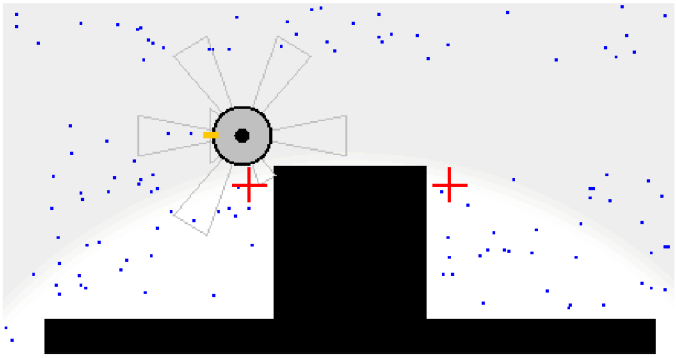
5.3 Detekce překážek

Pro správnou funkci všech použitý algoritmů je nejdůležitější správná detekce překážek. Bez té by nebylo možné správně určit, kde překážka začíná a kde končí. Problémem při detekci překážek je i menší množství sonarů a jejich samotné vlastnosti.

5.3.1 Začátek překážky

Protože se v běžné domácnosti vyskytují předměty různých tvarů, tak i detekce začátku překážky má několik možností. První možností je, že robot detekuje jako začátek překážky několik bodů v řadě za sebou (situace nastává, když jede rovnoběžně s překážkou). Nelze tedy určit jeden bod, který by udával pozici IN události. Řešením této situace je označení jako výchozí pozice dva odlišné body. Do každé nové buňky se tedy uloží jiná pozice začátku. Tato situace je znázorněna na obrázku 21.

Pro detekci jsou nejobtížnější překážky, které mají vzhledem k robotovi malou velikost. Takovou překážkou může být například noha od židle. Ve chvíli, kdy se robot nachází vedle překážky (je to moment, kdy by měla nastat událost IN), ji žádný sonar nedetekuje. Je to dáno rozmístěním a počtem sonarů, kterými je robot osazen. Proto se v této situaci postupuje obdobně jako v předchozím odstavci. První buňka se generuje ihned, jakmile jeden z předních bočních radarů zachytí překážku. Druhá se generuje v momentě, kdy na zadním bočním radaru se překážka začne vzdalovat.

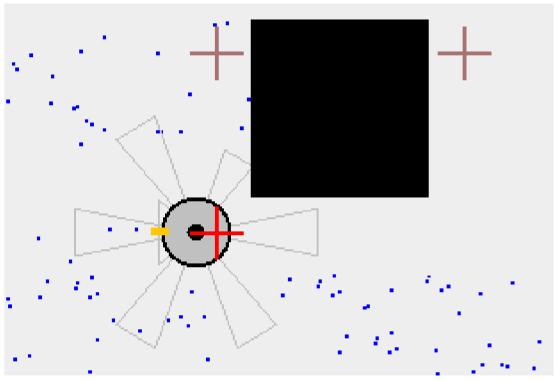


Obrázek 21: Událost IN má pro dvě nové buňky odlišné pozice naznačené červenými křížky. Pohyb robota je shora dolů

5.3.2 Konec překážky

Situace na konci překážky je obdobná situaci z předchozí kapitoly. Zde se ale projevují rozdíly mezi dekompozicí řezem a zbylými dvěma metodami.

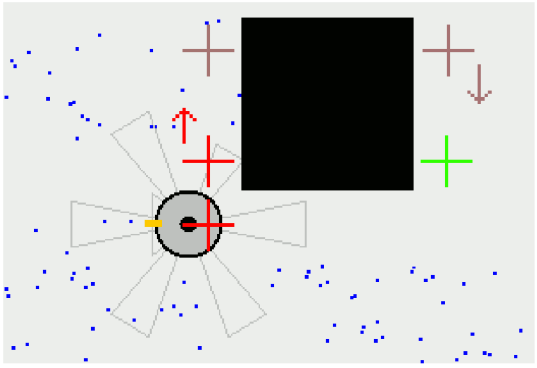
Boustrophedon metoda si při detekci konce překážky vytvoří jednu novou buňku, ve které bude robot pokračovat dál aktuálním směrem. Poté se navrátí na pozici nepokryté buňky vytvořené při IN události a pokryje i tuto druhou buňku. Situace je znázorněna na následujícím obrázku číslo 22.



Obrázek 22: Vytváření buněk na konci překážky – Boustrophedon metoda

Hnědé křížky představují buňky, které byly vytvořeny při události IN. Pravá z nich je již pokryta, levá na pokrytí teprve čeká. V následujícím kroku robot vyhledá cestu k levému křížku a provede pokrytí této buňky. Červený křížek udává pozici, na kterou robot musí zamířit po vyčištění levé buňky. Jedná se o pozici nově vytvořené buňky při události OUT.

U dekompozice řezem je situace mírně odlišná. V této situaci se nevytváří pouze jedna buňka, ale vytváří se buňky dvě tak, jak je ukázáno na obrázku. Díky tomuto způsobu tvoření buněk se pomocí dekompozice řezem odhalí i nekonvexní části překážek. Robot po dokončení jedné buňky vybírá poslední nově přidanou buňku. Nevydá se tedy na pozici události IN, ale začne ihned zpracovávat poslední přidaný bod (horní červený křížek). Jeho pohyb je ovšem v opačném směru než byl předchozí (červená šipka nahoru).



Obrázek 23: Vytváření buněk na konci překážky – Dekompozice řezem

Robot se tedy pohybuje od červeného křížku nahoru. Ve chvíli, kdy uzavře i tuto buňku a vybírá další, která je ještě nepokryta, nalezne buňku, která začíná na levém hnědém křížku. Tato buňka sice ještě nebyla pokryta, ale její počáteční pozice se již v pokrytém poli nachází (nachází se v červené buňce). Proto tento bod vyřadí. Stejně postupuje i u zeleného křížku, který se nachází v buňce pravého horního křížku.

5.4 Lokalizace

Jedním z hlavních problémů, při zprovoznění algoritmu na reálném robotovi, byla jeho lokalizace. Pro lokalizaci byl použit kompas, pomocí kterého se rozeznávalo otočení, a odometrie pro měření ujeté vzdálenosti. Pomocí kompasu a odometrie lze v ideálním prostředí (bez chyb měření) přesně určit polohu robota vůči nějakému referenčnímu bodu (např. původní pozice robota). V reálném prostředí je ale situace jiná. V následujících kapitolách budou popsány problémy, které se při zprovoznění lokalizace vyskytly a způsob, jakým byly vyřešeny.

5.4.1 Nastavení úhlu

V situaci, kdy bylo potřeba nastavit přesný úhel např. 0°, ho nebylo možné s přesností na 1° nastavit. Byl vytvořen algoritmus, který spustí otáčení robota směrem, který je nejblíže k požadovanému otočení a v případě, že robot tento úhel přetočí, je otáčení okamžitě zastaveno a dá se pokyn k otáčení druhým směrem. Tímto způsobem robot upravuje svoje natočení do té doby, dokud nenastaví požadovaný úhel. V tomto případě ale vznikal problém, v momentě kdy robot dorovnával malý úhel např. 5°. Kvůli tření nebyl robot schopen otáčet kolečky malou rychlostí a ve chvíli, kdy otáčky zvýšil, přeskočil cílový úhel na druhou stranu.

Bylo tedy nutné snížit přesnost. V prvních pokusech byla nastavena přesnost na jedno desetinné místo úhlu převedeného na radiány. Ve stupních je to tedy hodnota přibližně 5,73°. S touto přesností již bylo možné, průměrně po třech pokusech o dorovnání, cílový úhel nastavit. Tuto hodnotu považuji za nejpřesnější možnou, protože i s ní robotovi chvíli trvá, než se srovná.

Tímto zanedbáním se ovšem do měření zanesla chyba. Pokud by robot špatně nastavil úhel o 5°, přitom své natočení považoval za správné a urazil 200 cm, na konci své cesty by byl posunut o 17,43cm. Tato chyba je při ujetí pouhých 200 cm s jednou otáčkou při chybném nastavení úhlu o 5°. V reálné situaci bude samozřejmě otáček mnohem více, a pokud by se nepřesnosti sčítaly, můžeme se pohybovat s chybou měření více než 1 metr. Takováto chyba je pro reálné nasazení nepřípustná a je potřeba ji nějakým způsobem kompenzovat.

5.4.2 Kompenzace chyb

Protože robot během svého pohybu zná aktuální natočení, lze z každého přírůstku ujeté vzdálenosti vypočítat posun v osách x a y. I když robot chybně nastavil svoje natočení, tak podle dat z kompasu zle vypočítat jeho pozici. Přesnost pozice tedy závisí na přesnosti kompasu.

Další chyba, která může při jízdě vzniknout, je chyba odometrie. Pokud se bude robot pohybovat po kluzké podlaze, mohou zejména při rozjíždění nebo zastavování proklouznout kola. Tím by naměřená vzdálenost mezi otáčkami byla větší, než ve skutečnosti je. Společně s chybou při měření vzdálenosti se odchylka od skutečně ujeté vzdálenosti pohybovala okolo 2,5cm, což je v porovnání s chybou, která vzniká při nastavování otočení zanedbatelné.

Při práci s robotem bylo také třeba sledovat, zda je dostatečně nabitá baterie. Při intenzivním používání se baterie školního robota vybije poměrně rychle. Nejsnáze se tato situace pozná při otáčení. Robot se otáčí, ale zabírají kola pouze na jedné straně (na druhé straně stojí). To opět vnáší chybu do lokalizace, protože robot se neotočí kolem své osy, ale otáčí se kolem stojících kol. Výsledek je ten, že při otáčení změní svoji pozici. O této změně ale robot neví a tak předpokládá, že je pořád na stejném místě.

5.4.3 Umístění kompasu

Problémem kompasu je, že by mohl být rušen elektromagnetickým zářením, které produkují motory. Kompas byl tedy umístěn na horní plošině vedle FITkitu, aby byl dále od motorů. Zprvu se toto umístění jevilo jako výhodné, ale později se zdálo, že robot úhly měří pokaždé jinak. Byl tedy proveden test, jaká chyba při tomto umístění vzniká. Robot byl ručně položen tak, aby směřoval přesně na sever. Vycházelo se z hodnot, které kompas aktuálně ukazoval (motory byly vypnuté). Poté se robot nechal ujet cca 1 m a sledovala se změna směru. Po ujetí této vzdálenosti kompas hlásil otočení o přibližně 20°. Ve skutečnosti se ale robot otočil o maximálně 5° (odhad).

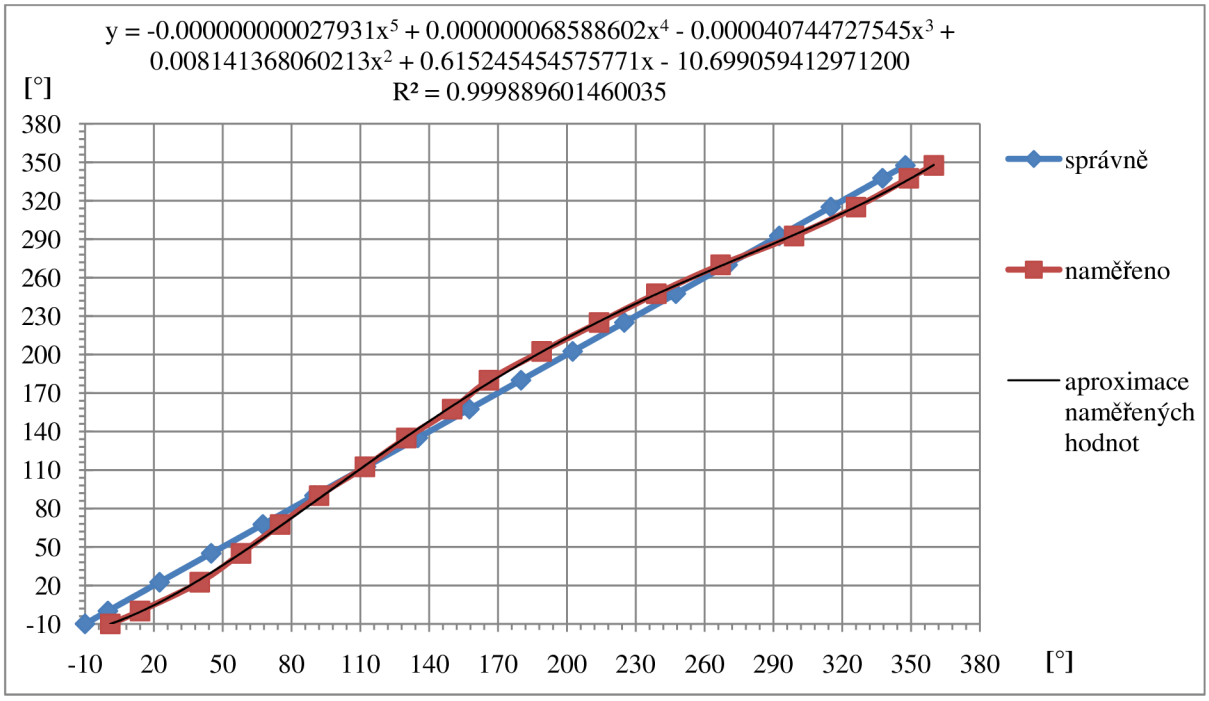
Bylo tedy nutné kompas umístit ještě dále od zdroje elektromagnetického zářeni. Kompas byl připevněn na hranol cca 15 cm nad horní plošinu. V této výšce (přibližně 25 cm nad motory) se údaje z kompasu jevily jako správné.

5.4.4 Chybná data z kompasu

Při práci s kompasem byla pozorována chyba měřených hodnot. Nezkalibrovaný (resp. kalibrovaný ve výrobě), vracel při otáčení hodnoty z rozmezí 240 – 300° [9]. V našem případě ale kompas vracel hodnoty z celého rozmezí 0 – 360°. Naměřené hodnoty ale byly chybné. Pokud se robot nasměroval tak, aby kompas ukazoval hodnotu 0 stupňů a poté se robot otočil přesně o 180°, kompas by měl vracet hodnotu 180, ale ve skutečnosti vracel hodnotu přibližně 165°. Bylo tedy nutné kompas zkalibrovat. To se ani po mnoha pokusech nepodařilo a kompas stále vracel chybné hodnoty.

Protože kompas vracel stále stejné chybné hodnoty, bylo možné provést jejich opravu alespoň softwarově. Pro zjištění chyby (deviace [19]) v jednotlivých natočeních byl robot ručně natáčen na dané úhly a zaznamenávala se naměřená a skutečná hodnota. Podle těchto hodnot byla sestavena deviační tabulka. Hodnoty byly aproximovány polynomem pátého stupně (3. stupeň byl nepřesný). Výsledná rovnice (1) je zde zaokrouhlena, méně zaokrouhlený tvar i hodnotou spolehlivosti R^2 lze nalézt v grafu číslo1.

$$y = -2{,}8 \times 10^{-13}x^5 + 6{,}9 \times 10^{-10}x^4 - 4{,}1 \times 10^{-7}x^3 + 8{,}1 \times 10^{-3}x^2 + 0{,}6x - 10.7 \qquad (1)$$



Graf 1: Průběh chybných hodnot kompasu v porovnání se správnými

V grafu číslo 1 si můžeme všimnout, že největší odchylka od správné hodnoty je 17,5° (snadněji lze vyčíst z tabulky v příloze 2). Tato chyba by způsobila velmi nepřesnou lokalizaci a tím i nefunkčnost čistících algoritmů. Při použití polynomu se hodnota odchýlí maximálně o 2,26°. Tuto odchylku již můžeme zanedbat, i když, s přihlédnutím na problémy popsané v předchozích kapitolách, se bude jednat o další zhoršení přesnosti.

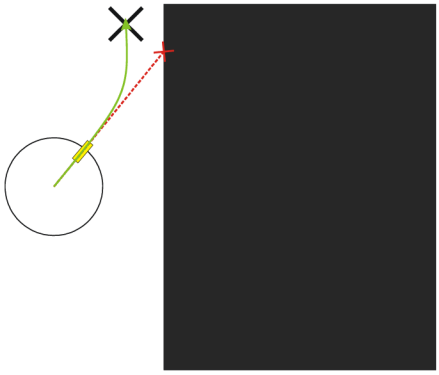
Při výpočtu správné hodnoty natočení podle rovnice (1) nebylo možné využít běžný datový typ (`double`), protože docházelo k zaokrouhlování, které velice znepřesnilo výsledek (při některých úhlech byla výsledná hodnota i o 100° jinak, než se očekávalo). Z toho důvodu musel být použitý datový typ `BigDecimal`, který hodnoty nezaokrouhluje (nebo zaokrouhluje podle námi stanovených pravidel).

Pokud by se měl robot využít k práci po delší době nečinnosti, je velice důležité zkontrolovat, jaké hodnoty kompas vrací a v případě potřeby aktualizovat deviační tabulku.

5.4.5 Hledání cesty

Kvůli chybám, které vznikají při lokalizaci robota, je stižené i hledání cesty pro přejetí do cílové buňky. Algoritmus vypočítá body, které musí robot projet, aby našel cílovou pozici a nenastala kolize s některou z překážek. Problémem je, že některé průjezdové body se mohou nacházet v těsné blízkosti překážky. Potom by i drobná chyba v lokalizaci mohla způsobit kolizi s překážkou.

Byl tedy vytvořen softwarový modul Watcher, který je vložen mezi řídící jednotku a jednotku, která vysílá informace ze sonarů. Při čistícím procesu je tento modul vypnut, resp. přijatá data posílá dále nemodifikována řídící jednotce. Ve chvíli, kdy má robot projet vytyčené body, se tento modul spustí a kontroluje data ze sonarů, zda se robot neblíží k překážce. Pokud hrozí kolize s překážkou, modul začne robotovi posílat signály k zatočení směrem od překážky, jízdu ale nezastavuje.



Obrázek 24: Práce modulu Watcher – kvůli chybné lokalizaci by robot při hledání průjezdového bodu (černý křížek) narazil do překážky (červená cesta), modul Watcher cestu upraví, aby robot nenarazil (zelená cesta).

5.5 Velikost kroku

Velikostí kroku se v tomto kontextu myslí vzdálenost, o jakou se posune robot při otáčce. Při zvažovaní, jakou konstantu zvolit je třeba brát v úvahu několik faktorů. První byl již zmíněn v kapitole 2.2.3, ve které se diskutovala velikost buňky přibližného rozkladu. V implementovaných algoritmech je význam buňky jiný než ve výše uvedené kapitole, ale problém s nepokrytými místy u stěn zde přetrvává. Zmenšením kroku sice robot zanechává menší nepokrytou plochu, ale za cenu snížení efektivnosti (zvýší se doba úklidu). Zanechávání nepokrytých míst u stěn ale bylo překryto daleko výraznějším problémem, diskutovaném v následujícím odstavci.

S ohledem na chyby, které vznikají při lokalizaci (5.4.1), byla velikost kroku volena na polovinu šířky robota i za cenu delší doby jízdy. Protože robot může jet při jízdě pod mírně jiným úhlem, než předpokládá, a vznikaly by při vysávání nepokrytá místa tam, kde se úhel mezi dvěma jízdami otevíral. Díky tomuto malému kroku se ale vedlejší stopy překrývají a tím se alespoň částečně zamezuje velkým nepokrytým plochám. Nelze ovšem s určitostí říct, že žádná nepokrytá místa mezi stopami nevzniknou. Protože robot může ujet i poměrně větší vzdálenost, odchylka může být více než je šířka robota a tím vznikne nepokrytá část mezi stopami. Toto je bohužel daň za nepřesnou lokalizaci.

Jako možné rozšíření současného stavu práce, by se po vzoru komerčních vysavačů nenavštěvovalo každé místo pouze jedenkrát, ale navštívilo by se vícekrát (*iRobot Roomba* navštíví každé místo 4 krát[6]). Neboť je chyba pokaždé jiná, je nepravděpodobné, že by na stejném místě vznikla čtyřikrát stejná chyba, takže se vyčistí i ta místa, která byla vynechána v předchozích krocích. Již na první pohled je vidět, že čas na úklid bude oproti implementovanému způsobu několikanásobně delší. Proto by bylo vhodné využít oba způsoby a nabídnout uživateli vysavače možnost vybrat si z rychlého úklidu, nebo z úklidu důkladného.

[6] Podle plakátu [16] a rozhovoru s majitelem robotického vysavače *iRobot* [19]

6 Testování a vyhodnocení

6.1 Demonstrační applety

Demonstrační applety jsem testoval na místnostech s překážkami, které jsou dostupné v combo boxu s výběrem místností. Pozoroval jsem, že pokud byl více zatížený počítač, vysavač v demonstračním appletu nedetekoval překážku a pokračoval skrz ni nebo i vyjel z místnosti. Tato skutečnost je dána tím, že detekování překážky v sonaru je výpočetně náročnější operace. Do budoucna by bylo vhodné se zaměřit na tuto část a pokusit se detekci překážky optimalizovat.

Applet s algoritmem založeným na Dekompozici řezem pro nabízené místnosti dosahuje poměrně dobrých výsledků. Nutno podotknout, že se jedná o ideální prostředí, ve kterém se nevyskytují chyby měření a chybné určování pozice. Ovšem i v takto ideálním prostředí se stává, že robot špatně vyhodnotí situaci a udělá chybu, která ho může zmást natolik, že nedokáže dále pokračovat.

6.1.1 Experimenty

Provedl jsem několik experimentů s demonstračním appletem, abych zjistil, jakých výsledků je algoritmus schopen dosáhnout. Zaměřil jsem se pouze na applet demonstrující činnost Dekompozice řezem, protože jiné metody nejsou schopné detekovat a pokrýt nekonvexní části překážek. Výsledky by tedy nebylo možné objektivně porovnat.

V příloze 3 jsou uvedené tabulky s experimenty pro jednotlivé místnosti. Můžeme si všimnout, že pro jednodušší prostředí (místnost 1 a místnost 3) algoritmus dosahuje poměrně dobrých výsledků a dobré úspěšnosti. U složitějších prostředí (místnost 2 a místnost 4) je výsledek čištění také dobrý, ale je zde větší množství nedokončených pokusů. To je dáno tím, že prostředí obsahuje větší množství překážek a je tedy větší pravděpodobnost, že robot některou z nich špatně detekuje. Úspěch či neúspěch pokusu také závisí na zatížení procesoru, na kterém applet běží. Protože jednotlivé moduly aplikace běží jako samostatná vlákna, může plánovač procesoru při velké zátěži jedno z nich plánovat méně často. Tento problém se týká hlavně řídícího algoritmu tedy modulu `Controler`, který řídí zastavování a spouštění jízdy.

6.2 Reálný robot

Protože nepřesná lokalizace zanáší do výpočtu pozice chybu, bylo nutné zavést omezení a podmínky, za kterých bude robot schopný pokrýt vytyčený prostor. Pomocí papírových krabic jsem vytvořil místnost pro pohyb robota. Stěny jsem postavil tak, aby byly přibližně kolmé na směr jeho jízdy. Tento vytyčený prosto jsem se snažil postavit co nejdále od zdrojů elektromagnetického záření, které by rušilo kompas.

6.2.1 Místnost bez překážek

Nejprve se podařilo zprovoznit robota tak, aby projel místo vedle místa v prostoru, ve kterém nejsou žádné překážky. V takto jednoduchém prostředí nepotřebuje natolik přesnou lokalizaci, protože se zde nenachází více buněk, kterými by projížděl při hledání cesty. V této části bylo nutné se zaměřit na sonary. V bakalářské práci [9] všechny sonary vysílaly zvukový signál naráz. Stávalo se tedy, že

některý zachytil signál od jiného. Kvůli tomu robot zastavoval, když jel těsně kolem překážky, protože přední sonar zachytával signál od bočního. Vysílání signálů jsem tedy upravil tak, že se nejdříve vyšle signál z předního a z bočních sonarů umístěných vzadu (je jen malá pravděpodobnost, že by přední sonar zachytil signál některého zadního sonaru). Po určité době (65 ms) se teprve vyšlou signály ze zbylých sonarů, tedy z bočních umístěných vpředu.

Dalším problémem sonarů je šum v jejich výstupu, kdy na malou chvíli detekují překážku, která před nimi ve skutečnosti není. Abych tento problém alespoň částečně odstranil, vytvořil jsem jednoduchý filtr, který vrací průměrnou hodnotu ze tří předchozích měření. Tímto se částečně odstraní výkyvy v naměřených hodnotách. Místo průměru by bylo možné použít medián, který by vracel prostřední hodnotu ze třech měření.

6.2.2 Místnost s překážkou

V dalším postupu jsem se pokusil o zprovoznění algoritmu v prostředí s jednoduchou překážkou. Protože v předchozím kroku se vyskytly problémy s nepřesnou lokalizací, jejichž řešení bylo časově náročné a výsledek nebyl nikterak přesný, nepodařilo se tento úkol dokončit. Když jsem se zaměřil na dílčí kroky při detekci překážky (IN, OUT událost nebo detekce konce překážky), robot je správně rozpoznával a choval se podle očekávání. Jako celek (při používání lokalizace) ovšem správně nefungoval.

Abychom zajistili uspokojivé výsledky práce robota, bylo by nutné použít další a spolehlivější senzory. Mohlo by se např. využít inerciální jednotky, která plynule snímá zrychlení ve třech směrových osách. Postupnou integrací zrychlení lze získat ujetou vzdálenost [18]. Společně s již dostupným vybavením by se mohlo jednat o znatelné zlepšení určování polohy robota.

Další možností by bylo použití navigace založené na orientačních bodech. Protože v současné době je robot pro detekci objektů v okolí osazen pouze sonary, které jsou pro tuto činnost nevhodné (nedokáží určit polohu objektu, ale pouze vzdálenost), nelze tuto metodu použít. Opět by se muselo přidat další zařízení (CCD kamerový nebo jiný vizuální systém) pro detekci orientačních bodů [18].

Závěr

Výsledkem práce je algoritmus, který dokáže řídit robota v neznámém prostředí. Z důvodu nepřesností, které vznikají při měření a nastavování otočení či ujeté vzdálenosti, bylo nutné zavést jisté podmínky, za kterých je robot schopen pracovat. Pokud bychom požadovali lepší výsledky, museli bychom robota osadit dalšími součástkami, které by zlepšily jeho orientaci v prostoru např. inerciální jednotka pro přesnější měření ujeté dráhy a kamera pro detekci orientačních bodů.

Vytvořený řídící algoritmus je využíván ve dvou aplikacích. V demonstračním appletu, kde se ukazuje jeho činnost a v aplikaci, která řídí reálného robota. Při vytváření druhé části byla pro samotné ovládání robota využita a rozšířena práce [9], která se zabývá řízením robota pomocí FITkitu.

Vytvořil jsem také aplikaci, pomocí které lze ověřit, zda jsou veškeré součástky, kterými je robot osazen, funkční a jestli jsou i hodnoty, které vrací, správné. Tato pomocná aplikace je napsaná stejně jako řídící algoritmy v jazyce Java. Protože nebyla cílem této práce, její popis zde není uveden.

Pro budoucí práci se ukazuje několik cest. Jednou z nich je již v prvním odstavci popsané rozšiřování robota o další součástky a s jejich pomocí vytvoření přesnější lokalizace. Pro snadnější orientaci v prostoru by bylo možné vytvářet mapu prostředí, ve kterém se robot pohybuje. Se současným vybavením robota se jeví jako vhodná technika *Simultánní lokalizace a mapování (SLAM)* [20].

Další možnou cestou by bylo vytvoření několika režimů činností např. objíždění překážek nebo spirálovitý pohyb. Pro toto pokračování mého projektu by bylo ovšem nutné vytvořit přesnou lokalizaci.

Literatura

[1] CHOSET, Howie, et al.: *Principles of robot motion: theory, algorithms, and implementation*. 2005. Cambridge (Massachussets): The MITPress, 2005. Cell Decomposition, s. 161-185. ISBN 0-262-03327-5.

[2] CHOSET, Howie; PIGNON, Philippe The Boustrophedon Cellular Decomposition. In *Coverage Path Planning: The Boustrophedon Cellular Decomposition* [online]. Pittsburg: International Conference on Field and Service Robotics, 1997 [cit. 2010-11-20]. Dostupné z WWW: <http://www.ri.cmu.edu/publication_view.html?pub_id=1416>.

[3] CHOSET, Howie.: "Coverage for robotics - A survey of recent results," *Annals of Mathematics and Artificial Intelligence*, Vol. 312001, pp. 113 - 126.

[4] *Robotika.cz* [online]. [cit. 2011-05-12]. Dostupné z WWW: <www.robotika.cz>.

[5] WONG, Sylvia.: *Qualitative Topological Coverage of Unknown Environments by Mobile Robots* [online]. New Zealand, 2006. iii, 203 s s. Dizertační práce. The University of Auckland, Department of Electrical and Computer Engineering. Vedoucí práce Dr Bruce A. MacDonald, Dr George Coghill. Dostupné z WWW: <http://www.researchgate.net/publication/39994066_Qualitative_Topological_Coverage_of_Unknown_Environments_by_Mobile_Robots>.

[6] ZBOŘIL, František.: *Základy umělé inteligence IZU: Studijní opora* [online]. Fakulta informačních technologií: Fakulta informačních technologií, 2006. 142 s. Studijní opora. Vysoké učení technické v Brně, Fakulta informačních technologií.

[7] ZELINSKY, A.; JARVIS, R.; BYRNE, J. C. & YUTA, S., *Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot*, In Proceedings of International Conference on Advanced Robotics, 1992, pp. 533 - 538

[8] FUČÍK, Otto, et al.: *FITkit* [online]. [cit. 2011-03-30]. FITkit. Dostupné z WWW: <http://merlin.fit.vutbr.cz/FITkit>.

[9] NOVOTNÝ, Tomáš.: Řízení robota pomocí FITkitu, bakalářská práce, Brno, FIT VUT v Brně, 2008

[10] *Rxtx* [online]. 1998 [cit. 2011-04-26]. The Prescription for Transmission. Dostupné z WWW: <http://users.frii.com/jarvi/rxtx/index.html>.

[11] KREMPA, Petr.: Odometria robota s pomocou FPGA, projekt do předmětu Robotika, Brno, FIT VUT v Brně, 2010

[12] *Ineltronics* [online]. 2011 [cit. 2011-05-10]. 4WD1 Robot Kit (Basic No Electronics). Dostupné z WWW: <http://www.ineltronics.com/index.php?main_page=product_info&products_id=365> .

[13] *Robot – electronics* [online], [cit. 2011-26-04]. CMPS03 documentation. Dostupné z WWW:<http://www.robot-electronics.co.uk/htm/cmps3doc.htm>.

[14] *Lynxmotion - Quadrature Motor Encoder w/Cable*. [online], [cit. 2011-26-04]. Dostupné z WWW:<http://www.lynxmotion.com/product.aspx?productid=448>

[15] *Robot – electronics* [online], [cit. 2011-26-04]. SRF08 Ultra sonic range finder. Dostupné z WWW: < http://www.robot-electronics.co.uk/htm/srf08tech.shtml>.

[16] *Robot* [online]. 2009 [cit. 2011-04-26]. IAdapt – nová technologii úklidu. Dostupné z WWW: <http://www.irobot.cz/file/tiskove-zpravy/1293021817_iadapt.pdf>.

[17] *Society of Robots* [online]. 2005 [cit. 2011-04-30]. Sharp IR Range Finder. Dostupné z WWW: <http://www.societyofrobots.com/sensors_sharpirrange.shtml>.

[18] ORSÁG, Filip.: *Robotika ROB: Studijní opora*. Fakulta informačních technologií: Fakulta informačních technologií, 2006. 128 s. Studijní opora. Vysoké učení technické v Brně, Fakulta informačních technologií

[19] DITTRICH, Petr.: Student doktorského studia; člen výzkumné skupiny STRaDe. Ústní sdělení.

[20] *OpenSLAM* [online], [cit. 2011-05-12]. Dostupné z WWW: <www.openslam.org>.

Seznam příloh

Příloha 1. Obsah přiloženého CD

- `xsojka01_BP/`
  - Tato písemná zpráva ve formátu PDF a ve zdrojovém tvaru
- `applet/slice_decomposition.zip`
  - Zdrojové soubory demonstračního appletu s algoritmem Dekompozice řezem
- `applet/boustrophedon.zip`
  - Zdrojové soubory demonstračního appletu s algoritmem Boustrophedon
- `real_robot/FITkit_app.zip`
  - Zdrojové soubory aplikace pro ovládání na straně FITkitu
- `real_robot/computer_app.zip`
  - Zdrojové soubory aplikace, která vzdáleně ovládá robota
- `documentation/`
  - Programová dokumentace, návod k instalaci
- `videos/`
  - Videa zachycující práci robota
- `photos/`
  - Fotografie robota

Příloha 2. Deviační tabulka

Skutečné natočení [°]	Naměřeno [°]	Chyba [°]	Opraveno [°]	Chyba opravy [°]
350.0	1.0	11.0	-10.08	0.08
0.0	14	14.0	-0.60	0.60
22.5	40	17.5	24.50	2.00
45.0	58	13.0	45.18	0.18
67.5	75	7.5	66.15	1.35
90.0	92	2.0	87.81	2.19
112.5	112	0.5	113.39	0.89
135.0	130	5.0	135.91	0.91
157.5	150	7.5	159.86	2.36
180.0	166	14.0	177.96	2.04
202.5	189	13.5	202.10	0.40
225.0	214	11.0	225.81	0.81
247.5	239	8.5	247.15	0.35
270.0	267	3.0	269.09	0.91
292.5	299	-6.5	293.41	0.91
315.0	326	-11.0	315.30	0.30
337.5	349	-11.5	336.58	0.92
347.5	359.9	-12.4	347.95	0.45

Tabulka 1: Deviační tabulka s vypočítanou chybou, opravenou hodnotou a s chybou opravy pro jednotlivá natočení

Příloha 3. Tabulka experimentů s demonstračním apletem (Dekompozice řezem)

Číslo	Chybných buněk	Úspěšné dokončení	Nevyčištěno [%]
1	1	ANO	1.3
2	0	ANO	1.2
3	0	ANO	1.4
4	0	ANO	0.4
5	1	NE	0.9
6	0	ANO	1.1
7	0	ANO	1.0
8	0	ANO	1.1
9	1	ANO	1.5
10	0	ANO	0.6
11	0	ANO	0.9
12	0	ANO	1.0
13	0	ANO	1.2

Tabulka 2: Tabulka experimentů – Místnost 1.

Číslo	Chybných buněk	Úspěšné dokončení	Nevyčištěno [%]
1	#	NE	31
2	#	NE	16
3	0	ANO	1
4	0	ANO	0.6
5	1	ANO	0.9
6	#	NE	14.3
7	0	ANO	0.9
8	0	ANO	1.5
9	0	ANO	1.3
10	0	ANO	1.1
11	0	ANO	0.8
12	0	ANO	1.7
13	0	ANO	1.3

Tabulka 3: Tabulka experimentů – Místnost 2. (# ve druhém sloupci značí, že počet chybných buněk nelze určit)

Číslo	Chybných buněk	Úspěšné dokončení	Nevyčištěno [%]
1	0	ANO	1.1
2	0	ANO	1.6
3	0	ANO	1.3
4	#	NE	35
5	0	ANO	0.7
6	0	ANO	0.6
7	0	ANO	0.4
8	0	ANO	1
9	0	ANO	0.9
10	0	ANO	1.4
11	0	ANO	1.1
12	0	ANO	0.9
13	0	ANO	1.2

Tabulka 4: Tabulka experimentů – Místnost 3. (# ve druhém sloupci značí, že počet chybných buněk nelze určit)

Číslo	Chybných buněk	Úspěšné dokončení	nevyčištěno [%]
0	0	ANO	1.5
1	0	ANO	1.7
2	0	ANO	2.3
3	#	NE	19.6
4	#	NE	7.3
5	0	ANO	2.1
6	0	ANO	2.3
7	0	ANO	2.3
8	0	ANO	1
9	0	ANO	2.9
10	0	ANO	1.6
11	#	NE	16.4
12	0	ANO	2.1
13	0	ANO	2.3

Tabulka 5: Tabulka experimentů - Místnost 4. (# ve druhém sloupci značí, že počet chybných buněk nelze určit)