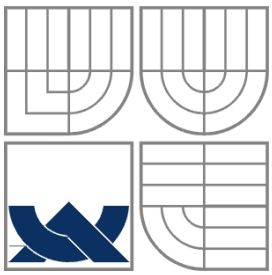


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

NAVIGACE MOBILNÍHO ROBOTU POMOCÍ FUZZY LOGIKY

MOBILE ROBOT NAVIGATION BY MEANS OF FUZZY LOGIC

DIPLOMOVÁ PRÁCE
DIPLOMA THESIS

AUTOR PRÁCE **BC. ALEŠ JANOVEC**
AUTHOR

VEDOUCÍ PRÁCE **RNDR. JIŘÍ DVOŘÁK, CSC.**
SUPERVISOR

BRNO 2010

ZADÁNÍ ZÁVĚREČNÉ PRÁCE

ABSTRAKT

V úvodní části diplomové práce je provedena analýza metod, jež se využívají k navigaci mobilních robotů. Hlavní část diplomové práce tvoří návrh systému řízení mobilního robotu. Řídicí systém robotu je založen na fuzzy modelování. Za účelem testování daného řídicího systému bylo vytvořeno simulační prostředí v jazyce C#, ve kterém byly následně provedeny experimenty.

ABSTRACT

In the introductory part of this thesis there is an analysis of methods, which are used for navigation of mobile robots. The main part of the thesis contains a proposal of control system of mobile robot. Robot control system is based on fuzzy modeling. To test the control, a system simulation environment is created in C #, in which experiments were performed.

KLÍČOVÁ SLOVA

Mobilní robot, navigace, fuzzy logika, C#, simulační prostředí.

KEYWORDS

Mobile robot, navigation, fuzzy logic, C#, simulation environment.

PODĚKOVÁNÍ

Tímto bych rád poděkoval RNDr. Jiřímu Dvořákovi, CSc. za cenné podněty a rady, které mi velmi pomohly při tvorbě mé diplomové práce.

Obsah:

1 ÚVOD

Ačkoliv bylo dosaženo v oblasti robotiky značných pokroků, zůstává nadále mnoho témat, kde se nabízí prostor pro další inovace. Jedním z velice významných faktorů, které mají zásadní vliv na robotiku, je navigace mobilního robotu. V dnešní době je využíváno mnoho přístupů založených na nejrůznějších technikách, jež jsou často inspirovány i samotnou přírodou. Roboty se tak mohou v určitých situacích chovat jako mravenci, kteří se snaží nalézt potravu, či využívat mechanismy, jež jsou založeny na pochodech probíhajících v biologických neuronových sítích. Cíl zůstává ve všech případech stejný. Dosáhnout pokud možno inteligentního chování, které by mohlo konkurovat lidskému. V oblasti navigace jde především o problém, kdy se robot musí přesunout z počáteční pozice do cílové. Robot v závislosti na svém řídicím systému musí být schopen reagovat na překážky a to nejen na statické, ale v reálném světě také na dynamické. V mnoha případech je tak nemožné spolehnout se na informace, kterými robot disponoval před zahájením samotného procesu navigace. Je třeba získávat co možná nejkvalitnější informace o okolí a následně je vhodným způsobem využít.

Člověk je vybaven dokonalým mechanismem získávání informací z okolí. Nicméně nezřídka nastávají situace, kdy je donucen učinit rozhodnutí na základě neúplných, či nepřesných informací. Tento proces by bylo zajisté vhodné využít i v oblasti navigace a právě fuzzy logika je matematický aparát, jenž umožňuje činit rozhodnutí v nejistých situacích.

Hlavním cílem práce je tedy návrh systému řízení mobilního robotu, jenž bude založen na fuzzy logice. Nejprve byly popsány dosavadní přístupy využívané při navigaci a plánování dráhy mobilních robotů. Aby bylo možné tento systém otestovat, bylo třeba vytvořit simulační prostředí, ve kterém byly následně provedeny ověřovací experimenty na jejichž základě bylo provedeno vyhodnocení navržených řídicích systémů.

2 ROBOT

Slovo robot se poprvé objevilo ve hře Karla Čapka RUR v roce 1920, nicméně podstata, kterou vystihuje, je stará jako lidstvo samo. Přístroje, jež by dokázaly napodobit člověka či usnadnit nebo nahradit jeho práci, byly snem již našich předků. V první polovině 20. století se již sny začínají měnit ve skutečnost a vznikají například zařízení pro manipulaci s radioaktivními prvky, spadající do oblasti teleoperátorů. První průmyslový robot byl do provozu uveden roku 1961 firmou General Motors a již za šest let poté byl postaven mobilní robot Shakey vybavený viděním [7].

V dnešní době si již robotika nalezla cestu do nejrůznějších oblastí lidské činnosti. Svoji pevnou pozici má v průmyslové výrobě např. při svařování či nanášení barev. Nezastupitelnou roli mají roboty i při průzkumu pro člověka nedosažitelného prostředí, jakým může být dno oceánu či povrch Marsu. Ať už se s nimi setkáváme jako se spolehlivými pomocníky lékařů při minimálně invazivní chirurgii či s fotbalisty snažícími se ve koordinovaných skupinách dostat míč do brány, je zřejmé, že tato vědní disciplína má velikou budoucnost.

2.1 Definice robotu

Jak tedy vlastně můžeme robot definovat? Existuje celá řada definic robotů. Nicméně ve snaze poskytnout všeobecně přijatelnou definici stanovila Mezinárodní organizace pro standartizaci definici robotu v normě ISO 8373, kde je robot definován jako [7]:

„automaticky řízený, opětovně programovatelný, víceúčelový manipulátor pro činnost ve třech nebo více osách, který může být buď upevněn na místě nebo mobilní k užití v průmyslových automatických aplikacích“

Tato definice se využívá v různých zemích, avšak Amerika či Japonsko, jakožto důležité země ve vývoji robotů, definují robot dle vlastních definic. Nejlépe vystihl otázku definice robotu průkopník průmyslové robotiky J. Engelberger, který poznamenal [7]:

„Neumím definovat robot, ale poznám ho, když ho uvidím.“

Robot je tedy zařízení, které je schopné provádět naprogramované úlohy, ať už na základě pevně definované posloupnosti příkazů či vlastního uvažování. Lze je dělit podle toho, zda mohou měnit svoji pozici na [5]:

- *Stacionární roboty*: zástupci této kategorie se nemohou v prostředí volně přemisťovat. Jejich pohyb je omezen v závislosti na typu jejich mechanických vazeb. Mezi tyto roboty patří průmyslové manipulátory, jejichž mechanická část je nejčastěji tvořena ramenem, zápěstím a chapadlem.

- *Mobilní roboty*: jejich pozice není omezená mechanickou vazbou k okolí a pokud jim to dané prostředí umožňuje, mohou se v závislosti na své mechanické konstrukci libovolně pohybovat. Podle prostředí, ve kterém se vyskytují, je dále můžeme dělit na roboty pohybující se po pevnině, ve vodě, ve vzduchu či ve vesmíru.

2.2 Rozdělení podvozků

Zásadním prvkem mobilního robotu, který ovlivňuje jeho celkové možnosti, je typ podvozku. Podvozky je možné dělit na kolové, pásové a kráčející platformy. V dalším budou probrány nejznámější typy kolových podvozků, jelikož tyto podvozky patří mezi nejčastěji využívané [5].

- *diferenciální kolové podvozky:* Často využívaná varianta patřící mezi nejjednodušší a nejlevnější. Podvozek je tvořen dvěma aktivními koly s jedním stupněm volnosti a dále jsou

zapotřebí stabilizační body. Ty mohou být realizovány například třecími elementy nebo dalšími koly nejlépe se dvěma stupni volnosti. Roboty s tímto podvozkem disponují výbornou manévrovatelností a jsou schopny se otáčet na místě. Hlavní nevýhodou je neschopnost překonávat vyšší překážky, čímž jsou předurčeny především pro použití uvnitř budov.

- *synchronní kolové podvozky:* V případě synchronního podvozku zajišťují pohyb tři kola se dvěma stupni volnosti, která jsou uspořádána do tvaru rovnostranného trojúhelníku. Všechna tři kola jsou natočena a zároveň se otáčejí stejným směrem stejnou rychlostí. Podvozek disponuje stejně jako předchozí typ dobrou manévrovatelností, ovšem také není vhodný na nerovné plochy.

- *trojkolové podvozky s řízeným předním kolem:* Tento typ podvozku, jak již název napovídá, je tvořen dvěma zadními koly, která zpravidla zajišťují pohyb a jedním předním kolem, jenž svým natočením udává směr pohybu. Na rozdíl od předchozích variant je možné tento typ používat i v těžším terénu a navíc umožňuje využití relativně levné a jednoduché odometrie. Nevýhodou ovšem je neschopnost otáčet se na místě.

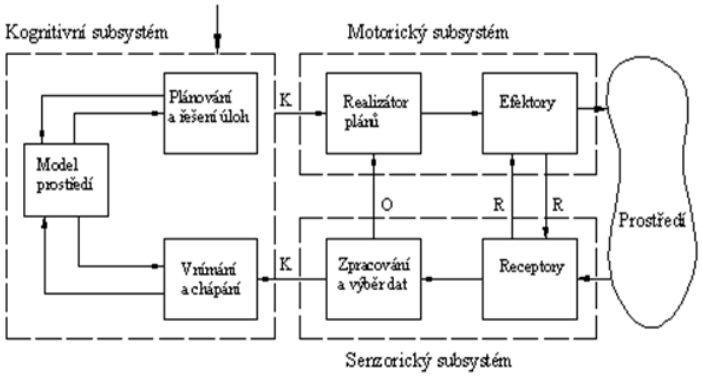
- *ackermanův podvozek:* Známé uspořádání kol z automobilů bývá v robotice využíváno především ve variantě poháněných zadních kol. Přední kola jsou natáčena podle směru pohybu a to tak, že vnitřní kolo je natáčeno více, jelikož opisuje menší poloměr. Hlavní nevýhodou je opět neschopnost otáčet se na místě.

- *podvozky se všesměrovými koly:* Všesměrové mobilní platformy je možné zkonstruovat díky použití kol, která se mohou otáčet ve dvou osách. Princip spočívá v tom, že hlavní kolo, pohybující se v jedné ose, je po obvodu tvořeno válečky, které zajišťují pohyb v druhé ose. Vhodným uspořádáním podvozku je pak umožněn pohyb všemi směry. Nevýhodou je, že je prakticky nemožné použít odometrii, jelikož princip pohybu je v podstatě založen na prokluzu kol a ani nasazení tohoto typu podvozku v těžkém terénu není možné.

2.3 Autonomní robot

Roboty, které mají pevně stanovenou posloupnost řídicích příkazů, jsou odkázány pouze na práci ve známém prostředí. V tomto prostředí jsou veškeré změny realizovány pouze působením robotu a na neočekávané jevy nedokáže robot reagovat. Do této kategorie spadají např. již zmíněné manipulátory. Autonomní robot na rozdíl od předchozího typu robotu je schopen pracovat v neznámém prostředí a inteligentně reagovat na změny v tomto prostředí [5].

O možnost fyzicky ovlivňovat prostředí či se v něm pohybovat se stará motorický subsystém prostřednictvím efektorů. Vnímání prostředí zajišťuje senzorický subsystém. Těmto podsystémům je nadřazen kognitivní subsystém, ve kterém probíhá rozhodovací a řídicí činnost.



Obr. 1 Blokové schéma obecného robotu[5].

2.4 Řídicí systémy

U autonomních robotů je možné při procesu navigace, čímž je myšleno přemístění robotu z počáteční pozice do cílové, využívat třech základních typů řídicího systému [18].

Deduktivní řídicí systém

Principem této metody je analýza mapy prostředí robotu, na jejímž základě je nalezena cesta, která je poté robotem sledována. Je tedy zapotřebí získat mapu prostředí. Tu je možné nahrát přímo do paměti robotu nebo je možné použít mapovací metody, díky nimž si robot sám vytváří vlastní mapu. Metody hledání cest jsou již natolik propracované, že pokud cesta opravdu existuje, tak je nalezena. Tento přístup je však odkázán především na statické prostředí, jelikož získání cesty může býti relativně časově náročné a v průběhu výpočtu by už informace využívané k tvorbě cesty nemusely být aktuální.

Reaktivní řídicí systém

Základem tohoto přístupu je úzká vazba mezi senzory a efektory. Řídicí program pracuje ve smyčce s konstantní délkou kroku. Nejprve získá informace od senzorů, které následně zpracuje a ihned na ně reaguje. Program, jenž analyzuje data a rozhoduje, jak se zachovat v určité situaci, nesmí být příliš složitý. Nejčastější situace řešené tímto přístupem jsou sledování stěny, průjezd koridorem či vyhýbání se překážkám. Hlavní nevýhodou je ovšem absence kompletní znalosti prostředí, což může například způsobit uvíznutí ve slepé uličce. Tudíž je tento přístup vhodný pouze pro jednoduché úlohy.

Kombinovaný řídicí systém

Jak již název napovídá, je tento systém kombinací dvou předchozích přístupů. Typicky pracuje ve dvou vrstvách, přičemž vyšší vrstva odpovídající deduktivnímu řídicímu systému má na starost naplánování dráhy, která je následně předávána ve formě podcílů vrstvě nižší. Ta je tvořena reaktivním řídicím systémem a zajišťuje dohled nad dynamickými změnami prostředí. Tím se odstraňují nevýhody samostatně pracujících systémů. Vzhledem k tomu, že robot má komplexnější informace o okolí, eliminuje tím například možnost uvíznutí ve slepé uličce.

2.5 Pracovní prostor

Prostor, ve kterém se robot pohybuje se nazývá pracovní prostor W a lze reprezentovat jako N-rozměrný euklidovský prostor R^N, kde N = 2, 3. V tomto pracovním prostoru se mohou nalézat překážky, jež omezují pohyb robotu. Aby bylo možné v tomto pracovním prostoru vhodně matematicky popsat pozici a orientaci robotu je zaveden konfigurační prostor C [19].

2.6 Konfigurační prostor

Konfigurační prostor je množina všech konfigurací robotu, ve kterých se může robot nacházet. Vektor q jednoznačně reprezentuje svými složkami orientaci robotu v konfiguračním prostoru. Je také třeba rozlišit mezi pozicí, která je pro robot dosažitelná a pozicí, která je vlivem překážek v prostředí nebo jiných omezujících podmínek pro robot nedosažitelná. Takový konfigurační prostor, který obsahuje pouze dosažitelné stavy robotu se nazývá volný konfigurační prostor (Cfree) [19].

3 METODY NAVIGACE MOBILNÍCH ROBOTŮ

Otázky spojené s navigací mobilních robotů jsou řešeny již od poloviny 60. let. Od té doby se objevilo veliké množství přístupů, které tuto velice důležitou součást robotiky posouvají stále kupředu. Strategie využívané v této problematice lze rozdělit do dvou základních kategorií, a to plánování globální cesty a reaktivní navigaci.

3.1 Metody plánování globální cesty

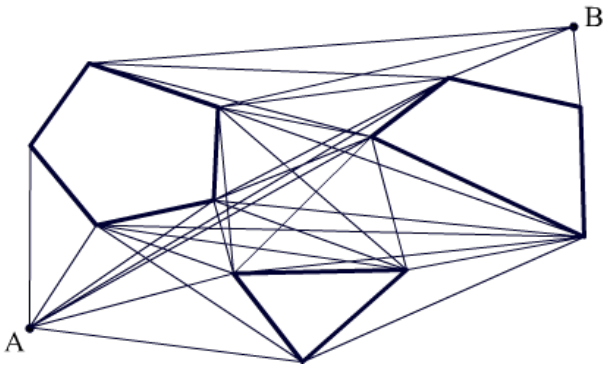
V tomto případě je snaha nalézt bezkolizní a optimální cestu robotu z počáteční pozice do cílové. Je však nutné mít k dispozici znalosti o prostředí. Zejména pak o překážkách, jejich rozměrech, orientaci či pohybu v prostředí. Současné klasické metody jsou variantou několika základních přístupů, mezi které patří: mapy cest, rozklad do buněk a potenciálová pole. V následujícím textu budou popsány některé z uvedených přístupů.

3.1.1 Mapy cest (Roadmaps)

V mapách cest je volný konfigurační prostor, to jest soubor proveditelných pohybů, redukován do síťového grafu. Hledání cesty robotu se pak mění na úlohu prohledávání grafu. Rozšíření zástupci map cest jsou např: graf viditelnosti, Voronoiuv diagram, siluety či síť podcílů [18].

Graf viditelnosti (Visibility Graph)

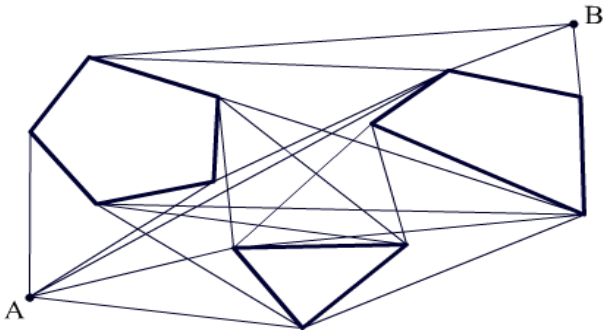
Vrcholy grafu jsou v tomto případě všechny vrcholy překážek, počáteční a cílová pozice. Hrany grafu jsou tvořeny spojeními mezi jednotlivými vrcholy, přičemž spoj nesmí protínat překážku. V případě výskytu nepolygonální překážky je nejprve nutné tuto překážku převést na polygonální. Množství hran může v komplikovaných prostředích značně narůst což vede k přílišným časovým nárokům [18].



Obr. 2 Graf viditelnosti.

Graf tečen

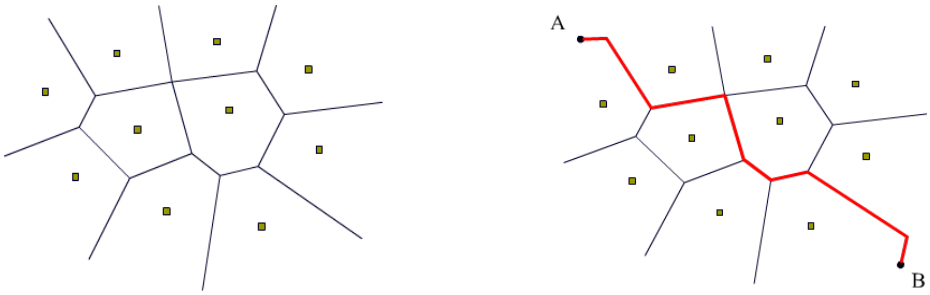
Velké množství hran v grafu viditelnosti se snaží eliminovat graf tečen, ve kterém jsou vytvořeny pouze ty hrany, jenž jsou tečnami jednotlivých vrcholů. Tím se značně urychluje prohledávání grafu [18].



Obr. 3 Graf tečen.

Voronoiuv diagram (Voronoi diagram)

Voronoiuv diagram je geometrická rovinná struktura reprezentující informace o sousedství objektů v dané rovině. Rovina je rozdělena tak, že každý bod je přiřazen nejbližšímu vrcholu, přičemž body, u kterých nelze přesně určit, ke kterému objektu náleží, tvoří Voronoiuv diagram. Ten je tedy tvořen body stejně vzdálenými od dvou či více objektů. Ty body, které jsou stejně vzdálené od třech a více objektů jsou považovány za uzly grafu a body stejně vzdálené od dvou objektů realizují hrany grafu. Dále je do grafu třeba ještě zakomponovat uzly reprezentující počáteční a cílovou pozici [18].



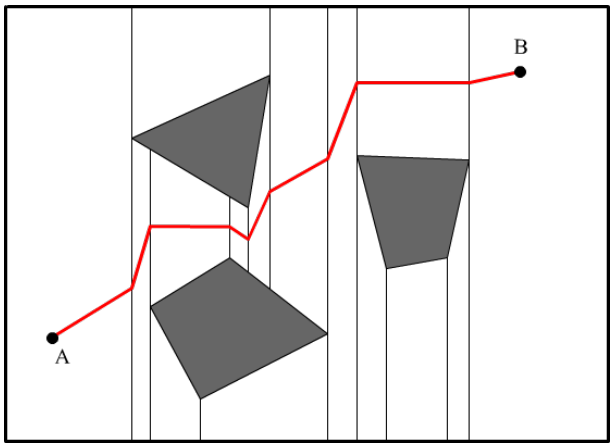
Obr. 4 Voronoiuv diagram

3.1.2 Rozklad do buněk

Metody rozkladu do buněk rozdělují prostředí robotu do buněk. Následně je třeba určit a označit jednotlivé buňky, zda obsahují překážku či nikoliv. Dále je sestaven graf, ve kterém vrcholy reprezentují buňky a hrany jsou zavedeny mezi sousedními buňkami. Tím je již problém hledání cesty robotu převeden na klasické prohledávání grafu, kde můžeme využít například algoritmus prohledávání do šířky či A*. Rozklad buněk lze rozdělit podle použité metody na dva základní přístupy.

Exaktní rozklad do buněk

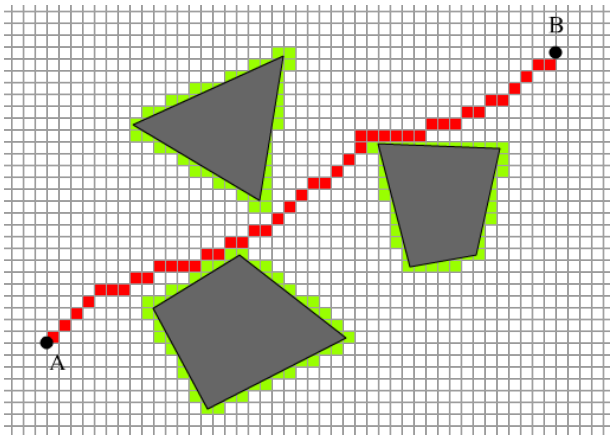
Principem exaktního rozkladu do buněk je rozdělení volného prostoru na nepřekrývající se buňky. Tvar buněk se volí jednoduchý, aby bylo možné snadno spočítat jejich hranice a přechody mezi sousedními buňkami. Nejčastěji se tedy volí lichoběžníková dekompozice. Poté je nalezena pomocí vhodné metody prohledávání grafu posloupnost buněk. Robot se tedy může bezpečně přemístit z počáteční polohy přes přechody jednotlivých buněk v nalezené posloupnosti až do cílové polohy. Pokud hledaná cesta existuje, tak je touto metodou vždy nalezena [24].



Obr. 5 Exaktní rozklad do buněk.

Aproximativní rozklad do buněk

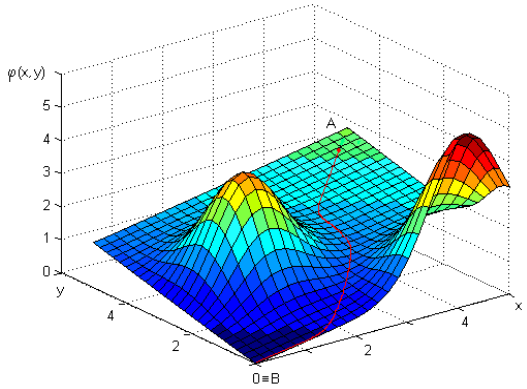
U tohoto způsobu rozkladu je prostředí robotu rozděleno na buňky stejného tvaru. Pokud překážka alespoň částečně zasahuje do buňky, ta je následně označena za obsazenou. Náročnost teto metody tedy závisí především na použitém rozlišení. Při použití aproximativního přístupu není zajištěno nalezení cesty, ovšem s rostoucím rozlišením se zvyšuje šance na její nalezení. Z důvodu urychlení prohledávání je možné nejprve volit rozlišení menší a následně při neúspěšném hledání cesty rozlišení postupně zvětšovat. Pokud místo dvouhodnotové logiky budeme reprezentovat obsazenost dané buňky pomocí vícehodnotové logiky, je pak např. možné reprezentovat buňkou jistotu, s jakou se v dané buňce překážka vyskytuje či nevyskytuje. Tím nám poté vznikají pravděpodobnostní mřížky [23].



Obr. 6 Aproximativní rozklad do buněk.

3.1.3 Potenciálová pole (Potential Fields)

Potenciálová pole vycházejí z teorie elektrického pole. Elektrické náboje se mohou pohybovat pouze mezi místy s různými potenciály. Pokud tedy vložíme kladný náboj do elektrického pole tvořeného také kladným nábojem, začne na tento náboj působit síla, která bude mít snahu přemístit tento náboj do oblasti s menším potenciálem. Prostředí robotu je tedy reprezentováno umělým elektrickým polem popsaným potenciálovou funkcí $E(x,y)$, kde překážky jsou kladně nabitá tělesa a pozice, která je pro robot označena jako cílová, je místo nejmenšího potenciálu [19].



Obr. 7 Funkce potenciálu pole vytvořená v prostředí Matlab [18].

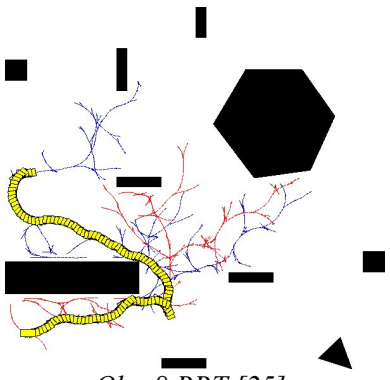
Výše uvedené klasické přístupy mohou trpět některými nedostatky, mezi než patří například vysoká časová náročnost při řešení složitých problémů nebo možnost uvíznutí v lokálním minimu. Z těchto důvodů bývají v praxi často neúčinné a bývají proto nahrazovány či doplňovány pravděpodobnostními přístupy.

3.1.4 Pravděpodobnostní mapy cest (Probabilistic Roadmaps)

Tento algoritmus pracuje ve dvou krocích. V prvním kroku, který je nazýván učící fáze, se generuje mapa cest. To se provádí náhodným vybíráním konfigurací robotu, které se následně ověřují, zda leží ve volném konfiguračním prostoru. Pokud tam náleží, jsou vytvořeny nové vrcholy grafu. Poté se algoritmus pokouší propojit nově vytvořené konfigurace s ostatními pomocí přímých cest. Pokud je takováto cesta celá obsažena ve volném konfiguračním prostoru, je přidána nová hrana. Tento cyklus se opakuje až do doby, kdy je vyčerpán čas či paměť. V druhé fázi, dotazovací, je nejprve připojena počáteční a cílová konfigurace a následně probíhá hledání cesty. Pokud cesta není nalezena, může to být způsobeno nedostatečným popsáním konfiguračního prostoru a je opakována první fáze. Tento algoritmus je vhodný především pro statická prostředí, kde je možné provádět opakovaně druhou fázi bez nutnosti opakování fáze první [19].

3.1.5 Pravděpodobnostní stromy (Rapidly-exploring random trees)

Základní myšlenkou tohoto algoritmu, který byl poprvé představen v roce 1998, je inkrementální vytváření prohledávacího stromu, jenž rychle a rovnoměrně prohledává konfigurační prostor. Strom je vytvářen tak, že se v každé iteraci rozroste směrem k náhodně zvolené konfiguraci robotu o nový vrchol. Vrcholy tedy představují konfigurace robotu a hrany představují akce, pomocí kterých se robot do dané konfigurace dostane. V praxi se základní verze algoritmu příliš nevyužívá, jelikož se generovaný strom rozrůstá všemi směry stejně, což vede k pomalé konvergenci k cíli. Úpravami vznikly např. následující plánovače: plánovač se směrem k cíli, plánovač se zaměřením na cíl nebo dvoustromový RRT plánovač [19].



Obr. 8 RRT [25].

K odstranění možnosti uvíznutí v lokálním minimu je využíváno při plánování cesty robotu mnoho heuristických a meta-heuristických algoritmů. Například kombinace simulovaného žíhání a potencialových polích eliminuje zmiňovaný problém. Heuristické algoritmy negarantují nalezení cesty, ale pokud již nějakou cestu naleznou, tak mnohem rychleji než deterministické metody. Dále budou popsány některé významné přístupy k řešení plánování cesty robotu.

Genetické algoritmy (Genetic Algorhitms)

Genetické algoritmy vycházejí z podstaty evolučního vývoje, při kterém se prosazují jedinci s žádoucími vlastnostmi, jejichž chromozóny jsou dále šířeny prostřednictvým potomků. Pro manipulaci s chromozomy bylo navrženo několik genetických operátorů, mezi které patří především selekce, mutace či křížení. Důležitým hlediskem, které může být zohledňováno při výběru rodičovských chromozomů, je hodnota „fitness“. Při řešení úloh za použití genetických algoritmů reprezentuje chromozom určité řešení a jeho fitness je kladná hodnota odpovídající hodnotě účelové funkce. Algoritmy pracují tedy tak, že je nejprve, náhodným generováním či za použití jiných heuristických metod, vytvořena počáteční populace určitého počtu chromozomů, která je následně měněna pomocí genetických operátorů do doby, než je splněna podmínka ukončení, což může např. být vyčerpání času či zjištění, že za určitou dobu nedošlo ke zlepšení účelové funkce. Dříve se pro reprezentaci chromozomů využívala především binární reprezentace, v dnešní době se však ukazuje jako výhodnější použití nebinární. Pro plánování dráhy robotu bylo navrženo velké množství metod, jak pro diskrétní, tak pro spojité prostředí. Metody využívají kromě klasických operátorů také operátory specifické, jako je např. operátor vyhlazení cesty [20].

Mravenčí algoritmy (Ant Colony Optimization)

Mravenčí algoritmy jsou inspirovány chováním reálných mravenců například při vyhledávání potravy, stavbě mraveniště, rozdělování prací či přepravě. Mravenci mají, i přesto, že jsou téměř slepí, schopnost nalézt nejkratší cestu mezi mraveništěm a potravou. Této schopnosti dosahují díky nepřímé komunikaci mezi členy kolonií tak, že mění feromonové stopy. Pro hledání cesty grafem se využívají tzv. umělí mravenci, kteří mají stejně jako jejich biologické předlohy pravděpodobnostní chování. To znamená, že čím vyšší hodnotu feromonu detekují na cestě, tím větší je pravděpodobnost, že se touto cestou vydají. Pokud tedy mravenci procházejí častěji určitou cestou, její feromonová vrstva narůstá, naopak pokud mravenci tuto cestu využívají méně, feromon se vypařuje a jeho hodnota klesá. Pro použití mravenčích algoritmů při plánování dráhy robotu je třeba popsat pracovní prostor pomocí grafu a určit startovní a cílový bod [21].

3.2 Reaktivní navigace

V reálném světě se většinou vyskytují problémy, k jejichž řešení nejsou k dispozici detailní informace o daném prostředí. Často se také dané prostředí mění v čase. Je ovšem nutné, aby i takové situace dokázal autonomní robot řešit a právě za tímto účelem jsou využívány reaktivní navigace, které se také velice uplatňují při vytváření mapy prostředí, jež může dále být využita při optimalizaci trajektorie. V reaktivní navigaci jde vlastně o mapování údajů získaných ze senzorů. Je možné tak získat například údaje o vzdálenosti překážky z infračervených, laserových či ultrazvukových senzorů. Reaktivní navigaci je možné rozdělit na navigaci založenou na modelu a navigaci založenou na prostředí [25].

Mezi metody reaktivního řízení, které jsou založeny na modelech, lze zařadit například metody potenciálových polí, metody detekce hran, detekce hranic překážek atd. Základem přístupu je model prostředí, na jehož základě jsou voleny řídicím systémem odpovídající akce.

Metody založené na prostředí mapují senzorické údaje přímo, což se v mnoha ohledech podobá reflexnímu chování biologických systémů.V těchto přístupech jsou často využívány přístupy založené na fuzzy logice, neuronových sítích či jejich kombinace.

V problematice reaktivní navigace je využíváno mnoho přístupů, mezi které patří především fuzzy logika, neuronové sítě, případové usuzování, umělá potenciálová pole atd.

Umělé neuronové sítě (artifical neural network)

Umělé neuronové sítě jsou založeny na své biologické podstatě a v dnešní době jsou využívány v mnoha aplikacích. Je tak možné se s nimi setkat například při rozpoznávání vzorů, optimalizaci, kategorizaci, predikci, asociaci či řízení. Základním prvkem jsou neurony, jež jsou vzájemně propojeny. Každý neuron může mít libovolný počet vstupů, ale pouze jeden výstup. Důležitým prvkem jsou přenosové funkce neuronu a váhy, které určují význam jednotlivých vstupů. Jednou z hlavních vlastností neuronových sítí je schopnost učit se. To může probíhat jak s učitelem, tak bez učitele. Neuronové sítě existují v různých variantách, mezi které patří například perceptron či rekurentní neuronová síť [27].

V problematice navigace mobilních robotů bývají neuronové sítě často využívány ve spojení s fuzzy logikou či genetickými algoritmy. V práci [14] je popsán přístup k reaktivní navigaci, jenž využívá třívrstvou dopřednou neuronovou síť ve spojení právě s genetickými algoritmy. První vrstva neuronové sítě je tvořena dvěma neurony, jež představují vstupní údaje, a to vzdálenost a úhel. Skrytá, tedy druhá vrstva, se skládá z dvaceti neuronů. Výstup ve třetí vrstvě tvoří dva neurony udávající zrychlení a natočení robotu. V [15] je prezentován přístup, jenž využívá při reaktivní navigaci robotu rekurentní neuronovou síť s radiální bází, na kterou je aplikována metoda posilovaného učení Q-learning.

Práce [16] popisuje návrh reaktivního řízení, jenž je založen na hierarchické vícevrstvé architektuře. Proces řízení se skládá z předzpracování a klasifikace vstupních dat, výběru odpovídající reakce (asociativní část) a jejího ověření. V klasifikační části je využívána Fuzzy-ART neuronová síť. ART (Adaptive Resonance Theory) je vícevrstvá neuronová síť bez učitele založená na teorii adaptivní rezonance, což znamená, že po přivedení vzoru na vstup dochází jak k dopřednému, tak zpětnému šíření signálu, jenž probíhá, dokud síť nerezonuje, tzn. dokud se neustálí na shodě, což je způsobeno nalezením shodného či podobného vzoru. Asociativní část je tvořena jednovrstvou neuronovou asociativní pamětí. Počet neuronů, jež jsou spojeny s výstupními neurony Fuzzy-ART neuronové sítě, je dán počtem řídicích pravidel.[28]

Fuzzy logika (fuzzy logic)

Fuzzy logika je díky své schopnosti pracovat s nepřesnými vstupními informacemi často využívána při reaktivní navigaci mobilních robotů. Práce [14] nabízí, kromě již zmíněného přístupu založeného na kombinaci neuronové sítě a genetických algoritmů, také přístup založený na kombinaci fuzzy logiky a genetických algoritmů. Slovními proměnnými jsou vzdálenost robotu od nejbližší překážky a úhel mezi osou robotu a touto překážkou. Výstupy jsou natočení a zrychlení robotu. Šířku fuzzy množin těchto proměnných je možné modifikovat pomocí koeficientů. Binární řetězec genetického algoritmu má délku 440 bitů a reprezentuje znalostní bázi FLC regulátoru. V [10] je prezentován přístup využívající 18 fuzzy pravidel k řízení robotu směrem k cíli, aniž by přišel do konfliktu s překážkami. Návrhem vhodného algoritmu, jenž využívá virtuální cíle je řešen problém uvíznutí v lokálním minimu. V práci [9] je popsán přístup, kombinující chování robotu v určitých situacích. Vstupní informace jsou zpracovávány pomocí fuzzy logiky. Stejně tak výsledný směr pohybu a rychlost robotu jsou odvozovány na základě fuzzy pravidel. Problém uvíznutí v lokálním minimu je řešen pomocí zaznamenávání údajů do matice, jež jsou následně pomocí fuzzy logiky zpracovávány. V [17] je využit fuzzy inferenční systém inspirovaný lidským chováním, jenž je založen na vyhledávání volného prostoru, kde se nevyskytují překážky, při dosahování cílové pozice. V případě některých konkávních překážek tento způsob selhává a je spuštěna druhá strategie, která je založena na sledování stěny s využitím podcílů. V práci [30] je popsána metoda, ve které jsou slepé uličky detekovány pomocí porovnávání změny úhlu natočení robotu. Robot se v případě detekce slepé uličky snaží uniknout pomocí sledování stěny. Směrování k cíli je spuštěno opět v okamžiku, kdy se robot nachází na přímce spojující cílovou pozici a bod, ve kterém bylo únikové chování spuštěno. Přístup prezentovaný v [31] je také založen na sledování stěny. To je ovšem spuštěno ve chvíli, kdy robot rozpozná pozici, ve které se již vyskytoval v předchozím průběhu navigace. Sledování stěny je ukončeno, když robot opustí danou obdélníkovou oblast.

Případové usuzování (case-based reasoning)

Případové usuzování je alternativou k pravidlově řízeným či objektově orientovaným systémům. Nepracuje na principu vytváření obecných vztahů mezi problémem a jeho řešením, ale snaží se využít zkušenosti získané z řešení již vyřešených situací, což je způsob řešení problémů často využívaný v reálném životě.

Práce [13] se zabývá využitím případového usuzování v problematice reaktivní navigace mobilního robotu ve spojitém prostředí za využití spojitých informací ze senzorů a kontinuálního revidování navržené cesty. V uvedené práci je využito případové usuzování kombinované s posilovaným učením. Systém využívá kromě aktuální situace i nedávnou historii situací, jež vedly k té stávající. Řídicí systém robotu neustále hledá situace v historii, jež by byly podobné té stávající. Pokud nalezené situace byly úspěšně vyřešeny, jsou následně s patřičnými modifikacemi v závislosti na aktuální situaci využity. Není-li nalezen vhodný případ, použije robot patřičné parametry a výsledek této situace je uložen jako nový případ. Spolu s případem jsou ukládány i znalosti potřebné k adaptaci chování.

Umělé imunitní systémy (atrifical immune systems)

Tento přístup je inspirován principy a procesy biologického imunitního systému, jenž zajišťuje obranyschopnost živočichů proti virům a bakteriím (antigeny). Své uplatnění nachází například v úlohách rozpoznávání vzorů, optimalizaci, datové analýze či detekci anomálií. Imunitní systém pracuje ve třech vrstvách, přičemž první z vrstev je vrstva fyzická, která tvoří bariéru. Pokud patogen, pronikne touto bariérou, pokouší se ho zastavit vrozená imunita. Pokud neuspěje, přichází na řadu adaptivní imunita. Její buňky mohou být specifické pro různé antigeny a mají větší paměťovou kapacitu než buňky vrozené imunity. Mezi významné rysy imunitního systému patří možnost samoorganizace, rozpoznávání, adaptace či schopnost uchovávat informace a učit se. Imunitní systém se skládá z agentů (imunitních buněk), které mají adaptační a učící schopnosti podobné umělým neuronovým sítím s tím rozdílem, že imunitní systém je založen na dynamické kooperaci agentů. Vzhledem ke svým vlastnostem je tento systém stále častěji využíván i v oblasti robotiky.[22]

V práci [12] je prezentován přístup využívající reaktivní imunitní síť. Ta je založena právě na imunitním systému a s její pomocí je realizována reaktivní navigace robotu. Vstupní informace získané ze senzorů odpovídají antigenům a protilátky, neboli antibody, představují směry, jimiž se robot muže pohybovat. Je pak následně vybírána protilátka s největší koncentrací, díky níž je robot naváděn směrem k cílové pozici, aniž by přišel do konfliktu s překážkami.

4 FUZZY

Za zakladatele fuzzy logiky je považován L. A. Zadeh, který publikováním článku „Fuzzy sets“ roku 1965 položil základy této matematické disciplíny. Značné obliby se jí však dostalo až na přelomu osmdesátých a devadesátých let a to především v Japonsku. V dnešní době je již tento přístup značně rozšířen a je hojně využíván v nejrůznějších odvětvích. Zejména v oblasti řízení a regulace našla fuzzy logika své uplatnění, kde je s úspěchem implementována například při řízení soustav rychlovýtahů v mrakodrapu či automatickém řízení podzemní dráhy. Je možné se s ní setkat jak v běžných domácích spotřebičích, tak i při řešení komplikovaných problémů, jakým byla například oprava Hubbleova teleskopu pomocí automatického ramena, řízeného právě za pomoci fuzzy logiky. Dalšími oblastmi uplatňujícími fuzzy logiku je rozpoznávání obrazců, kde je využita k rozpoznávání rukou psaného písma, databázové systémy, či nejrůznější ekonomické aplikace. Velká obliba tohoto matematického přístupu je založena především na možnosti pracovat s informacemi, které jsou více či méně nepřesné a ve většině případů specifikované pouze pomocí přirozeného jazyka. Tato kapitola je zpracována na základě [1], [2], [3], [4], [6] a [18].

4.1 Fuzzy množiny

Pojem fuzzy množiny je ústředním pojmem fuzzy logiky a jedná se o jisté zobecnění klasického pojmu množiny. V teorii klasických množin prvek x do množiny A buď patří, nebo nepatří, kdežto u fuzzy množin je prvku x přiřazena určitá pravdivostní hodnota nazývaná stupeň příslušnosti, vyjadřující míru našeho přesvědčení, nakolik daný prvek x do množiny patří.

Při definici fuzzy množiny je nejprve zapotřebí definovat množinu X nazývanou univerzum, což je množina prvků libovolného druhu, nejčastěji však čísel. Poté fuzzy množinou A univerza X rozumíme objekt popsaný charakteristickou funkcí, která se také může nazývat funkce příslušnosti.

$$\mu_A : X \rightarrow [0,1]$$

Pro každý prvek $x \in X$ nám tedy hodnota $\mu_A(x) \in [0,1]$ značí, do jaké míry je x prvkem fuzzy množiny A. Pokud máme stupeň příslušnosti $\mu_A(x) = 0$, můžeme s jistotou říci, že daný prvek do množiny A nepatří. Naproti tomu u stupně příslušnosti $\mu_A(x) = 1$ prvek zcela jistě množině A náleží. Pokud však $\mu_A(x) \in (0,1)$, nemůžeme s jistotou říci zdali prvek x množině A náleží či nenáleží. Odhad stupňů příslušnosti je subjektivní, nicméně z experimentálních výsledků je zřejmé, že různí lidé odhadují stupně podobně. Což mimo jiné dokazuje i schopnost přenášet informace prostřednictvím přirozeného jazyka.

Explicitně je fuzzy množinu možné popsat takto:

$$A = \{ \frac{a_1}{x_1} , ... , \frac{a_n}{x_n} \}, \text{ kde}$$

$x_1,\ldots, x_n \in X$ jsou prvky, kterým jsou přiřazeny stupně příslušnosti $a_1, ..., a_n \in (0,1]$. Prvky se stupněm příslušnosti 0 nejsou tedy zahrnuty.

V teorii fuzzy množin hrají důležitou roli následující klasické množiny:

a) Nosič

$$Supp(A) = \{x | A(x) > 0\},$$

Nosičem fuzzy množiny A je množina všech prvků univerza, jejichž stupeň příslušnosti do A je nenulový.

b) α – řez

$$cut_\alpha(A) = \{x | A(x) \geq \alpha\},$$

α – řez je množina prvků majících stupeň příslušnosti větší nebo roven zadanému stupni α. Tuto množinu tedy získáme z fuzzy množiny A ořezáním všech prvků se stupněm příslušnosti menším než α.

c) Jádro

$$Ker\ (A) = \{x | A(x) = 1\},$$

jádro je množina prvků, o kterých můžeme s jistotou říci, že patří do fuzzy množiny A.

Fuzzy množinu můžeme nazývat normální, jestliže $Ker(A) \neq \emptyset$. Pokud stupně příslušnosti všech prvků dané množiny jsou menší než 1, nazývá se taková množina subnormální. Důležitou roli hraje i fuzzy jednoprvková množina, nazývaná také singleton představující fuzzy analogii klasické jednoprvkové množiny:

$$A = \{ \ \frac{a_1}{x_1} \ \},$$

Fuzzy množina je konvexní, jestliže je definována na univerzu reálných čísel R a pokud $cut_\alpha(A)$ je pro všechna $\alpha \in [0,1]$ spojitý interval.

U fuzzy množin lze, stejně tak jako u klasických množin, definovat základní operace, mezi které patří především:

Sjednocení – sjednocením fuzzy množin A a B v univerzu X vznikne množina $C = (X, \mu_C)$, pro kterou platí:

$$\mu_C(x) = \max(\mu_A(x), \mu_B(x)) \text{ pro každé } x \in X.$$

Průnik – průnikem fuzzy množin A a B v univerzu X vznikne množina $C = (X, \mu_C)$, pro kterou plati:

$$\mu_C(x) = \min(\mu_A(x), \mu_B(x)) \text{ pro každé } x \in X.$$

Doplněk (komplement) – doplňkem fuzzy množiny A je množina $\bar{A}$, pro kterou platí:

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \text{ pro každé } x \in X.$$

Dvě fuzzy množiny A a B v univerzu X jsou totožné v případě, že platí $\mu_A(x) = \mu_B(x)$ pro všechna $x \in X$.

Fuzzy množina A je podmnožinou fuzzy množiny B v univerzu X, platí-li, že $\mu_A(x) < \mu_B(x)$ pro všechna $x \in X$.

Kartézský součin fuzzy množin $A = (X, \mu_A)$ a $B = (Y, \mu_B)$ je definován jako fuzzy množina $A \times B = (X \times Y, \mu_{A \times B})$, $\mu_{A \times B}(x,y) = \min \{\mu_A(x), \mu_B(y)\}$

Fuzzy relace je fuzzy množina $R = (U_1 \times U_2 \times ... \times U_n, \mu_R)$, kde $U_1, U_2, ..., U_n$ jsou klasické množiny a μ_R: $U_1 \times U_2 \times .... \times U_n \rightarrow [0,1]$.

4.2 Vícehodnotová logika

Vícehodnotová logika je logika s více než dvěma pravdivostními hodnotami. Tyto hodnoty jsou zpravidla v intervalu [0, 1]. Množinu $C = [0, 1]$ budeme považovat za množinu logických hodnot, přičemž 0 odpovídá pravdě a 1 nepravdě. Logická konstanta je libovolné reálné číslo z intervalu [0, 1]. Logická proměnná je proměnná, která nabývá hodnot z množiny C [29].

Množinu $L = \{\vee , \wedge , \& , \Rightarrow\}$ nazveme množinou logických spojek. Její prvky jsou logické spojky s názvy: disjunkce, konjunkce, odvážná konjunkce a implikace.

Nechť W je konečná množina logických proměnných. Následujícími pravidly definujeme řetězec jenž nazveme formulí.

1. Je-li $\alpha \in C$, pak α je formule.
2. Je-li $\beta \in W$, pak β je formule.
3. Jestliže φ a ψ jsou formule a $* \in L$, pak $(\varphi * \psi)$ je formule.

Nechť Q je množina formulí a $\Omega(Q)$ množina jejich interpretací. Interpretace formule je dosazením logických konstant za logické proměnné.

Pravdivostním ohodnocením nazveme zobrazení $V{:}\Omega(Q) \rightarrow C$, které splňuje tyto požadavky:

1. $V(\alpha) = \alpha$ pro každé $\alpha \in$ C.
2. $V(\varphi \vee \psi) = \max(V(\varphi), V(\psi))$ pro všechna $\varphi, \psi \in \Omega(Q)$.
3. $V(\varphi \wedge \psi) = \min(\mathrm{V}(\varphi), \mathrm{V}(\psi))$ pro všechna $\varphi, \psi \in \Omega(Q)$.
4. $V(\varphi \,\&\, \psi) = \max(0, V(\varphi) + V(\psi) - 1)$ pro všechna $\varphi, \psi \in \Omega(Q)$.
5. $V(\varphi \Rightarrow \psi) = \min(1,1 - V(\varphi) + V(\psi))$ pro všechna $\varphi, \psi \in \Omega(Q)$.

Operace negace je definována jako:

$$\neg\varphi = \varphi \Rightarrow 0$$

Pro pravdivostní ohodnocení negace dostáváme vztah:

$$V(\neg\varphi) = V(\varphi \Rightarrow 0) = \min(1, 1 - V(\varphi)) = 1 - V(\varphi)$$

4.3 Lingvistické proměnné

Lingvistické čili slovní proměnné jsou stěžejním prvkem fuzzy logiky a to především díky své schopnosti reprezentovat hodnoty pomocí běžného jazyka. Tím umožňují efektivně popsat problémy, kde nelze vytvořit přesné matematické modely, nebo by vytvoření takových modelů bylo časově či finančně náročné. Podle L. A. Zadeha je možné slovní proměnnou definovat jako uspořádanou pětici:

$$x = (X, T, U, G, M), \text{ kde}$$

X ... je název proměnné *x,*
T ... množina slovních hodnot, kterých může proměnná *x* nabývat,
U... univerzum,
G... syntaktické pravidlo, pomocí kterého jsou tvořeny jazykové výrazy z množiny *T*,
M...sémantické pravidlo, pomocí kterého je každému jazykovému výrazu z množiny *T* přiřazen jeho význam, který je fuzzy množinou.

Pokud je množina slovních hodnot *T* zadána pomocí konečného výčtu hodnot, což znamená, že není zapotřebí gramatika *G*, a identifikátor každé hodnoty z *T* je totožný s názvem fuzzy množiny interpretující tuto hodnotu, nazýváme takovou lingvistickou proměnnou nestrukturovanou (normální). Takovou lingvistickou proměnnou můžeme reprezentovat pomocí následující trojice:

$$x = (X, T, U), \text{ kde}$$

T ... konečná množina slovních hodnot, kterých může proměnná *x* nabývat

4.4 Fuzzy implikační funkce

Každé fuzzy pravidlo je vyjádřeno pomocí spojky THEN. Spojka THEN je obecněji nazývána jako fuzzy implikační funkce. Implikaci je možné vyjádřit pomocí základních logických operací následovně [18]:

$$p \Rightarrow q \;=\; \neg p \vee q \;=\; (p \wedge q) \vee \neg p$$

Ve fuzzy logice máme k dispozici mnoho přístupů jak interpretovat implikační funkci. V následující části budou uvedeny některé z nich, za předpokladu, že A je fuzzy množina v univerzu U a B je fuzzy množina v univerzu V [2].

Kleene-Dienes $\mu_{A\Rightarrow B}(x, y) = \max \{1 - \mu_A(x), \mu_B(y)\}$

Lukasiewicz $\mu_{A\Rightarrow B}(x, y) = \min \{1, 1 - \mu_A(x), \mu_B(y)\}$

Zadeh (Willmott) $\mu_{A\Rightarrow B}(x, y) = \max \{ \min [\mu_A(x), \mu_B(y)], 1 - \mu_A(x)\}$

Mamdani $\mu_{A\Rightarrow B}(x, y) = \min \{\mu_A(x), \mu_B(y)\}$

Larsen $\mu_{A\Rightarrow B}(x, y) = \mu_A(x) \,.\, \mu_B(y)$

4.5 Fuzzy pravidla

Logické řízení je obecně založeno na vyhodnocování rozhodovacích pravidel ve formě:

IF <fuzzy výrok> THEN <fuzzy výrok>.

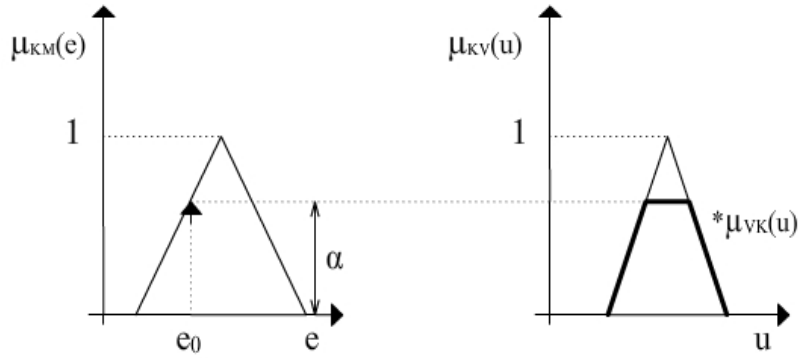
Tato podmínka je označována jako “produkční pravidlo”, kde první fuzzy výroková množina je předpoklad též nazývaný ancedent a druhý fuzzy výrok důsledek neboli konsekvent. Ancedent bývá často složeným fuzzy výrokem, jehož jednotlivé výroky jsou spojeny pomocí logických spojek.

Implikace jednorozměrné závislosti

Budeme uvažovat následující známou situaci, kdy v ancedentu rozhodovacího pravidla bude lingvistická proměnná popisující regulační odchylku *E*, jejíž hodnota bude *“KM”*, což značí kladná malá. Jako výstup budeme požadovat hodnotu akční veličiny, tudíž v konsekventu bude lingvistická proměnná reprezentující akční veličinu *U*, s hodnotou *“VK”*, tedy velká kladná. Průběhy funkcí příslušností jsou zřejmé z obr. 9.

IF *e.KM* THEN *u.VK*

Nejprve je nutné změřit hodnotu regulační odchylky e_0, kterou následně převedeme prostřednictvím funkce příslušnosti na odpovídající stupeň příslušnosti, s jakým daná hodnota náleží do fuzzy množiny “*KM*” slovní proměnné *E*. Poté je třeba nalézt fuzzy množinu konsekventu splňující podmínkovou část pravidla. Tento proces vychází, na základě Mandaniho implikace, z logického předpokladu, že důsledek může mít stupeň příslušnosti maximálně o hodnotě stupně příslušnosti konsekventu. Tak prostřednictvím převodu vstupní ostré hodnoty získáme hladinu α, pomocí které jsme schopni modifikovat fuzzy množinu v důsledkové části pravidla. Její funkce příslušnosti pak bude $*\mu_{VK}(u)$.



Obr. 9 Průběh implikace jednorozměrné závislosti.

Implikace dvourozměrné závislosti s jedním pravidlem

V tomto případě budeme uvažovat ancedent skládající se ze dvou dílčích výroků spojených logickou spojkou. Předchozí příklad tedy modifikujeme tak, že do ancedentu přidáme další slovní proměnnou Δe, která bude představovat regulační odchylku a její hodnota bude "kladná" (K). Pravidlo by tedy ve výsledku vypadalo v případě použití logické spojky AND nasledovně

IF *e.KM* AND Δ*e.K* THEN *u.VK*

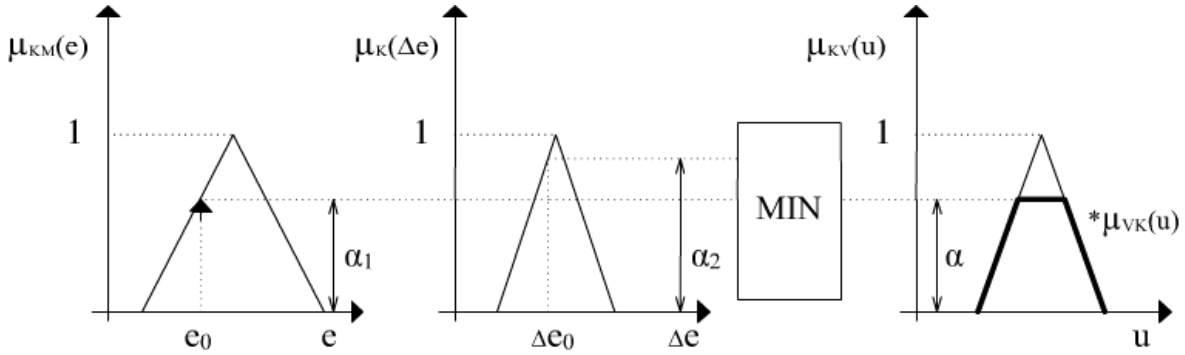
Stejně jako v předchozím příkladě je nejprve nutné převést ostré vstupní hodnoty na stupně příslušnosti jednotlivých fuzzy množin. Poté v závislosti na zvolené Mamdaniho implikaci je možné psát:

$$\mu_{VK}\ (e, \Delta e) = \min\ \{\mu_{KM}(e), \mu_K(\Delta e)\}.$$

Minimalizací je vyjádřena skutečnost, že konsekvent může mít stupeň příslušnosti maximálně takový jaký má ancedent.

Následně opět získáme hladinu α a s jejím využitím modifikujeme fuzzy množinu v konsekventu.

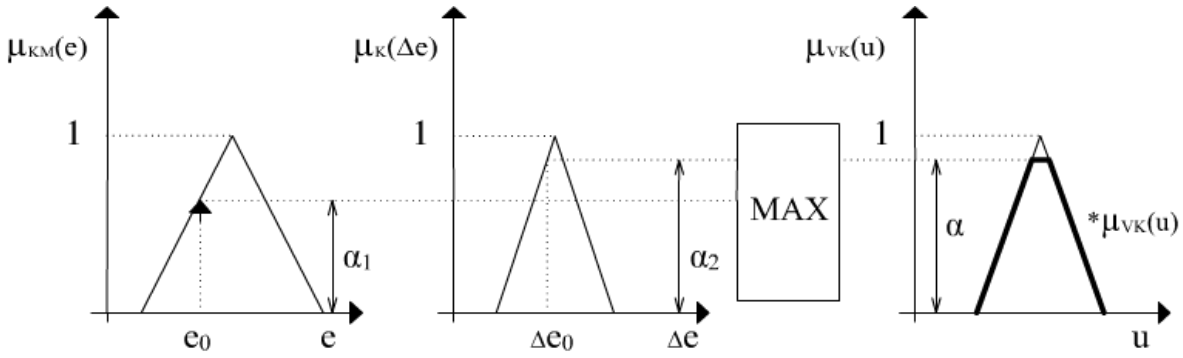
$$\alpha = \mu_{KM}(e)\ \mu_K(\Delta e) = \min\ \{\mu_{KM}(e_0), \mu_K(\Delta e_0)\}$$
$$^*\mu_{VK}(u) = \alpha\ \mu_{VK}(u) = \min\{\ \alpha, \mu_{VK}(u)\}$$



Obr. 10 Průběh implikace dvourozměrné závislosti s jedním pravidlem při použití spojky AND .

V případě použití logické spojky OR v ancedentu bychom hodnotu α hledali jako maximum z odpovídajících stupňů příslušností uvažovaných vstupních hodnot.

$$\alpha = \mu_{KM}(e) \ \mu_K(\Delta e) = \max \{\mu_{KM}(e_0), \mu_K(\Delta e_0)\}$$



Obr. 11 Průběh implikace dvourozměrné závislosti s jedním pravidlem při použití spojky OR.

Implikace dvourozměrné závislosti se dvěma pravidly
V tomto případě budou uvažována pravidla ve tvaru:

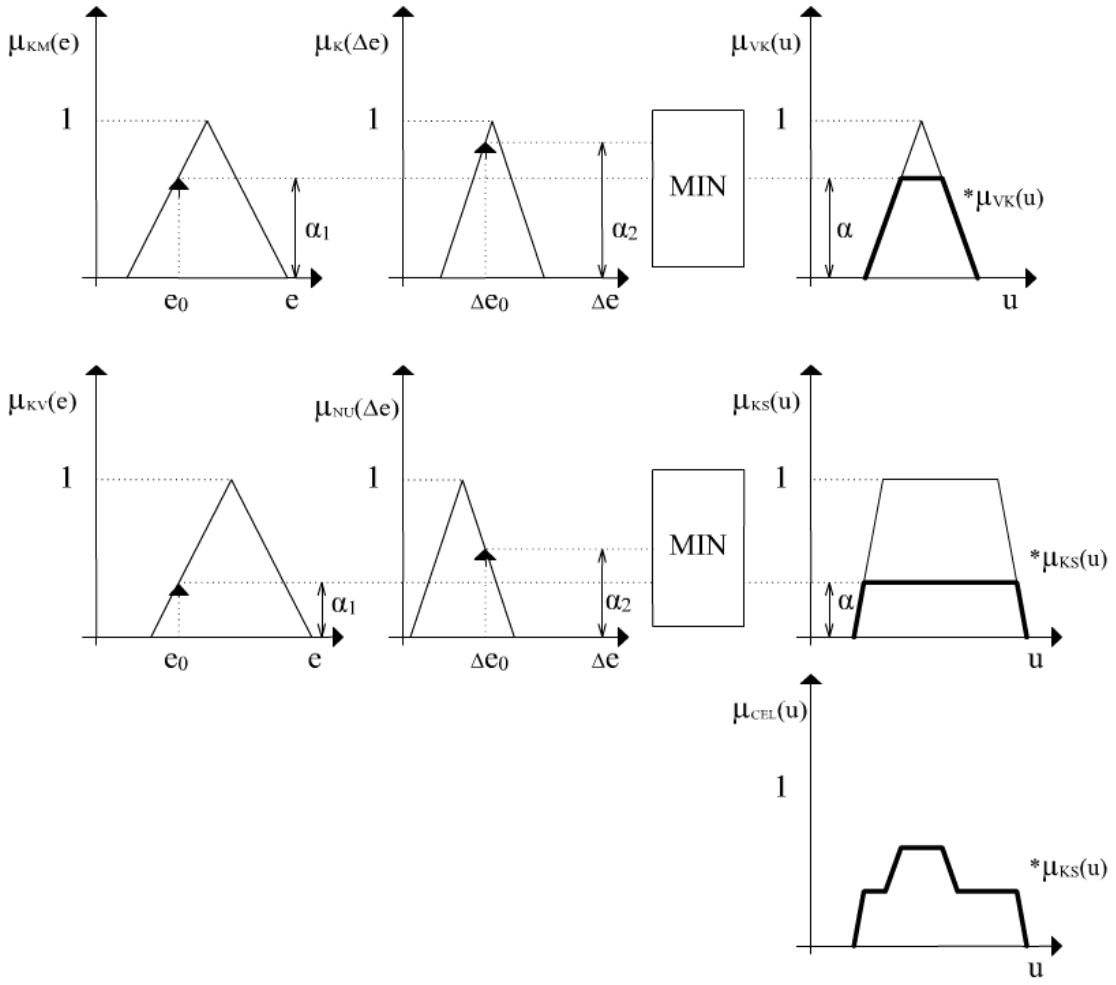
IF *e.KM* AND Δ*e.K* THEN *u.VK* ELSE
IF *e.KV* AND Δ*e.NU* THEN *u.KS*

V případě použití Mamdaniho implikace postupujeme jako v předchozím případě, určením funkce příslušnosti fuzzy množiny konsekventu pro každé pravidlo samostatně.

$$\alpha_1 = \mu_{KM}(e) \ \mu_K(\Delta e) = \min \{\mu_{KM}(e_0), \mu_K(\Delta e_0)\}$$
$$\alpha_2 = \mu_{KV}(e) \ \mu_{NU}(\Delta e) = \min \{\mu_{KV}(e_0), \mu_{NU}(\Delta e_0)\}$$
$$^*\mu_{VK}(u) = \alpha_1 \, \mu_{VK}(u) = \min\{ \alpha_1, \mu_{VK}(u)\}$$
$$^*\mu_{KS}(u) = \alpha_2 \, \mu_{VK}(u) = \min\{ \alpha_2, \mu_{VK}(u)\}$$

Výslednou funkci příslušnosti pak lze získat pomocí následující rovnice

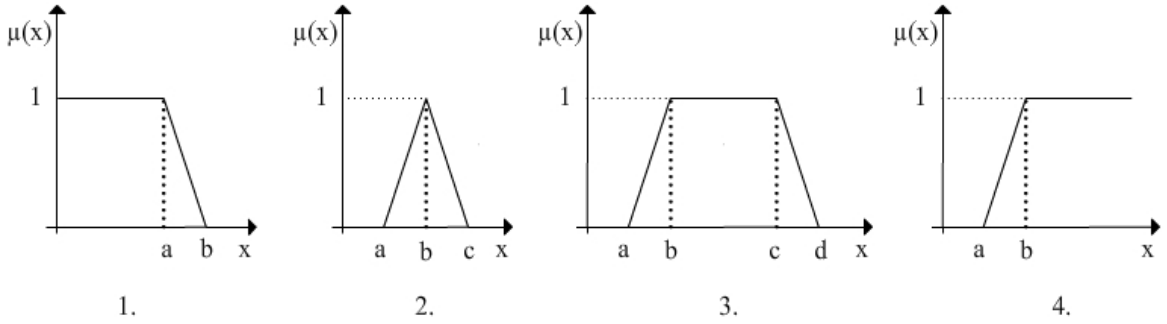
$$^*\mu_{CEL}(u) = \max\{ \min\{ \alpha_1, \mu_{VK}(u)\}, \min\{ \alpha_2, \mu_{KS}(u)\}\}.$$



Obr. 12 Průběh Mamdaniho implikace dvourozměrné závislosti se dvěma pravidly .

4.6 Fuzzifikace

Fuzzifikace je proces, při kterém se naměřené vstupní hodnoty převádějí na stupně příslušnosti jedné nebo více funkcí příslušnosti fuzzy množin. Základním předpokladem je kompletní pokrytí uvažovaného univerza nosiči příslušných fuzzy množin a to tak, aby v každém bodě univerza bylo možné transformovat ostrou vstupní hodnotu minimálně alespoň na jednu funkci příslušnosti s hodnotou stupně příslušnosti větším jak nula. Hovoříme-li o ε pokrytí univerza, znamená to, že se funkce příslušnosti protínají v úrovních $\geq \varepsilon$. Hodnota ε se volí alespoň 0.5 čímž je zajištěno, že bude vždy existovat alespoň jedna fuzzy množina, která bude mít pro danou hodnotu univerza stupeň příslušnosti minimálně s hodnotou ε. Z hlediska množství fuzzy množin pokrývajících dané univerzum, bylo na základě odborných studií zjištěno, že ideální počet by se měl pohybovat mezi 3-7 fuzzy množinami. Důležitou otázkou při návrhu je, jaké tvary funkcí příslušností by měly fuzzy množiny mít. Tvar funkce je možné, v závislosti na uvažovaném typu úlohy, zvolit předem nebo na základě měřených dat. Ve druhém případě je pak možné využít tzv. expertních odhadů, či algoritmů modifikovaných pro práci s fuzzy množinami, mezi které patří např. fuzzy ISODATA nebo fuzzy C-means. Pokud volíme tvar funkce příslušnosti předem, je výhodné volit tento tvar co nejjednodušší, pokud možno složený pouze z lineárních úseků. Dále jsou na obrázku prezentovány některé často využívané příklady funkcí příslušnosti.



Obr. 13 Často využívané průběhy funkcí příslušnosti.

4.7 Defuzzifikace

Ačkoliv fuzzy logika umožňuje pracovat a nepřesnými informacemi, je ve většině případů nakonec potřeba získat na výstupu přesnou hodnotu. Tento proces, při kterém je fuzzy množina transformována na konkrétní číslo, se nazývá defuzzifikace. K defuzzifikaci je možné využít různé metody vycházející z empirických ověření a heuristických přístupů. Nejvyužívanější budou dále popsány.

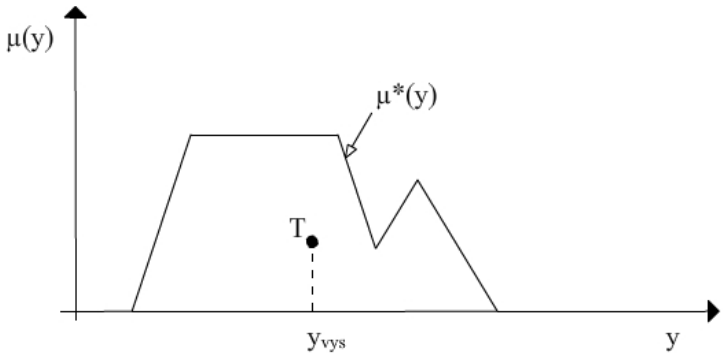
4.7.1 Metody těžiště (Center Of Gravity)

U tohoto přístupu získávání ostré hodnoty z výstupních fuzzy množin jsou k dispozici dvě metody.

Těžiště plochy (Center of Gravity)

Výsledná defuzzifikovaná ostrá hodnota y_{vys} je určena na základě souřadnice polohy těžiště plochy, jenž vznikne sjednocením dílčích ploch, jejichž hranice jsou určeny funkcemi příslušnosti jednotlivých termů. Je možné ji spočítat za pomocí následujícího vzorce, kde $\mu*$ odpovídá funkci příslušnosti sjednocené plochy.

$$y_{vys} = \frac{\int_{-\infty}^{\infty} y\,\mu*(y)\,dy}{\int_{-\infty}^{\infty} \mu*(y)\,dy}$$



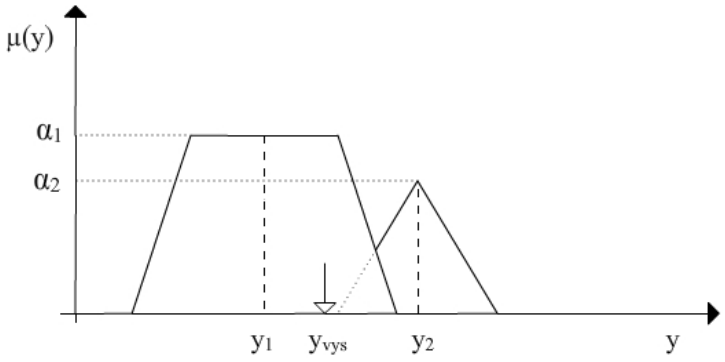
Obr. 14 Průběh defuzzifikace metodou těžiště plochy.

Tato metoda patří mezi nejpoužívanější, její nevýhodou je však větší výpočetní složitost.

Těžiště singletonů (Center of Maximum)

Při využití této metody je nejprve nutné nahradit funkční závislost jednotlivých termů jejich typickou hodnotou. Poté se ostrá výstupní veličina určí jako těžiště daných typických hodnot využitím níže uvedeného vzorce, kde α_i je stupeň příslušnosti i-tého termu a y_i je souřadnice výstupní veličiny i-tého termu.

$$yvys = \frac{\sum_{i=1}^{n} \alpha_i y_i}{\sum_{i=1}^{n} \alpha_i}$$

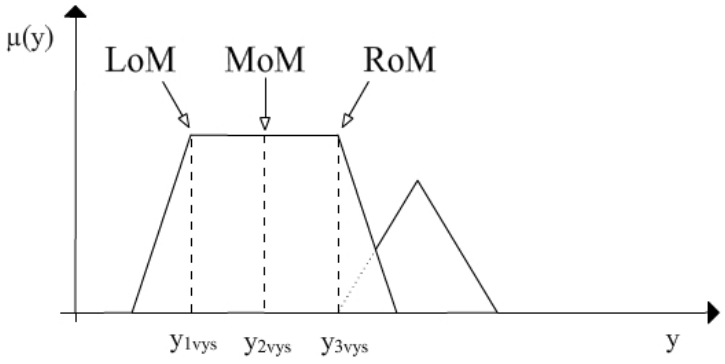


Obr. 15 Průběh defuzzifikace metodou těžiště singletonů.

4.7.2 Metody nejvýznamnějšího maxima (Mean of Maximum)

Tato metoda je výpočetně jednodušší než metoda těžiště. Principem je nalezení tzv. přijatelného řešení, které vyhovuje podmínkám daným v rozhodovacích pravidlech. Ze všech termů je vybrán ten s největší hodnotou funkce příslušnosti a poté je nalezena maximální hodnota této funkce příslušnosti, která svojí pozicí, odvíjející se od zvolené metody, určí hodnotu výstupní veličiny. Nevýhodou těchto metod je, že se výstupní veličina může měnit nespojitě. Je možné využít následující varianty metody:

- ***Left of Maximum (LoM)*** – výsledek je určen jako nejvíce vlevo položená hodnota maximální hodnoty funkce příslušnosti.
- ***Mean of maximum (MoM)*** – výsledkem je ve středu položená hodnota maximální hodnoty funkce příslušnosti.
- ***Right of Maximum (RoM)*** - výsledek je určen jako nejvíce vpravo položená hodnota maximální hodnoty funkce příslušnosti.



Obr. 16 Průběh defuzzifikace metodami nejvýznamnějšího maxima.

5 NÁVRH SYSTÉMU ŘÍZENÍ ROBOTU

Po předchozích kapitolách, jež měly za úkol seznámení s teoretickými základy problematiky navigace robotu a fuzzy logiky, následuje praktická část, ve které jsou představeny principy a výsledky navržených přístupů využívajících právě výše zmíněné teorie. Před konkrétním návrhem metod navigace robotu je třeba nejprve provést analýzu daného problému, definovat a zodpovědět některé významné otázky, mezi které patří především otázka vlastností robotu a prostředí nebo otázky spojené s fuzzy logikou.

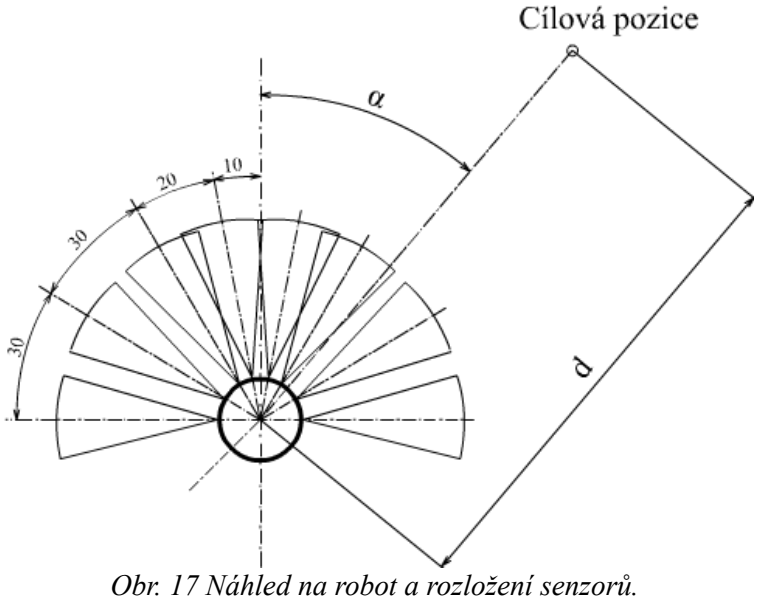
5.1 Analýza prostředí a vlastností robotu

Hlavním úkolem této práce je navrhnout systém řízení robotu v předem neznámém prostředí. Robot by měl být schopen přemístit se z počáteční polohy do cílové, aniž by nastala kolize. Dále by měl měnit svoji rychlost v závislosti na blížící se překážce či cílové pozici. Prostředí robotu není blíže specifikováno, ovšem předpokládá se výskyt jak konvexních tak konkávních statických překážek. V prostředí, kde se vyskytují pouze překážky s konvexním tvarem a kde tedy nemůže nastat situace, že robot uvízne například ve slepé uličce, je možné využívat pouze reaktivní řídicí systém. V případě výskytu složitějších překážek již tento systém selhává a je třeba zaznamenávat získané informace pro pozdější využití.

Základním prvkem při návrhu metody navigace robotu, jsou vlastnosti robotu. A to především jeho podvozek, který zásadně omezuje pohybové možnosti. V případě této práce bude robot vybaven diferenciálním podvozkem, jenž byl krátce popsán již ve druhé kapitole. Omezením robotu bude schopnost pohybovat se pouze dopředu, ve směru osy natočení robotu.

Robot disponuje v každém okamžiku znalostmi o aktuální vzdálenosti cíle a jeho relativnímu natočení vůči ose robotu. Důležitou součástí robotu je senzorický systém navržený dle [10]. Je tedy využito 8 blíže nespecifikovaných senzorů, které jsou uspořádány dle obr. 17 a umožňují detekovat překážku ve vzdálenosti až 100 cm. Získáváme tedy údaje *d0, d1, d2, d3, d4, d5, d6, d7*. Senzory robotu jsou seskupeny do tří oblastí (LEFT, FRONT, RIGHT) neboli levá, přední a pravá oblast. Hodnota vzdálenosti překážky v jednotlivých oblastech je pak získána dle níže uvedených rovnic.

$$d_{left} = \min(d0, d1),$$
$$d_{front} = \min(d2, d3, d4, d5),$$
$$d_{right} = \min(d6, d7).$$



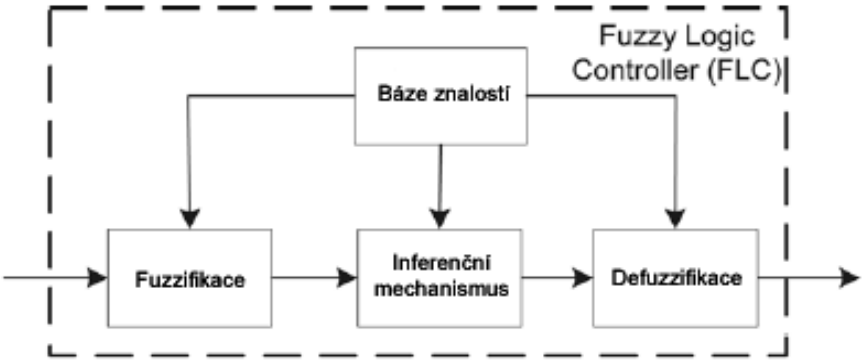
Obr. 17 Náhled na robot a rozložení senzorů.

Rozměry robotu nejsou pro návrh systému řízení až tak podstatné, avšak alespoň základní parametry robotu byly navrženy na základě robotu použitého v práci [11].

Průměr robotu: 0,5 m
Maximální translační rychlost: 0,5 m/s
Maximální rychlost natáčení: 45 °/s

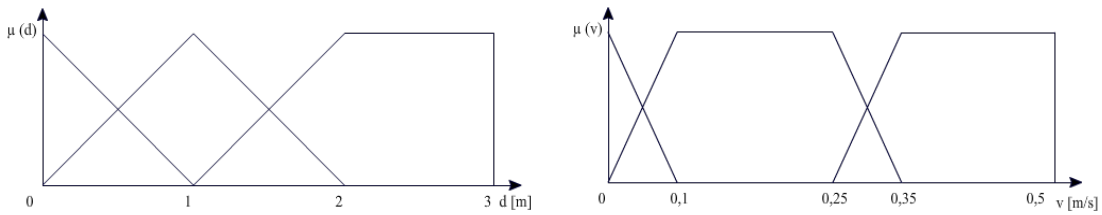
5.2 Analýza a výběr základních vlastností fuzzy regulátoru

Základní součástí systému řízení robotu je fuzzy regulátor, jehož nejvýznamnější části jsou zřejmé z obrázku. Fuzzy regulátor se skládá z bloku fuzzifikace, kde jsou "ostrá" vstupní data převáděna na slovní hodnoty. Dále pak z bloku báze znalostí, který slouží k uchovávání pravidel, jež jsou v části inferenčního mechanismu využívána k odvození hodnot slovních výstupních proměnných. Ty jsou pak následně opět převedeny na číselné hodnoty v bloku deffuzzifikace. Volbou vlastností inferenčního mechanismu a poté především výběrem deffuzzifikační metody jsou značně ovlivněny výsledné hodnoty.



Obr. 18 Fuzzy regulátor [9].

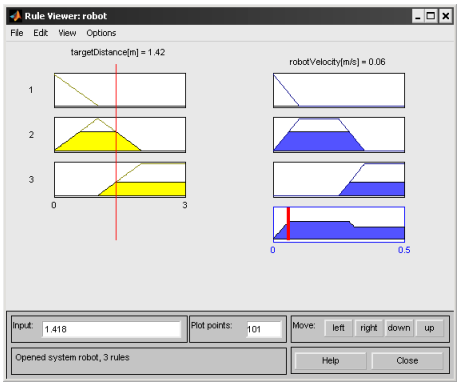
Defuzzifikačních metod je k dispozici celá řada. Některé z nich, například left of maximum, right of maximum či mean of maximum získávají výslednou výstupní ostrou hodnotu pouze z dominantní výstupní fuzzy množiny. Jiné metody, mezi které patří především často využívaná metoda centrum of gravity, bere v úvahu všechny fuzzy množiny výstupní proměnné, které mají stupeň příslušnosti větší než nulový. Aby bylo možné porovnat tyto metody a následně vybrat pokud možno tu nejvhodnější, je třeba provést jednoduchou simulaci, na jejíž základě budou získány průběhy jednotlivých defuzzifikačních metod. Pro testování byla zvolena situace, kdy se robot blíží k cílové poloze a v závislosti na vzdálenosti k této poloze snižuje rychlost. Vstupním parametrem bude tedy vzdálenost robotu k cíli a výstupem rychlost robotu. Hodnoty proměnných a jejich funkce příslušnosti jsou zřejmé z následujících obrázků.



Obr. 19 Průběhy funkcí příslušnosti fuzzy množin slovních proměnných "target distance" (vlevo) a "robot velocity" (vpravo).

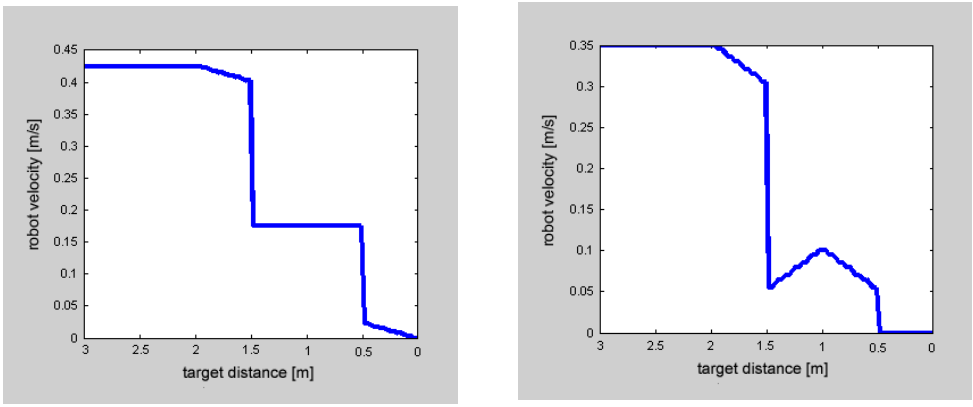
Navržena jsou následující tři pravidla:
IF target distance is *LARGE* THEN robot velocity is *LARGE*,
IF target distance is *SMALL* THEN robot velocity is *SMALL*,
IF target distance is *ZERO* THEN robot velocity is *ZERO*.

Již na základě těchto tří pravidel je možně získat zajímavé průběhy výstupní proměnné. K simulaci je využito programu Matlab a jeho toolboxu pro práci s fuzzy logikou. V tomto prostředí je díky funkci *Rule Viewer* možné ihned zobrazit hodnotu výstupní proměnné na základě zadání ostré vstupní hodnoty. Je tak patrné, z jaké fuzzy množiny či kombinací fuzzy množin je výstupní hodnota získána.

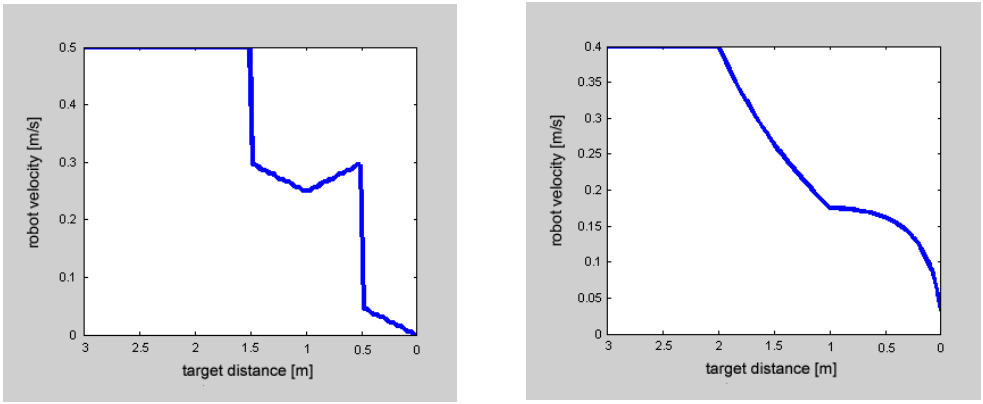


Obr. 20 Rule viewer.

Dále je pak možné pomocí funkce *Surface Viewer* získat zmiňované průběhy výstupní veličiny v závislosti na vstupní. Na následujících obrázcích jsou získané průběhy.



Obr. 21 Průběhy defuzzifikace MOM (vlevo) a LOM (vpravo).

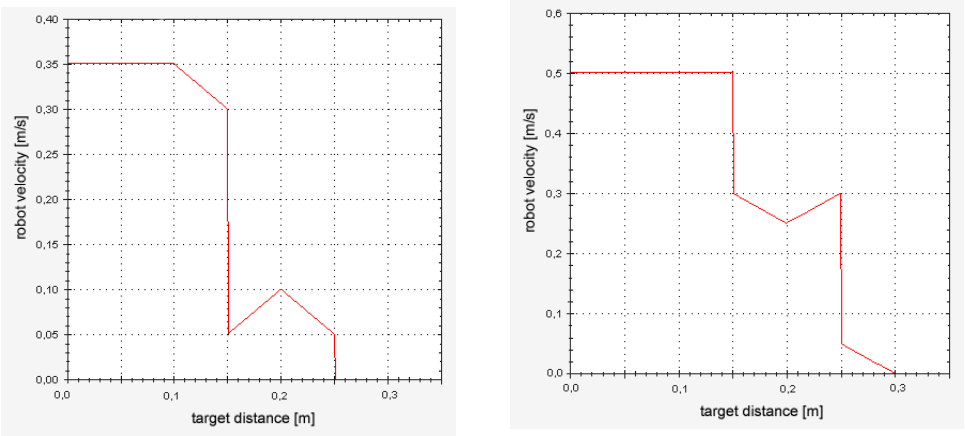


Obr. 22 Průběhy defuzzifikace ROM (vlevo) a COG (vpravo).

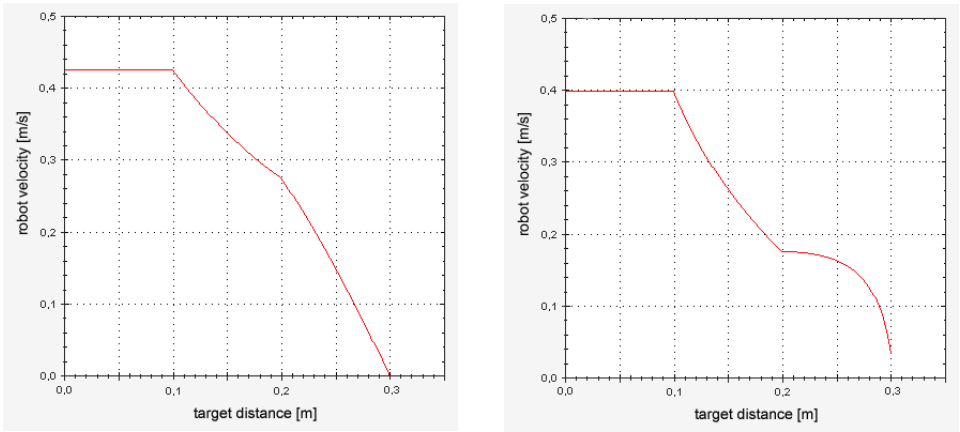
Z uvedených výsledků je patrné že metody získávající výslednou hodnotu pouze z dominantní fuzzy množiny vykazují náhlé skokové změny výstupní hodnoty v situacích, kdy se mění dominantní fuzzy množina. V případě metod left of maximum a right of maximum hodnota rychlosti robotu

dokonce v jistém okamžiku opět stoupá, což v tomto řešeném problému zajisté není žádoucí. Jako nejvhodnější se tak jeví v této situaci využití metody center of gravity, kde je průběh nejplynulejší.

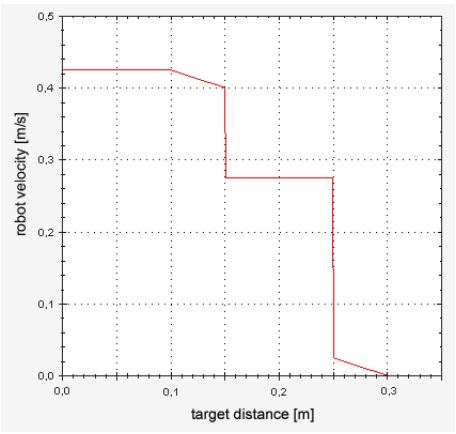
V simulačním prostředí, které bude prezentováno v samostatné kapitole, je implementováno 5 defuzzifikačních metod. K předcházejícím testovaným tak přibývá metoda center of maximum. Na následujících obrázcích jsou prezentovány průběhy získané právě ze zmiňovaného simulačního prostředí a to především z důvodu ověření správnosti implementace těchto metod. Výsledky odpovídají těm, které byly získány z Matlabu. Především nově prezentovaná metoda center of maximum se jeví jako výhodná k řešení daného problému. Nadále bude tedy volena tato metoda.



Obr. 23 Průběhy defuzzifikace LOM (vlevo) a ROM (vpravo).



Obr. 24 Průběhy defuzzifikace COM (vlevo) a COG (vpravo).



Obr. 25 Průběh defuzzifikace MOM.

5.3 Návrh řízení robotu

Návrh řídicího systému robotu se skládá z několika fází, přičemž každá z těchto fází má zásadní vliv na celkové chování výsledného systému. Hlavním prvkem řídicího systému robotu je fuzzy regulátor, jehož schéma je na obr. 18. Nejprve je nutné definovat vstupní a výstupní proměnné a na jejich základě lingvistické proměnné. V další fázi jsou navrženy pro každou lingvistickou proměnnou fuzzy množiny. Poté následuje tvorba pravidel, která popisují chování robotu v jednotlivých situacích.

5.3.1 Zavedení vstupních a výstupních proměnných

Volba výstupních proměnných je předurčena tím, že k řízení robotu potřebujeme znát v každém okamžiku rychlost, kterou se má pohybovat a úhel natočení robotu. Vstupní proměnné vycházejí především z informací o okolí, které je robot schopen získat. Vzhledem k tomu, že senzorický systém je rozdělen do tří částí, budeme uvažovat také tři vstupní proměnné udávající hodnoty o výskytu překážek v těchto oblastech. K navedení robotu k cíli je potřeba mít k dispozici vstupní proměnné udávající úhel mezi cílovou pozicí a osou robotu a vzdálenost cílové pozice. Aby bylo možné využívat tyto vstupní a výstupní proměnné ve fuzzy logice, je třeba zavést slovní neboli lingvistické proměnné. Proměnné jsou tedy definovány následovně:

Vstupní proměnné:

- d (nezávislá) vzdálenost středu robotu od středu cílové polohy [m]
  slovní proměnná X = "TARGET DISTANCE"
  $U \subset R$

- α (nezávislá) úhel mezi cílem a osou robotu [°]
  slovní proměnná X = "TARGET ANGLE"
  $U \subset R$

- s_{front} (nezávislá) vzdálenost překážky signalizovaná přední oblastí senzorů [m]
  slovní proměnná X = "FRONT SENSOR"
  $U \subset R$

- s_{left} (nezávislá) vzdálenost překážky signalizovaná levou oblastí senzorů [m]
  slovní proměnná X = "LEFT SENSOR"
  $U \subset R$

- s_{right} (nezávislá) vzdálenost překážky signalizovaná pravou oblastí senzorů [m]
  slovní proměnná X = "RIGHT SENSOR"
  $U \subset R$

Výstupní proměnné:

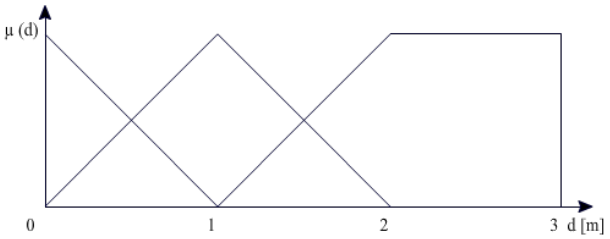
- v (závislá) rychlost robotu [m / s]
  slovní proměnná X = "ROBOT VELOCITY"
  $U \subset R$

- θ (závislá) změna natočení robotu [° / s]
  slovní proměnná X = "ROBOT ANGLE"
  $U \subset R$

5.3.2 Volba fuzzy množin

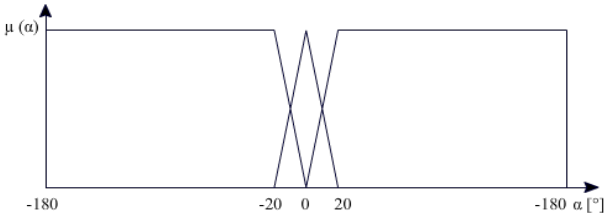
V této fázi je potřeba navrhnout fuzzy množiny pro jednotlivé slovní proměnné. Jak již bylo pojednáno v teoretické části, tento proces vychází především ze zkušeností a následného testování. Zejména je nutné pokrýt celé univerzum nosiči příslušných fuzzy množin a to tak, aby nemohla nastat situace, že v jednom okamžiku nezískáme ani jednu fuzzy množinu se stupněm příslušnosti větším než nula. Ovšem jako vhodnější se jeví navrhnout takzvané ε pokrytí, kdy se funkce příslušnosti protínají v úrovních >= ε. Tato vlastnost zaručuje, že existuje alespoň jedno dominantní pravidlo. V jistých případech jsou aktivováná dvě pravidla pro stejnou ostrou hodnotu se stupněm příslušnosti $\mu = \varepsilon$. Hodnotu ε je třeba volit alespoň 0,5. Volba fuzzy množin je tedy následující.

X = "TARGET DISTANCE"
T = {ZERO, NEAR, LARGE}



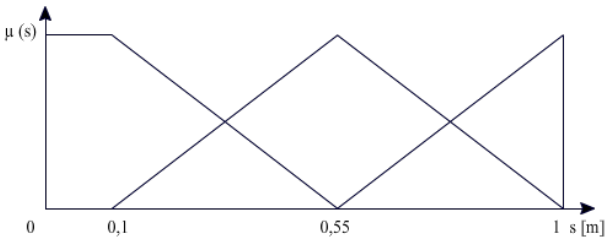
Obr. 26 Fuzzy množiny slovní proměnné "target distance".

X = "TARGET ANGLE"
T = {LEFT, ZERO, RIGHT}



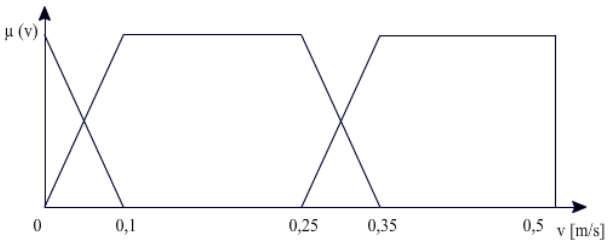
Obr. 27 Fuzzy množiny slovní proměnné "target angle".

X = "FRONT SENSOR"
T = {OBSTACLE, OBSTACLE NEAR, OBSTACLE FAR}
X = "LEFT SENSOR"
T = {OBSTACLE, OBSTACLE NEAR, OBSTACLE FAR}
X = "RIGHT SENSOR"
T = {OBSTACLE, OBSTACLE NEAR, OBSTACLE FAR}



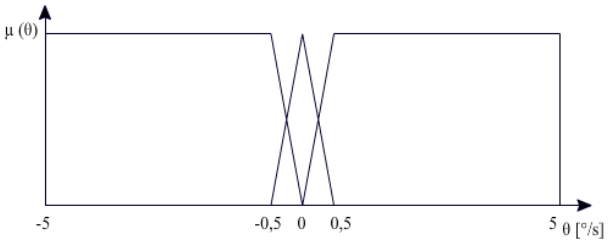
Obr. 28 Fuzzy množiny pro slovní proměnné senzorů.

X = "ROBOT VELOCITY"
T = {ZERO, SMALL, LARGE}



Obr. 29 Fuzzy množiny slovní proměnné "robot velocity".

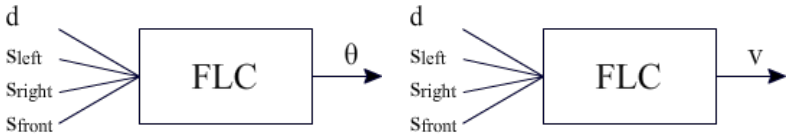
X = "ROBOT ANGLE"
T = {LEFT, ZERO, RIGHT}



Obr. 30 Fuzzy množiny slovní proměnné "robot angle".

5.3.3 Tvorba pravidel

V případě, že jsou již definovány slovní proměnné a jejich hodnoty ve formě fuzzy množin, je možné začít s návrhem pravidel. Využití fuzzy logiky v případě řízení je vhodné především v situacích, kde nedokážeme problém přesně matematicky popsat, ale jsme schopni ho vzhledem ke zkušenostem řešit. Znamená to tedy, že proces návrhu pravidel je tedy procesem přenesení zkušeností odborníka na daný problém do formy, která bude řešitelná pomocí matematického přístupu. Bude využito dvou fuzzy regulátorů. První z regulátorů bude sloužit k získávání rychlosti robotu, přičemž vstupními hodnotami bude vzdálenost cílové polohy *d* a údaje ze senzorů s_{left}, s_{front}, s_{right}.



Obr. 31 Schémata použitých fuzzy regulátorů.

Pravidla by v případě tohoto regulátoru měla být navrhnuta tak, aby robot snížil rychlost v závislosti na blížící se překážce či cíli. Pokud by robot dosáhl cílové polohy, měl by zastavit stejně jako v případě, že by se vyskytla překážka přímo před robotem ve směru jeho pohybu. Zastavit by ovšem neměl v případě, pokud by překážka nebránila přímo pohybu. Pravidla jsou tedy navržena následovně:

IF target distance is *ZERO* OR front sensor is *OBSTACLE* THEN robot velocity is *ZERO*
IF target distance is *NEAR* AND NOT front sensot is *OBSTACLE* THEN robot velocity is *SMALL*
IF NOT tatget distance is *ZERO* AND front sensor is *OBSTACLE NEAR*
THEN robot velocity is *SMALL*
IF NOT target distance is *ZERO* AND NOT front sensor is *OBSTACLE* AND right sensor is
OBSTACLE THEN robot velocity is *SMALL*
IF NOT target distance is *ZERO* AND NOT front sensor is *OBSTACLE* AND left sensor is
OBSTACLE THEN robot velocity is *SMALL*
IF target distance is *LARGE* AND front sensor is *OBSTACLE FAR* AND NOT left sensor is
OBSTACLE AND NOT right sensor is *OBSTACLE* THEN robot velocity is *LARGE*

V případě pravidel pro získání změny natočení robotu není návrh již tak jednoznačný. Zde se naskytuje již možnost ovlivnění dráhy robotu volbou fuzzy množin v pravidlech. Lze tak například ovlivnit, v jaké vzdálenosti od překážky se robot začne vyhýbat, či v jaké fázi objíždění překážky se začne opět natáčet směrem k cíli. V některých variantách je ovšem možná i kolize s překážkou. Proto je velice důležité vhodně zvolit pravidla a důkladně je otestovat. Další korekce mohou být provedené především v parametrech fuzzy množin. Aby bylo zajištěné, že bude každý ze stavů, ve kterém se může robot vyskytnout, popsán pravidlem, je třeba definovat 81 pravidel. Počet pravidel je dán 4 vstupními proměnnými o třech možných hodnotách. Pravidla jsou v následujícím tvaru:

IF target angle is *LEFT* AND front sensor is *OBSTACLE FAR* AND left sensor is *OBSTACLE FAR* AND right sensor is *OBSTACLE FAR* THEN robot angle is *LEFT*

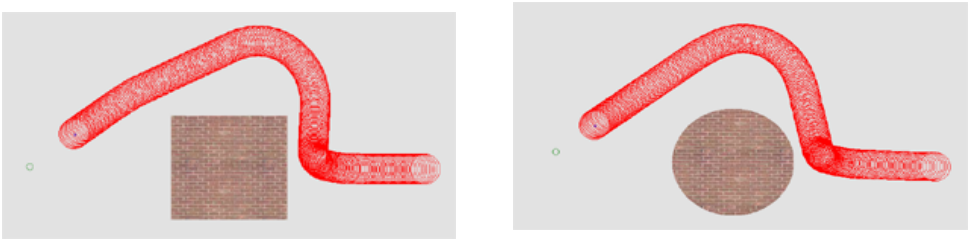
V situaci, která je popsaná výše uvedeným pravidlem je reakce jasná. Cíl se nachází vlevo od osy směru pohybu robotu, senzory na trase mezi robotem a cílem nesignalizují překážku, tudíž je možné robot bez problému navigovat k cíli. Zajímavější je ovšem situace zobrazená na níže uvedených obrázcích, kde robot objíždí překážku. Otázkou v této situaci je tedy, v jaké fázi robot opět směrovat k cíli. Tato situace je popsána především následujícími pravidly, kdy je robot směrován k cíli:

IF target angle is *LEFT* AND front sensor is *OBSTACLE FAR* AND left sensor is *OBSTACLE FAR* AND right sensor is *OBSTACLE FAR* THEN robot angle is *LEFT*
IF target angle is *LEFT* AND front sensor is *OBSTACLE FAR* AND left sensor is *OBSTACLE NEAR* AND right sensor is *OBSTACLE FAR* THEN robot angle is *LEFT*
IF target angle is *LEFT* AND front sensor is *OBSTACLE FAR* AND left sensor is *OBSTACLE* AND right sensor is *OBSTACLE FAR* THEN robot angle is *LEFT*

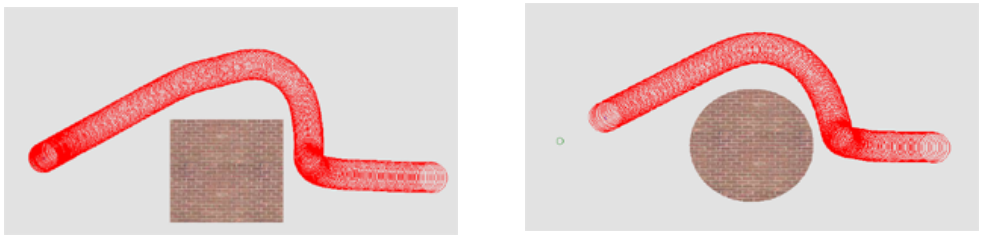
V případě, že by robot nebyl směrován k cíli by byla pravidla následující:

IF target angle is *LEFT* AND front sensor is *OBSTACLE FAR* AND left sensor is *OBSTACLE FAR* AND right sensor is *OBSTACLE FAR* THEN robot angle is ZERO
IF target angle is *LEFT* AND front sensor is *OBSTACLE FAR* AND left sensor is *OBSTACLE NEAR* AND right sensor is *OBSTACLE FAR* THEN robot angle is *ZERO*
IF target angle is *LEFT* AND front sensor is *OBSTACLE FAR* AND left sensor is *OBSTACLE* AND right sensor is *OBSTACLE FAR* THEN robot angle is ZERO

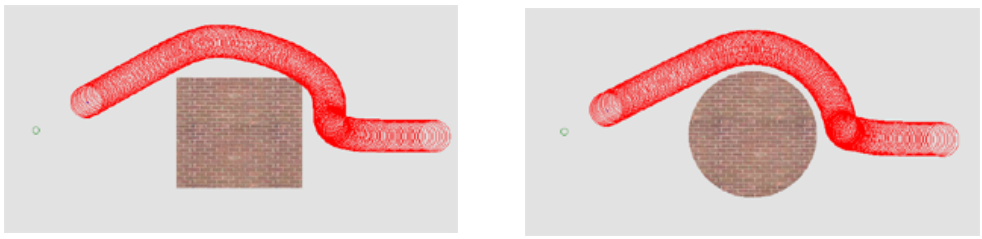
Základním předpokladem je, že přední senzor nesignalizuje překážku. V předchozích pravidlech je tedy možnost rozhodnout se, zda robot pojede stále rovně (robot angle is *ZERO*), nebo již začne otáčet vlevo (robot angle is *LEFT*). Z obrázků je zřejmé, že pokud je robot směrován na cíl, i když levý senzor signalizuje překážku přímo u robotu, objíždí kruhovou překážku takřka ideálně ovšem při jiném tvaru překážky nastaly situace, kdy došlo ke kolizi. Druhý extrém je takový, kdy robot směřuje k cíli až ve chvíli, kdy levý senzor signalizuje překážku ve velké vzdálenosti což způsobuje zbytečně velké úhybné manévry. Jako optimální se tedy jeví střední možnost, což znamená, že levý senzor nesignalizuje překážku přímo u robotu ale signalizuje ji v jeho blízkosti. V tu chvíli je robot natáčen směrem k cíli.



Obr. 32 Průjezd robotu okolo překážky v případě, že levý senzor signalizuje překážku daleko (left sensor is OBSTACLE FAR).

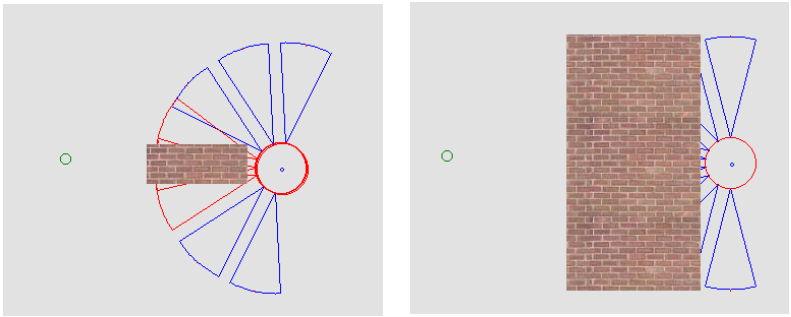


Obr. 33 Průjezd robotu okolo překážky v případě, že levý senzor signalizuje překážku blízko (left sensor is OBSTACLE NEAR).



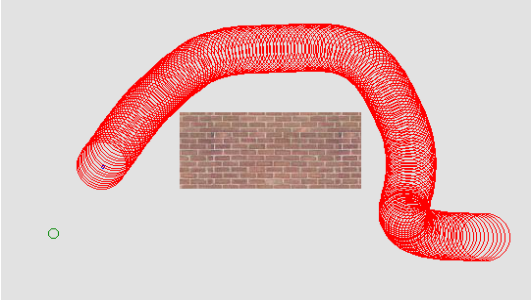
Obr. 34 Průjezd robotu okolo překážky v případě, že levý senzor signalizuje překážku přímo u robotu (left sensor is OBSTACLE).

Výraznější problém při takto navrženém řídicím systému nastává v situaci, kdy přední senzor signalizuje překážku a cílová pozice je na úrovni osy robotu. V takovém případě jsou navržena pravidla tak, aby robot volil směr pohybu vpravo. Pokud se ovšem začne natáčet vpravo, hodnota vstupní proměnné udávající úhel natočení cílové pozice se změní z "target angle is *ZERO*" na hodnotu "target angle is *LEFT*" a v takovém případě se robot začne otáčet vlevo a v jistém okamžiku, to jest na hranici, kde se protíná fuzzy množina s hodnotou LEFT a ZERO robot uvízne. To je způsobeno především tím, že v těchto okamžicích boční senzory nemohou ještě detekovat překážku. Situace je zřetelná z obr. 35 (vlevo). Na obrázku obr. 35 (vpravo) je zobrazena další varianta tohoto problému. V tomto případě signalizují všechny tři skupiny senzorů překážky. Robot se opět začne otáčet až do chvíle, kdy dosáhne hranice průniku fuzzy množin LEFT a ZERO.



Obr. 35 Případy uvíznutí robotu.

Řešit tento problém je možné několika způsoby. Jednak by bylo možné přeorganizovat senzory tak, abychom dostali co nejpodrobnější informace o poloze překážky vzhledem k natočení robotu, což by ale znamenalo zvýšit počet vstupních proměnných ze senzorů a nějakým vhodným způsobem z nich odvozovat polohu překážky. Tato varianta je zajisté obtížnější a proto se nadále přikloníme k dosavadnímu systému senzorů. Další možností je využívání virtuálních cílů, což by zajisté vedlo k úspěšnému řešení daného problému, nicméně virtuální cíle budou využity v jiné části návrhu systému řízení robotu. Nejjednodušší řešení problému je takové, že by robot objížděl překážku vždy v předem určeném směru. V tomto případě volí robot vždy směr vpravo.



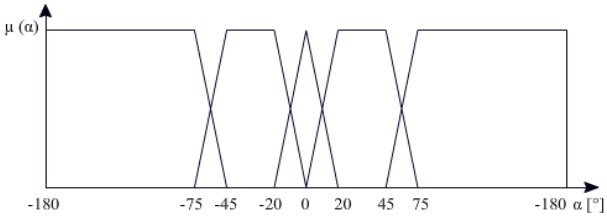
Obr. 36 Průjezd robotu kolem překážky.

V jistých případech se tento přístup jeví jako neoptimální (obr. 36). Robot objíždí překážku zprava i když by bylo výhodnější zvolit směr vlevo. Nedostatek je způsobený spíše faktem, že robot detekuje překážku nejprve předním senzorem a ihned začne úhybný manévr. Pokud by se začal vyhýbat překážce až ve chvíli, kdy by se k ní dostal blíže, stačil by ji zachytit pravým senzorem, což by způsobilo, že by robot objížděl překážku zleva. Nicméně robot nedisponuje znalostmi o tvaru překážky a proto je pouze na základě informací se senzorů zcela nemožné optimálně zvolit směr, kterým by měla být překážka objížděna.

Další zajímavou myšlenkou, jak vylepšit stávající řešení, je optimalizovat proces, kdy je robot natočen směrem od cíle ve větší vzdálenosti. V takovém případě je získána z regulátoru maximální hodnota rychlosti, což vede k tomu, že se robot vydá k cíli nejprve tak, že opisuje kruhovou trajektorii.

V tomto případě by bylo vhodné využít schopnosti robotu otáčet se na místě. Řešením je zvýšení hodnot slovní proměnné "target angle" ze tří na pět.

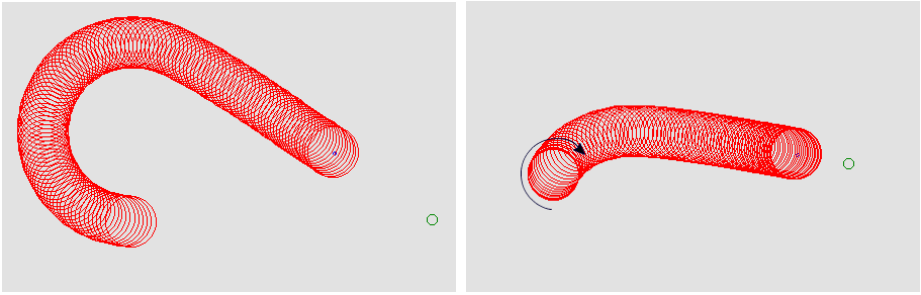
X = "TARGET ANGLE"
T = {LEFT, SMALL LEFT, ZERO, SMALL RIGHT, RIGHT}



Obr. 37 Modifikace počtu fuzzy množin.

Dále je třeba upravit bázi pravidel. Jednak modifikovat pravidlo, které předepisuje maximální rychlost robotu a přidat pravidlo popisující stav, kdy je hodnota úhlu cílové polohy v oblasti fuzzy množin LEFT nebo RIGHT.

IF target distance is *LARGE* AND front sensor is *OBSTACLE FAR* AND NOT left sensor is *OBSTACLE* AND NOT right sensor is *OBSTACLE* AND NOT target angle is *LEFT* AND NOT target angle is *RIGHT* THEN robot velocity is *LARGE*
IF target angle is *LEFT* OR target angle is *RIGHT* THEN robot velocity is *ZERO*



Obr. 38 Modifikace rychlosti robotu v případě, kdy se cíl nachází za robotem. Původní přístup(vlevo). Modifikovaný přístup (vpravo). V tomto případě se robot nejprve otáčí na místě.

Tato modifikace se ukázala ovšem při testování jako zcela nevhodná. Pokud robot objíždí překážku, může nastat situace, kdy se cílová poloha nachází na druhé straně překážky a úhel cílové pozice je v oblasti nově přidaných fuzzy množin, což vede k tomu, že robot zastaví a nemůže překážku objet. Výhodnějším přístupem by bylo, kdyby se rychlost robotu nesnižovala v závislosti na poloze cílové pozice, ale v závislosti na požadované rychlosti natáčení robotu. Takový přístup by řešil jednak prezentovanou situaci, kdy je cílová pozice za robotem, ale zároveň by byla rychlost snížena v každé situaci, kdy by robot prudce zatáčel. Při tomto řešení již problém se zastavením robotu odpadá. Je však nutné do fuzzy regulátoru, který slouží k získání rychlosti robotu, přivést další vstupní proměnnou, kterou by byla přímo výstupní proměnná z regulátoru pro získání úhlu natočení a následně poté modifikovat pravidla. Dále bude od této modifikace upuštěno a bude využíván přístup použitý v základním návrhu řídicího systému.

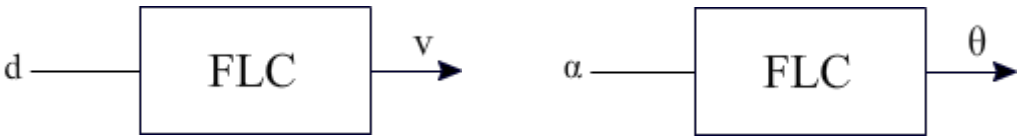
5.4 Dekompozice dosavadního řešení

Často využívaným přístupem při návrhu systému řízení robotu je dekompozice na jednotlivá chování. Tato chování robotu jsou pak spouštěna v odpovídajících situacích. Mezi taková chování patří například objíždění překážky, směřování k cíli či sledování stěny nebo koridoru. Výhoda tohoto přístupu je především ve fázi návrhu pravidel. Rozdělením na chování se zmenší počet vstupních proměnných do jednotlivých regulátorů, čímž se sníží počet pravidel a zjednoduší jejich návrh. Nevýhodou ovšem je zvýšení počtu regulátorů a nutnost rozhodovat, které chování bude v danou chvíli využíváno.

Předchozí řešení je možné modifikovat tak, že zavedeme dva druhy chování, a to chování, které bude mít za úkol směřovat robot k cíli, a chování, jež zajistí, že se robot nedostane do kolize s překážkou. V tomto jednoduchém případě nejsou výhody tohoto přístupu až tak zřejmé, nicméně je třeba tento princip otestovat pro pozdější využití ve složitější variantě.

směřování k cíli (goal seeking - GS)

Toto chování bude mít za úkol, jak již z názvu plyne, navést robot k cílové poloze. Za tímto účelem jsou navrženy dva regulátory, které opět slouží k získání požadované rychlosti a natočení robotu. Vzhledem k tomu, že rychlost robotu v tomto chování je ovlivněna pouze vzdáleností cílové polohy, bude do regulátoru vstupovat pouze jedna proměnná, a to právě vzdálenost cílové polohy. Vstupní (*target distance*) a výstupní (*robot velocity*) proměnné regulátoru tedy odpovídají těm, které byly použity již v předchozím řešení a to včetně odpovídajících fuzzy množin. Rozdíl je především v počtu pravidel, která jsou pouze tři. Tato úloha a tedy i její pravidla zcela odpovídají úloze, která byla využita pro získání průběhů deffuzzifikačních metod.



Obr. 39 Schéma fuzzy regulátorů pro získávání v (vlevo) a θ (vpravo).

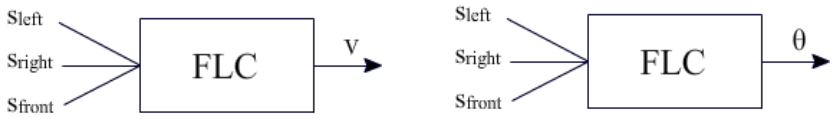
U regulátoru, určeného k získání změny úhlu natočení, je na vstup v případě tohoto chování přivedena pouze proměnná udávající natočení cílové polohy relativně vůči ose robotu. Báze pravidel regulátoru obsahuje poté vzhledem k jedné vstupní a jedné výstupní proměnné pouze tři pravidla popisující chování robotu. Pravidla jsou v tomto případě navržena zcela logicky následovně:

IF target angle is *LEFT* THEN robot angle is *LEFT*
IF target angle is *ZERO* THEN robot angle is *ZERO*
IF target angle is *RIGHT* THEN robot angle is *RIGHT*

objíždění překážek (obstacle avoidance - OA)

Účelem tohoto chování je, jak již název napovídá, zajistit, aby se robot vyhnul konfliktu s překážkami, které se mohou vyskytnout na jeho trase. Vstupními proměnnými tedy budou údaje ze senzorů, na jejichž základě bude opět odvozena požadovaná hodnota rychlosti a natočení robotu. Jako

v případě předchozího chování budou i v tomto případě využívány proměnné s fuzzy množinami, které byly již navrženy v předchozím přístupu navigace robotu.



Obr. 40 Schéma fuzzy regulátorů pro získávání v (vlevo) a θ (vpravo).

U návrhu pravidel pro regulátor sloužící k získávání požadované rychlosti robotu je řešení vcelku jednoznačné. Robot by měl snižovat rychlost, pokud se blíží k překážce. Pokud je v těsné blízkosti překážky, měl by zastavit a v případě objíždění překážky v těsné blízkosti by opět měl svoji rychlost snížit. Pravidla jsou tedy obdobná těm, která byla navržena v původním systému řízení s tím, že v tomto případě se nebere v úvahu vzdálenost cílové polohy.

IF front sensor is *OBSTACLE* THEN robot velocity is *ZERO*
IF NOT front sensor is *OBSTACLE* AND right sensor is *OBSTACLE* THEN robot velocity is *SMALL*
IF NOT front sensor is *OBSTACLE* AND left sensor is *OBSTACLE* THEN robot velocity is *SMALL*
IF front sensor is *OBSTACLE FAR* AND NOT left sensor is *OBSTACLE* AND NOT right sensor is *OBSTACLE* THEN robot velocity is *LARGE*
IF front sensor is *OBSTACLE NEAR* AND NOT left sensor is *OBSTACLE* AND NOT right sensor is *OBSTACLE* THEN robot velocity is *SMALL*

V případě regulátoru určeného k získání změny natočení robotu jsou na vstup přivedeny také proměnné reprezentující údaje ze senzorů. V původním návrhu systému řízení bylo vzhledem k tomu, že na vstupu byla navíc hodnota natočení cílové polohy, zapotřebí 81 pravidel. Návrh takového množství pravidel je již celkem obtížný. V případě tohoto přístupu stačí k popsání všech možných situací 27 pravidel, což je z hlediska návrhu mnohem jednodušší. V tomto je tedy znatelná jedna z výhod dekompozičního přístupu. Všechna pravidla jsou zapsána v následujícím tvaru s patřičnými změnami v závislosti na daných fuzzy množinách.

IF front sensor is *OBSTACLE* AND left sensor is *OBSTACLE FAR* AND right sensor is *OBSTACLE* THEN robot angle is *LEFT*

Pravidla jsou navržena tak, že pokud má robot stejné hodnoty jak z levého tak pravého senzoru, je na výstupu hodnota *robot angle* is ZERO. Znamená to tedy, že se tento regulátor nepřiklání ani k jednomu směru a rozhodnutí nechává na regulátoru, jenž odvozuje hodnotu natočení na základě cílové pozice. Ovšem pokud je ze senzorů zřejmé, která oblast je vhodnější pro úhybný manévr, volí robot tuto oblast pro směr dalšího pohybu.

Zásadním krokem při dekompozičním přístupu je ovšem rozhodování, kdy a s jakou mírou dané chování využít. Za tímto účelem je přidán regulátor, který na základě vstupů ze senzorů rozhoduje o využití jednotlivých chování. Je tedy zavedená nová výstupní proměnná a navrženy její možné hodnoty ve formě fuzzy množin.

- w_{OA} (závislá) váhový koeficient pro OA
  slovní proměnná X = "OA WEIGHT"
  $U \subset R$
  T = {ZERO, SMALL, LARGE}

Obr. 41 Rozložení fuzzy množin pro slovní proměnnou OA WEIGHT.

Navržený váhový koeficient udává tedy, s jakou mírou bude v určité situaci využíváno chování zajišťující předcházení kolizím s překážkami. Především na základě experimentů byla nakonec navržena níže uvedená pravidla, která v podstatě odpovídají pravidlům pro získávání požadované rychlosti.

Obr. 42 Schéma fuzzy regulátorů pro získávání hodnoty OA WEIGHT.

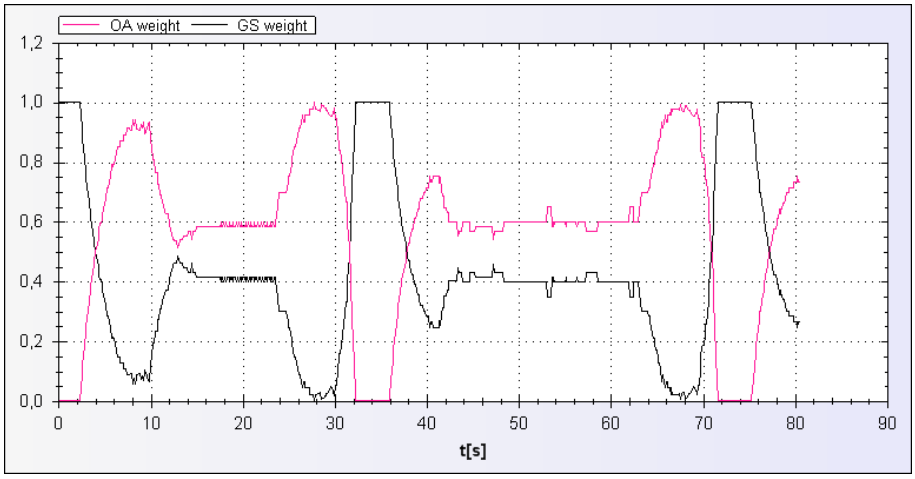
IF front sensor is *OBSTACLE* THEN weight is *ZERO*
IF NOT front sensor is *OBSTACLE* AND right sensor is *OBSTACLE* THEN weight is *SMALL*
IF NOT front sensor is *OBSTACLE* AND left sensor is *OBSTACLE* THEN weight is *SMALL*
IF front sensor is *OBSTACLE FAR* AND NOT left sensor is *OBSTACLE* AND NOT right sensor is *OBSTACLE* THEN weight is *LARGE*
IF front sensor is *OBSTACLE NEAR* AND NOT left sensor is *OBSTACLE* AND NOT right sensor is *OBSTACLE* THEN weight is *SMALL*

Nyní je ovšem ještě nutné odvodit váhový koeficient pro směrování robotu k cílové poloze. Vzhledem k tomu, že jsou navržena pouze dvě chování, je možné dopočítat druhý váhový koeficient w_{GS} dle následujícího vzorce:

$$w_{GS} = 1 - w_{OA},$$

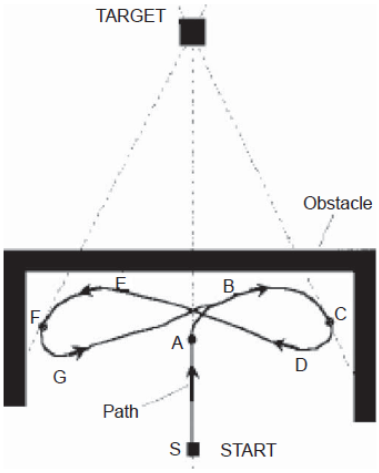
a požadovanou rychlost a změnu natočení robotu dopočítat dle rovnic:

$$v = w_{GS} . v_{GS} + w_{OA} . v_{OA},$$
$$\theta = w_{GS} . \theta_{GS} + w_{OA} . \theta_{OA}.$$



Obr. 43 Průběhy váhových koeficientů získaných v simulačním prostředí.

Experimentální výsledky tohoto i ostatních návrhů systému řízení budou prezentovány v samostatné kapitole, kde bude také provedeno jejich vzájemné porovnání. Dosavadní přístupy dosahují kvalitních výsledků pouze v případě, že se v prostředí robotu nevyskytují konkávní překážky. V opačném případě mohou nastat situace, kdy robot uvízne v lokálním minimu. Tento nedostatek je způsobený především faktem, že robot využívá reaktivní řídicí systém a nedisponuje informacemi z předchozích kroků řízení. Z obr. 44 je patrná situace, kdy robot uvízne v lokálním minimu. Robot jede směrem k cíli, dokud přední senzory nedetekují překážku. V tu chvíli se začne robot vyhýbat vpravo. Po určitém úseku přední senzory opět detekují překážku, což způsobí další natáčení robotu vpravo. Tím se cílová pozice přesune na pravou stranu relativně od osy robotu, což vede robot k dalšímu natáčení. Ta samá situace se opakuje zrcadlově i v druhé části překážky. Robot je tak uvězněn v nekonečné smyčce.



Obr. 44 Případ uvíznutí robotu v lokálním minimu [10].

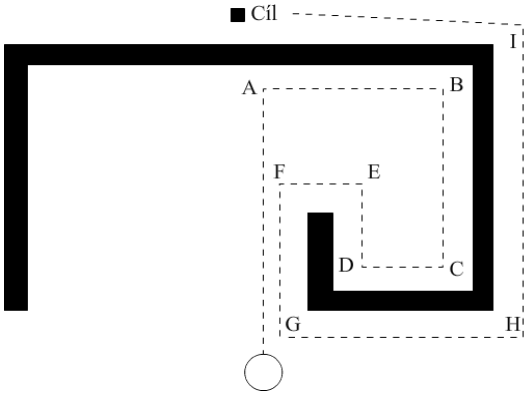
V následujících návrzích systému řízení robotu budou využity dva přístupy, s jejichž pomocí je řešen tento problém.

5.5 Dekompozice s využitím virtuálních cílů

Jedním z často využívaných přístupů při řešení problémů s konkávními překážkami jsou virtuální cíle. Hlavní myšlenkou těchto metod je záměna skutečné cílové polohy za virtuální. Jednotlivé varianty se liší především v následujících vlastnostech:

- spouštěcím mechanismem, který zapříčiní změnu cílové pozice
- způsobem, jakým je nalezena poloha virtuálního cíle
- mechanismem, který způsobí změnu opět na skutečný cíl

V této práci je testován jednoduchý přístup, vycházející z [10], který je založen na sčítání úhlu natočení. Modifikován bude řídicí systém, založený na dekompozici, prezentovaný v předcházející kapitole. Základ tohoto systému zůstane nezměněn. Princip metody je vysvětlen na následujícím příkladu.



Obr. 45 Princip metody využívající sčítání úhlů.

Na obrázku robot směřuje k cíli, až do bodu A, kde detekuje překážku. Ve chvíli, kdy přední senzor zaznamená překážku a cíl je v oblasti před robotem, to znamená relativní úhel cílové polohy vůči ose robotu je v rozmezí -45° až 45°, je spuštěno sčítání úhlů a nastavena nová virtuální cílová poloha. Robot se začne vyhýbat překážce směrem vpravo a celkový úhel natočení se dostane na hodnotu přibližně 90°. Vzhledem k tomu, že úhel je kladný, tak virtuální cíl je nastaven na -90° relativně od osy robotu. Znamená to tedy, že pokud se robot otáčí vpravo a úhel natočení vzrůstá, je snaha o to, aby se úhel natočení vykompenzoval a vrátil se zpět na hodnotu přibližně 0°, kde bude cílová poloha přepnuta z virtuální na reálnou. Tímto způsobem si robot dokáže poradit s některými variantami konkávních překážek. V bodě B ilustračního příkladu robot pokračuje v natáčení směrem vpravo, čímž se úhel dostává na hodnotu cca 180°. V případě předchozího systému řízení by v tuto chvíli robot, vzhledem k tomu, že by detekoval cílovou polohu na pravé straně, začal směřovat opět přibližně do bodu A, což by následně vedlo k zacyklení. V tomto případě je ovšem virtuální poloha cíle stále na -90° a robot pokračuje podél stěny až do bodu C a následně pak bodu E, kde je již hodnota úhlu 360°. V bodě E se však již naskytne možnost kompenzovat úhel a robot se začne natáčet vlevo. Tím se dostává do bodu F a otáčí se na hodnotu cca 180°. Následuje pozice G, kde se otočí na hodnotu 90° a v bodě H se hodnota úhlu vrací zpět na 0°. V tuto chvíli je robot již směrován ke skutečnému cíli, jelikož cíle jsou přepínány zpět na skutečnou polohu, když se hodnota úhlu vykompenzuje na hodnotu -45° až 45°.

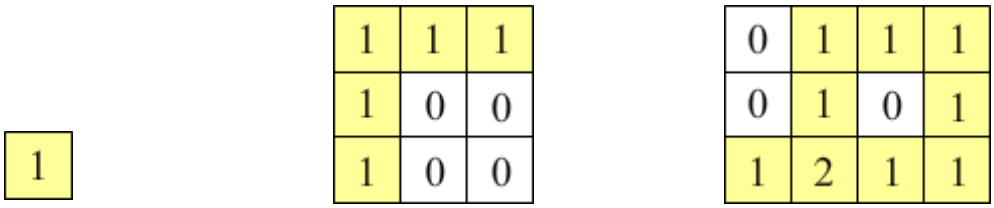
Jak bude prezentováno v kapitole věnované experimentům, dokáže si tento způsob poradit i v situacích, kde doposud navrhované systémy řízení selhávaly.

5.6 Dekompozice s využitím matice

Ani předchozí přístup, který byl prezentován nezaručí, že se robot vždy dostane do cílové polohy. K tomu, aby se zajistila větší šance na úspěch, je zapotřebí získat co nejvíce informací o prostředí robotu. Za tímto účelem bylo vyvinuto velké množství metod, které využívají různé způsoby reprezentace znalostí o prostředí v paměti robotu. Jedním ze způsobů, který bude využit v této práci, je zaznamenávání předešlé dráhy robotu do matice. Matice je pak využívána k volbě následného pohybu robotu a je tak zajištěno, že se robot nebude neustále vracet do míst, kde se již vyskytoval. Základní myšlenka přístupu vychází z práce [9].

Matice je navržena tak, že v počátku obsahuje pouze jeden prvek s počáteční polohou robotu a dále její rozměry narůstají podle pohybu robotu. Pokud se robot vyskytne v místě reprezentovaném

příslušnou buňkou matice, je její hodnota navýšena o hodnotu 1. Velikost buňky odpovídá velikosti robotu. K vytvoření matice je použito dynamické datové struktury a to seznamu.



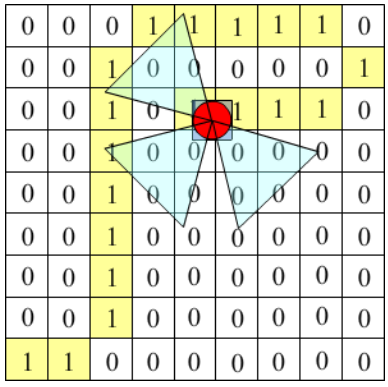
Obr. 46 Ukázka rozšiřování matice a zaznamenávání pohybu robotu.

Z matice jsou poté hodnoty získávány tak, že jsou v okolí robotu vytvořeny tři trojúhelníkové regiony a hodnota každého regionu je spočítána jako suma hodnot všech buněk, které se v daném regionu vyskytují. Do regionů nejsou započítávány buňky, jež zasahují do čtvercové oblasti přímo u robotu. Tato korekce je provedena především proto, aby nebyly brány v úvahu buňky, ve kterých se robot aktuálně vyskytuje a také buňky v blízkosti robotu, na které by nedokázal reagovat. Také buňky, nacházející se ve větší vzdálenosti od robotu, nejsou z důvodu zvýšení výkonnosti testovány, zdali se nachází v jednotlivých regionech. Následně jsou k dispozici tři hodnoty Ψ_{left}, Ψ_{front}, Ψ_{right}, jejichž hodnota je závislá na frekvenci výskytu robotu v daném regionu. Z těchto hodnot jsou následně odvozeny koeficienty pro jednotlivé regiony, které budou následně použity jako vstupní lingvistické proměnné:

$$\tau_{left} = \frac{\Psi_{left}}{max(\Psi_{left}, \Psi_{front}, \Psi_{right})},$$

$$\tau_{front} = \frac{\Psi_{front}}{max(\Psi_{left}, \Psi_{front}, \Psi_{right})},$$

$$\tau_{right} = \frac{\Psi_{right}}{max(\Psi_{left}, \Psi_{front}, \Psi_{right})}.$$



Obr. 47 Získávání informací z matice.

Nyní je možné již přistoupit k samotnému návrhu systému řízení. Ten bude opět vyvinut na základě dekompozice, která byla provedena již dříve. V podstatě půjde pouze o doplnění mechanismu, díky kterému bude možné při řízení využívat sektorové koeficienty získané z matice. Je však zapotřebí oproti původnímu řešení navýšit počet regulátorů o 3. U chování GS je nutné přidat regulátor pro získávání váhového koeficientu. Dva regulátory budou využity u chování, které bude sloužit k vyhledávání míst s menší frekvencí výskytu robotu.

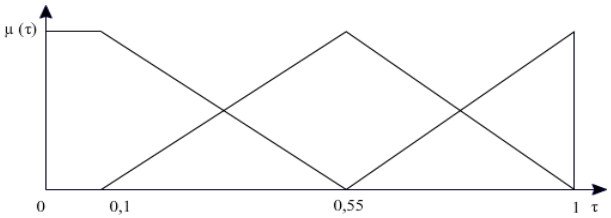
hledání cesty (path searching - PS)

Toto chování, jak již bylo řečeno, bude robot směrovat do oblastí, které ještě nebyly

prozkoumány či do oblastí, kde se robot vyskytoval, ale méně než v ostatních oblastech. Důležitým bodem je tedy způsob, jakým bude toto chování zakomponováno do celkového řídicího systému. Toto chování bude ovlivňovat pouze směr natočení robotu a nikoliv jeho rychlost, která bude nadále závislá pouze na vzdálenosti robotu od překážky a cílové polohy. K získání hodnoty změny natočení robotu je navržen regulátor, který bude mít na vstupech koeficienty z jednotlivých sektorů. Dále bude zapotřebí získat váhový koeficient, který bude udávat, s jakou mírou bude toto chování využíváno.

Obr. 48 Schéma fuzzy regulátorů pro získávání w (vlevo) a θ (vpravo) pro PS chování.

- $\tau_{left,\ front,\ right}$(nezávislá) sektorový koeficient
  slovní proměnná X = "LEFT SECTOR", "FRONT SECTOR", "RIGHT SECTOR"
  $U \subset R$
  T = {ZERO, SMALL, LARGE}



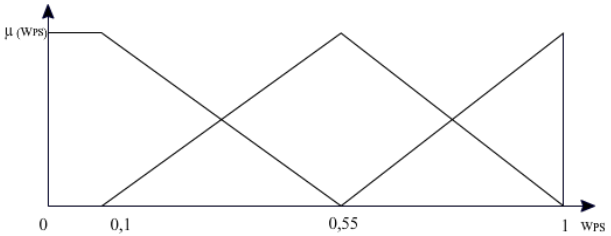
Obr. 49 Rozložení fuzzy množin pro vstupní proměnnou τ.

Nyní je možné navrhnout pravidla, která budou na základě sektorových koeficientů určovat změnu úhlu natočení robotu. Pravidel je v tomto případě 27 a jsou všechna v následujícím tvaru s patřičnými změnami fuzzy množin.

IF front sector is *ZERO* AND left sector is *SMALL* AND right sector is *SMALL* THEN robot angle is *LEFT*

Aby bylo možné jednotlivá chování zkombinovat, je třeba přidat další dva váhové koeficienty. A to jeden pro PS chování a dále pak pro GS chování, který nebyl v případě dekompozice na 2 chování potřeba. V dekompozici na 3 chování je však tento koeficient již nutný.

- w_{PS}(závislá) váhový koeficient pro PS
  slovní proměnná X = "PS WEIGHT"
  $U \subset R$
  T = {ZERO, SMALL, LARGE}

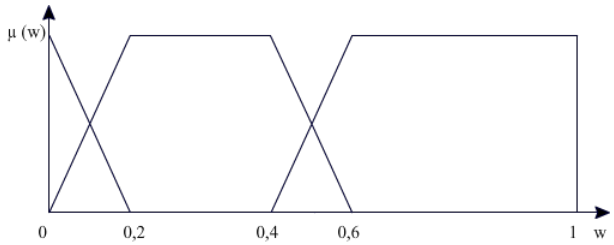


Obr. 50 Rozložení fuzzy množin pro výstupní proměnnou w_{PS}.

Pravidla jsou v případě získání w_{PS} navržena následovně:
IF front sector is *ZERO* THEN ps weight is *ZERO*
IF front sector is *SMALL* THEN ps weight is *SMALL*
IF front sector is *LARGE* AND left sector is *LARGE* THEN ps weight is *SMALL*
IF front sector is *LARGE* AND right sector is *LARGE* THEN ps weight is *SMALL*
IF front sector is *LARGE* AND NOT left sector is *LARGE* THEN ps weight is *LARGE*
IF front sector is *LARGE* AND NOT right sector is *LARGE* THEN ps weight is *LARGE*

Jak již bylo řečeno, je třeba získat také váhový koeficient pro GS chování.

- w_{GS}(závislá) váhový koeficient pro GS
  slovní proměnná X = "GS WEIGHT"
  $U \subset R$
  T = {ZERO, SMALL, LARGE}



Obr. 51 Rozložení fuzzy množin pro výstupní proměnnou w_{GS}.

U váhového koeficientu pro GS jsou navrhnuta tři jednoduchá pravidla.
IF ps weight is *ZERO* AND oa weight is *ZERO* THEN gs weight is *LARGE*
IF NOT ps weight is *ZERO* AND NOT oa weight is *ZERO* THEN gs weight is *ZERO*
IF ps weight is *ZERO* AND oa weight is *SMALL* THEN gs weight is *SMALL*

Nejdůležitější fází při návrhu tohoto systému řízení je zkombinování jednotlivých chování a doladění parametrů fuzzy množin váhových koeficientů. Jak již bylo řečeno, rychlost robotu je závislá pouze na OA a GS. Rychlost je tak možné získat stejným způsobem jako v základním dekompozičním přístupu podle vzorce:

$$v = (1 - w_{OA}) \cdot v_{GS} + w_{OA} \cdot v_{OA}$$

V případě změny úhlu natočení je kombinace výsledků z regulátorů jednotlivých chování provedena dle následující rovnice

$$\theta = \frac{w_{OA} \cdot \theta_{OA} + w_{PS} \cdot \theta_{PS} + w_{GS} \cdot \theta_{GS}}{w_{OA} + w_{PS} + w_{GS}}$$

V průběhu testování tohoto přístupu byly zjištěny některé nedostatky prezentovaného přístupu, které se nepodařilo odladit změnami v pravidlech či parametrech fuzzy množin. Řešení si nakonec vyžádalo spojení tohoto přístupu s odlehčenou verzí virtuálních cílů. Ve chvíli, kdy přední senzor detekuje překážku, kterou ovšem nezaznamenají boční senzory a dále je jeden z koeficientů bočních sektorů větší jak nula, je robot naváděn na virtuální cíl, který se nachází na té straně robotu, jejíž sektorový koeficient má menší hodnotu. Virtuální cíl je přepnut zpět na skutečný ve chvíli, kdy přední sektor přestane signalizovat překážku. Tím je dokončen návrh daného řídicího systému a jeho výsledky budou prezentovány v samostatné kapitole.

6 SIMULAČNÍ PROSTŘEDÍ

6.1 Analýza problému a požadavků na simulační prostředí

V dnešní době je k dispozici velké množství aplikací, které umožňují pracovat s fuzzy logikou. Některé z nich se specializují pouze na tuto problematiku jiné bývají komplexnější a umožňují kupříkladu využívat fuzzy modelování v rámci nadstavbového toolboxu v rozsáhlých systémech. Dále je možné tyto aplikace také dělit na komerční produkty, či produkty volně šiřitelné, často vytvářené na akademické půdě. Mezi komerční programy je možné zařadit kupříkladu Matlab s toolboxem Fuzzy logic, či fuzzy knihovnu v aplikaci ECANSE. Dále pak specializované programy, mezi které patří mimo jiné i Fuzzy Control Manager, Fuzzy Control++ nebo FuzzyTECH. Z volně šiřitelných je možno uvést například Xfuzzy 3.0, z akademických produktů kupříkladu LFLC 2000 Ostravské univerzity [32].

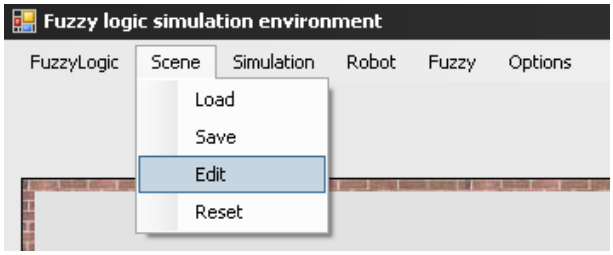
Tvorba simulačního prostředí tvoří významnou část diplomové práce. Toto prostředí by mělo umožňovat otestování navrženého řídicího systému robotu a zároveň by bylo vhodné, aby umožnilo také snadnou konfiguraci daného systému a to především fuzzy oblasti. Tím je myšlena především schopnost ladit fuzzy množiny, či efektivně spravovat a editovat fuzzy pravidla. Požadavky na aplikaci jsou tedy následující:

- možnost editace prostředí robotu (tvorba překážek)
- simulace chování robotu s reálnými parametry
- práce s fuzzy logikou (editace fuzzy množin a pravidel)
- možnost analýzy výsledků prostřednictvím grafů
- schopnost zaznamenání průběhu simulace

K vývoji aplikace je využito platformy Microsoft .NET. Tato platforma využívá princip řízeného běhového prostředí což znamená, že zdrojové kódy jsou převedeny běhovým prostředím do strojového kódu až na cílové platformě. Což řeší kupříkladu problém nepřenositelnosti aplikací mezi jednotlivými platformami jako je to například u aplikací vytvořených v jazyce Delphi či Visual Basic, kde je aplikace zkompilována přímo pro danou platformu. Jako programovací jazyk byl zvolen C#, který plnohodnotně podporuje objektově orientované programování, které bylo následně výhradně využíváno. Mezi nejvýznamnější objekty patří především objekty reprezentující scénu, robot a mechanismy fuzzy logiky. K vývoji aplikace byla využita řada programovacích technik jako například dědičnost či polymorfismus [33].

6.2 Struktura a možnosti simulačního prostředí

Aplikace vytvořená v rámci této diplomové práce se skládá z několika významných částí, které budou popsány v této kapitole. Základní menu programu obsahuje volby pro editaci překážek, simulaci, nastavení robotu, nastavení aplikace a fuzzy nadstavbu.



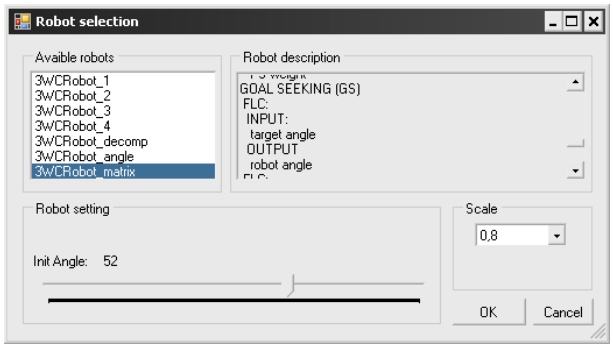
Obr. 52 Hlavní menu simulačního programu.

Neodmyslitelnou součástí každého simulačního programu pro testování navigace robotů je editor prostředí. Editor, jenž je součástí této aplikace, umožňuje vytvoření obdélníkové či kruhové

překážky. Jejich kombinacemi je poté možné navrhnout složitější tvary. Překážky se při vytvoření nacházejí v editačním módu a je umožněna změna jejich základních parametrů, případně jejich další přemístění či smazání. Ve chvíli, kdy je prostředí připraveno je k dispozici ihned k využití při simulaci, nebo je možné dané uspořádání překážek uložit pro pozdější využití do XML souboru.

Obr. 53 Menu editace překážek.

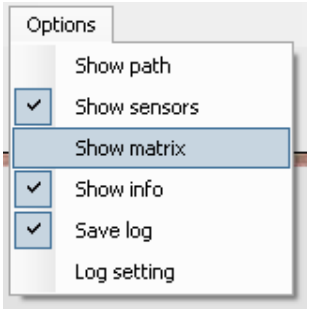
Před samotnou simulací je třeba nejprve umístit robot a cílovou pozici do prostředí. K dispozici je několik variant robotů, které se liší ve využívaném systému řízení. V dialogovém okně pro výběr robotu jsou jednotlivé typy popsány a je také umožněno nastavení počátečního natočení robotu. Další možností je změna měřítka robotu, což je důležité především v případech, kdy je testováno chování robotu v rozsáhlých prostředích.



Obr. 54 Menu výběru robotu a jeho vlastností.

Ve chvíli, kdy je umístěn robot a cílová poloha, je možné přistoupit k samotné simulaci. Nejprve je třeba z hlavního menu spustit simulační panel, kde se nacházejí tři základní tlačítka, a to tlačítko pro spuštění simulace, zastavení simulace a krokování simulace. Dále se zobrazí informační panel, který obsahuje všechny základní informace. Mezi tyto informace patří především aktuální rychlost robotu, natočení cílové polohy vzhledem k robotu, údaje ze senzorů a dále také čas, který uběhl od začátku simulace.

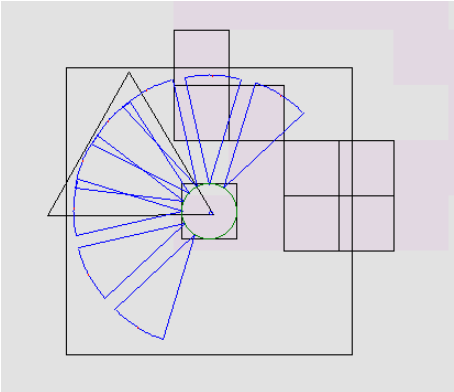
Program také nabízí možnost modifikace některých parametrů. V nastavení jsou volby, kterými lze například ovlivnit, zda bude vykreslena trajektorie robotu. Je možné skrýt či zobrazit senzory robotu. Zajímavou volbou je vykreslování údajů z matice, které umožňuje zobrazit buňky, na jejichž základě jsou odvozovány sektorové koeficienty.



Obr. 55 Nastavení aplikace.

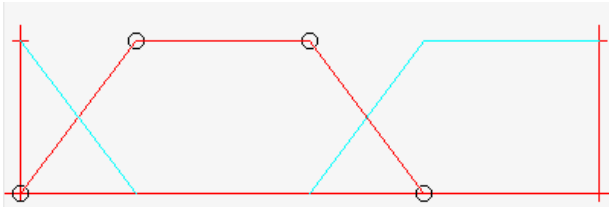
Za účelem ladění programu a především procesů spojených s fuzzy logikou byl navrhnut jednoduchý logovací systém, který umožňuje výpis nejrůznějších údajů získaných při běhu programu do textového souboru, kde je následně možná jejich analýza. Tyto informace byly cenné především při

návrhu programovacích technik, které implementují fuzzy logiku. Bylo pak následně možné zkontrolovat průběh fuzzifikace, inference či defuzzifikace. Dále byl tento logovací systém využit při hledání chyb a k ladění výkonu aplikace.



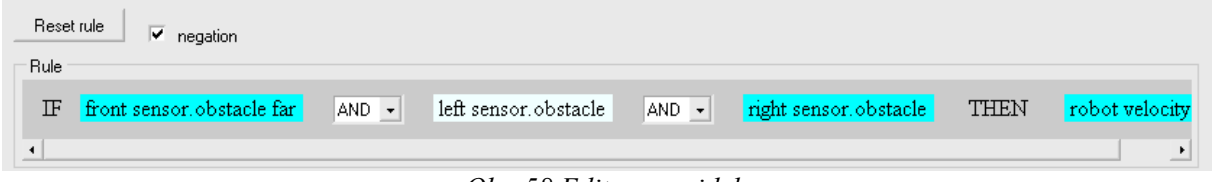
Obr. 56 Náhled na robot při spuštěném vykreslování matice.

Jedním z významných prvků vytvořené aplikace je fuzzy nadstavba. Tato nadstavba je vytvořena za účelem snadné editace parametrů spojených s fuzzy logikou. Navržené systémy řízení disponují konfiguračními xml soubory, které je poté možné načíst v tomto fuzzy editoru. Načtením souborů jsou získány vstupní a výstupní slovní proměnné, které robot využívá v daném systému řízení. Program poté umožňuje navrhnout pro konkrétní slovní proměnnou její hodnoty ve formě fuzzy množin. K dispozici je 5 základních, často využívaných typů fuzzy množin, jejichž parametry je poté možno upravovat v grafickém editoru.



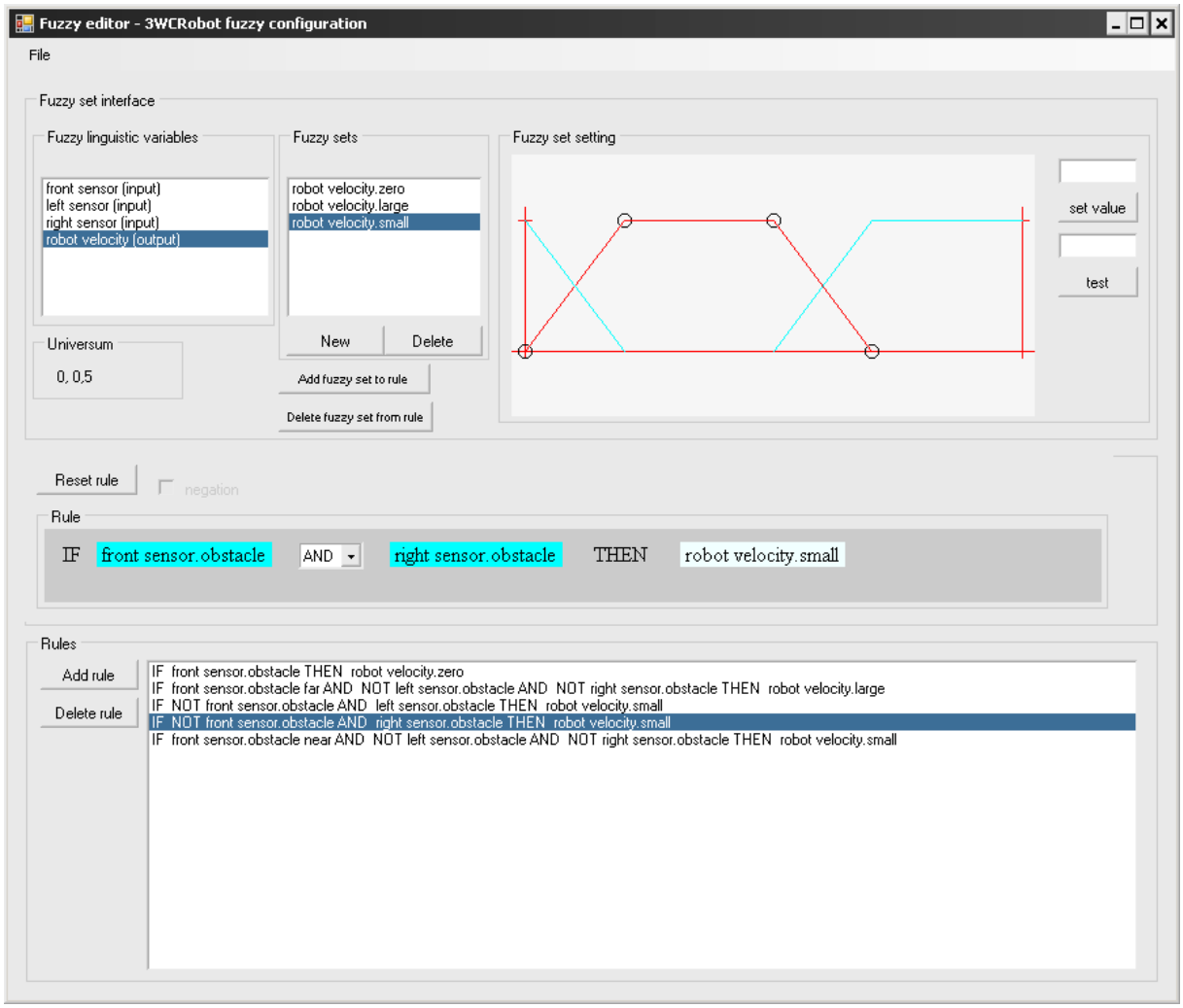
Obr. 57 Grafický editor parametrů fuzzy množin.

Ve chvíli, kdy je ukončen návrh fuzzy množin, je možné přikročit k návrhu pravidel. Pravidlo je tvořeno ve speciální části nadstavby tím, že do podmínkové části jsou vkládány vstupní proměnné a do důsledkové výstupní proměnné. Následně je umožněna volba logických spojek mezi jednotlivými fuzzy množinami, či negace fuzzy množin. Nakonec je pravidlo přesunuto do seznamu, kde jsou seskupena všechna navržená pravidla.



Obr. 58 Editor pravidel.

Tímto procesem jsou vytvořeny fuzzy množiny a navržena pravidla fuzzy regulátoru, který je následně využíván v systému řízení robotu. K bezchybnému chodu je třeba vše uložit do xml souboru, jehož jméno je pevně definováno.



Obr. 59 Celkový pohled na fuzzy nadstavbu programu.

7 OVĚŘOVACÍ EXPERIMENTY

Závěrečná kapitola je určena k porovnání jednotlivých navržených řídicích systémů. Experimentům budou podrobeny následující čtyři návrhy systémů:

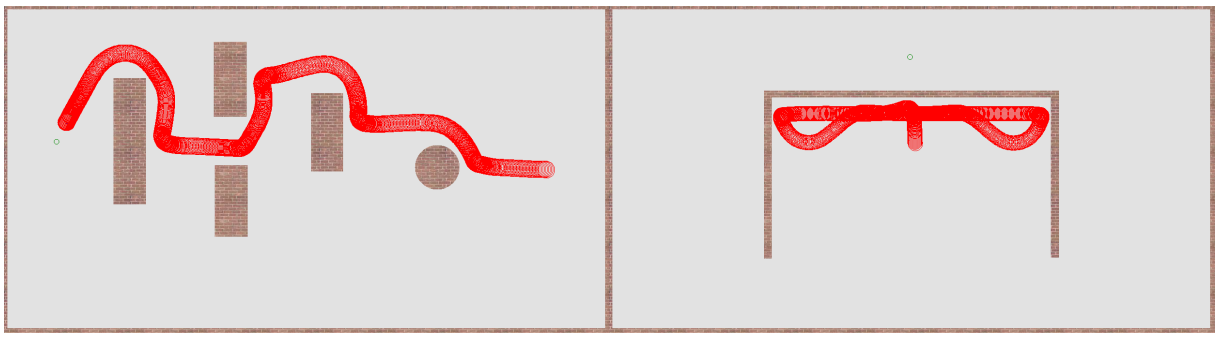
1. první z navrhovaných řídicích systémů využívající 2 regulátory
2. dekompoziční přístup (OA + GS)
3. dekompoziční přístup využívající virtuální cíle (OA + GS)
4. dekompoziční přístup využívající matici (OA + GS + PS)

Při hodnocení výsledků jednotlivých návrhu bude hodnoceno především, zdali je robot využívající daný systém schopen přesunout se do cílové polohy. V případě úspěchu jsou pak přístupy hodnoceny dle dalších parametrů, mezi které patří především časová náročnost a trajektorie, po které se robot dostal do cílové polohy.

Aby bylo možné jednotlivé metody porovnat, je třeba navrhnout mapy prostředí, které budou následně při experimentech využity. Vzhledem k tomu, že ne všechny řídicí systémy si dokáží poradit s konkávními překážkami, budou vytvořeny nejprve mapy obsahující pouze konvexní překážky. Komplexnější přístupy jsou pak dále testovány ve složitějších prostředích, kde budou také prezentovány nedostatky reaktivních řídicích systémů.

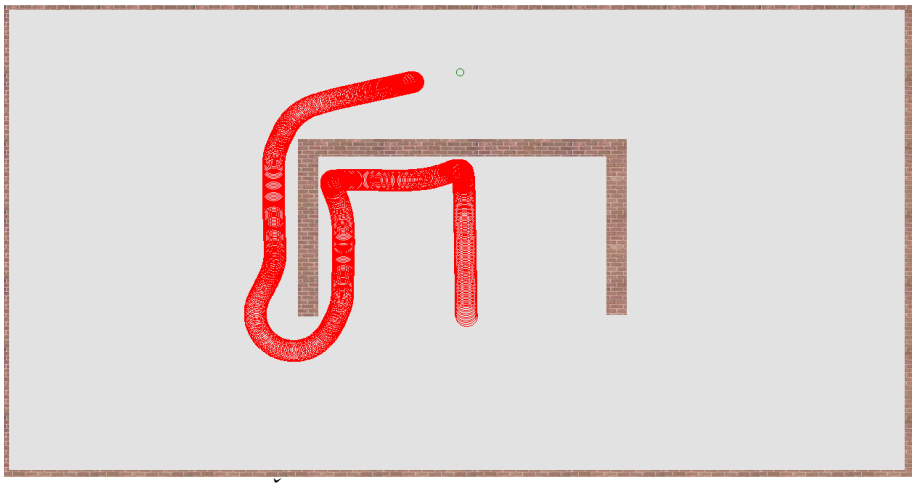
7.1 Vyhodnocení navržených přístupů

První dva navržené systémy si dokáží poradit v prostředí s konvexními překážkami. Pokud se v prostředí vyskytnou konkávní překážky, tyto metody selhávají. Situace je názorná z následujícího typického případu.



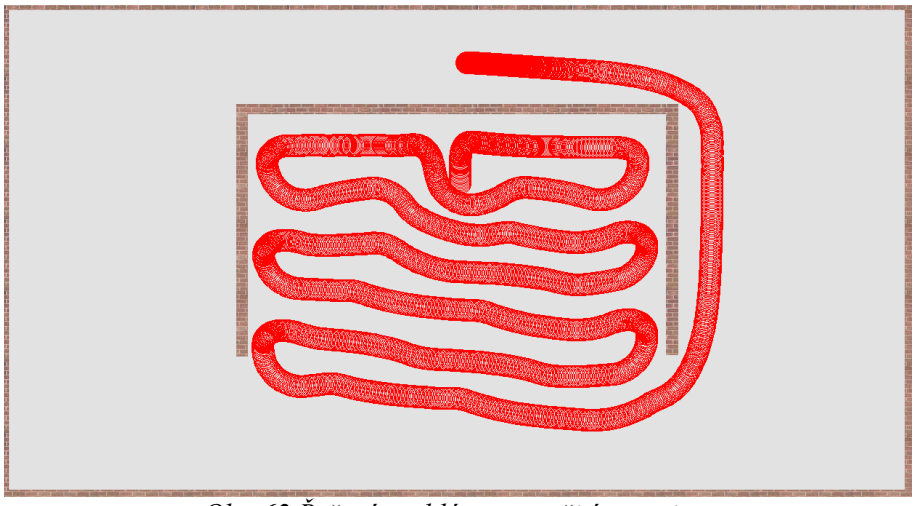
Obr. 60 Robot v prostředí s konvexními překážkami (vlevo) s konkávní překážkou (vpravo).

Dle předpokladu je nutné k řešení toto případu použít komplexnější přístupy. Následuje tedy ukázka druhého a třetího přístupu, které se s touto konkávní překážkou dokáží již vypořádat.



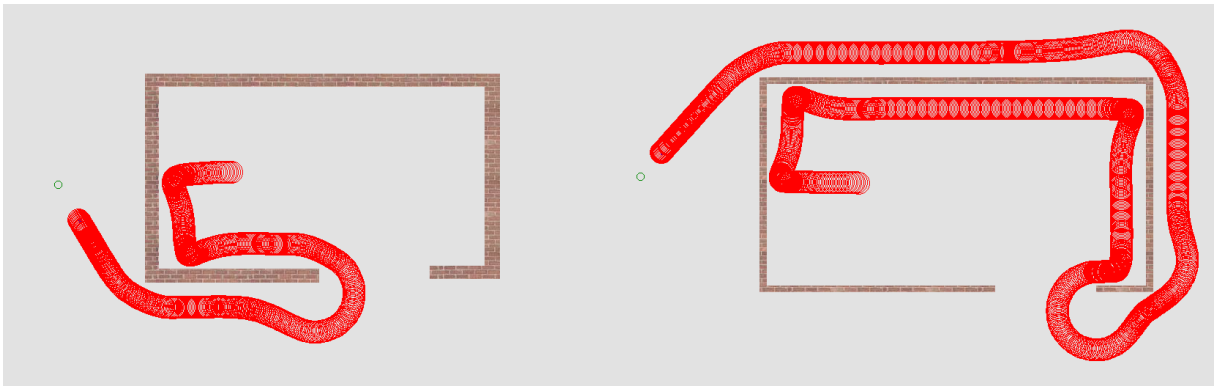
Obr. 61 Řešení problému pomocí virtuálních cílů

Přístup využívající virtuální cíle se jeví v tomto případě jako velice účinný. Robot se dostane k cíli v podstatě po nejkratší možné trajektorii. Naproti tomu řídicí systém využívající matici postupně prohledává okolí a až ve chvíli, kdy je vnitřní část konkávní překážky zcela prozkoumána vydává se robot směrem k cíli.

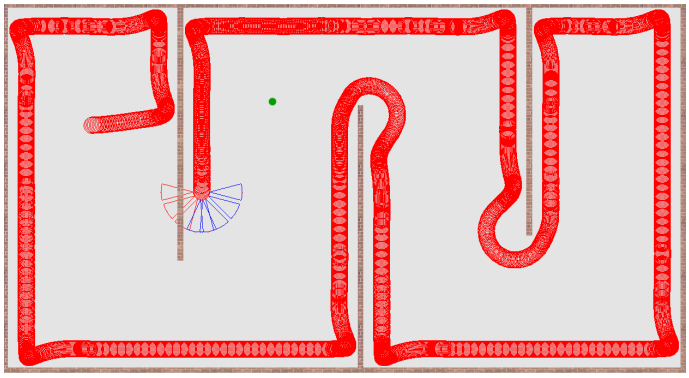


Obr. 62 Řešení problému s využitím matice.

Obdobně se metoda virtuálních cílů, jak je zřejmé z následujících obrázků, chová i ve složitějších případech obdobného charakteru. Robot ve všech případech sleduje stěnu a snaží se vykompenzovat úhel. K tomu, aby byla tato metoda účinná, je třeba, aby konkávní překážka, na kterou robot narazí byla volně v prostoru. Pokud je takováto překážka umístěna uvnitř jiné a je s ní v kontaktu, může nastat situace, že robot začne sledovat stěnu na které nebude mít možnost vykompenzovat úhel, což znemožní opětovné přepnutí na skutečný cíl. Úspěch a efektivita metody záleží tedy velkou měrou na počátečním směru objíždění překážky, jenž robot zvolí. K odstranění popsaného problému by bylo nutné navrhnout komplexnější mechanismus přepínání cílů. Takový mechanismus by mohl být založen na porovnávání pozic robotu. Robot by udržoval v paměti pozici, ve které bylo přepnuto na virtuální cíl a následně by porovnáváním s aktuální pozicí rozhodoval, zdali je ještě vhodné využívat virtuální cíl. Navržení takovéhoto mechanismu, který by poskytoval uspokojivé výsledky v jakémkoliv prostředí, je obtížný proces.



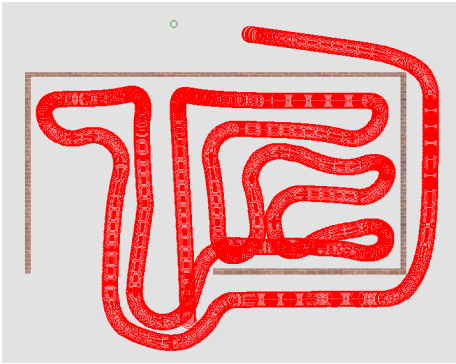
Obr. 63 Rozdíl v trajektorii robotu v závislosti na volbě směru vyhýbání (metoda virtuálních cílů).



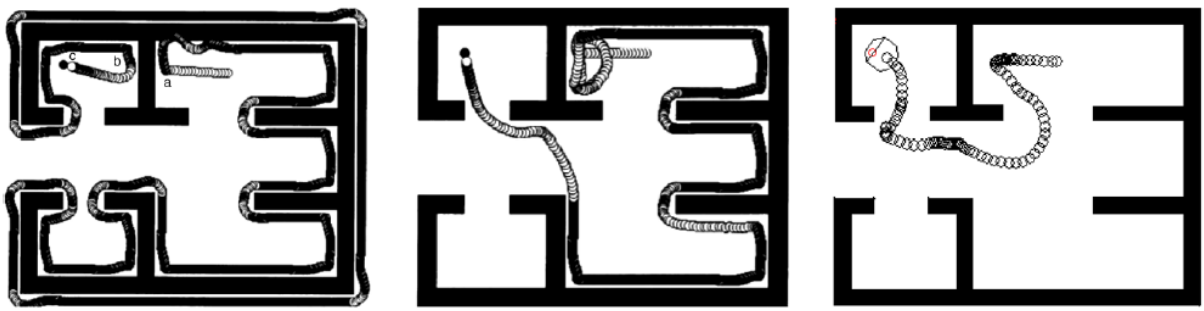
Obr. 64 Případ situace selhání metody virtuálních cílů.

Z uvedených obrázků je tedy zřejmé, že metoda virtuálních cílů v testované verzi poskytuje zaručené výsledky pouze v případě volných konkávních překážek v otevřeném prostoru. Její nasazení ve vnitřních prostorech s konkávními oblastmi je bez patřičné úpravy mechanismu přepínání cílů prakticky nemožné.

Naproti tomu přístup, jenž využívá matici se jeví ve většině případů jako spolehlivý. Důležitým faktorem tak je především fakt, zdali je robot schopen dostat se z oblastí, ze kterých si vzhledem ke svému předchozímu pohybu uzavřel cestu. Z testů, jež byly provedeny, je patrné, že si v takových situacích robot dokáže poradit. Při návrhu řídicího systému, založeného na matici byl testován i přístup využívající virtuální překážky. Obsazené pozice v matici představovaly překážky, které poté robot detekoval obdobným způsobem jako reálné. V takovém případě se ovšem vyskytly situace, kdy robot uvíznul. Aktuální řídicí systém využívá sektorové koeficienty. Ve spojení s odpovídající volbou vah pro jednotlivá chování je robotu umožněno vydat se do oblasti ve které již byl.

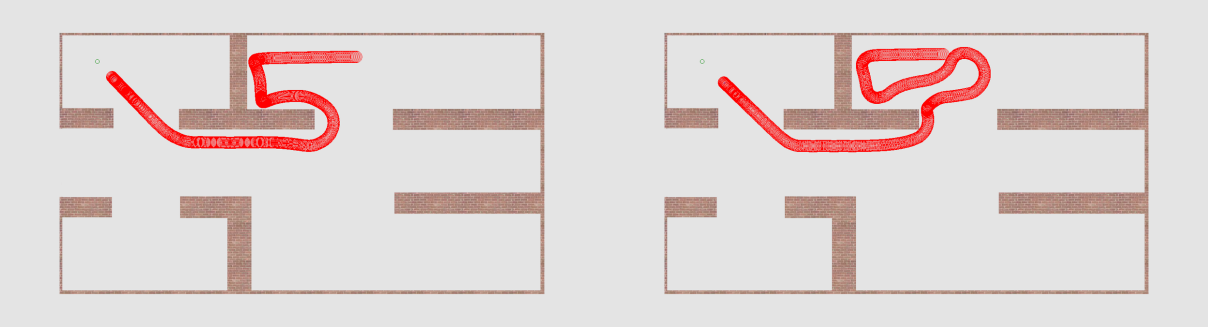


Obr. 65 Robot s řídicím systémem využívající matici v případě, kdy si uzavřel cestu.



Obr. 66 *Porovnání metod prezentované v práci [9]. a) Huang and Lee's [30] b) Krishna and Kalra's [31] c) Wang [9].*

Následný experiment je proveden na mapě, která je navržena dle experimentu uskutečněného v práci [9], kde jsou v uvedeném prostředí testovány tři metody. Tyto metody byly krátce popsány v teoretické části, kde byly prezentovány přístupy používané při reaktivní navigaci. Výsledky navržených řídicích systémů jsou v rámci této mapy obdobné. V případě metody virtuálních cílů je podstatné, že robot zvolil při prvním kontaktu z překážkou směr otáčení vlevo. V případě opačné volby by robot cílové polohy nedosáhl.



Obr. 67 Výsledky navržených systémů. Metoda virtuálních cílů (vlevo). Metoda využívající matici (vpravo).

Dále je uvedena tabulka, v níž jsou popsány nejdůležitější vlastnosti navržených metod, a to především do jakého prostředí jsou vhodné a pak také jejich výhody a nevýhody. Tato část práce porovnávala jednotlivé přístupy z globálního hlediska. V následující kapitole budou jednotlivé přístupy podrobeny testům především z hlediska kvality nalezené cesty.

Název přístupu	**prostředí s konvexními překážkami**	**prostředí s konkávními překážkami**	**výhody**	**nevýhody**
základní přístup využívající 2 regulátory	ano	ne	intuitivní návrh pravidel, menší počet regulátorů	velké množství pravidel
dekompozice	ano	ne	možnost specializace na dané chování (např. při ladění)	nutnost vyladit spolupráci chování
dekompozice + virtuální cíle	ano	částečně	v některých případech je trajektorie robotu kratší než v případě maticové metody	úspěch metody závisí na složitosti prostředí
dekompozice + matice	ano	ano	schopnost nalézt cestu i ve složitém prostředí	paměťové nároky, trajektorie robotu není optimální, velké množství regulátorů a obtížné vyladění parametrů

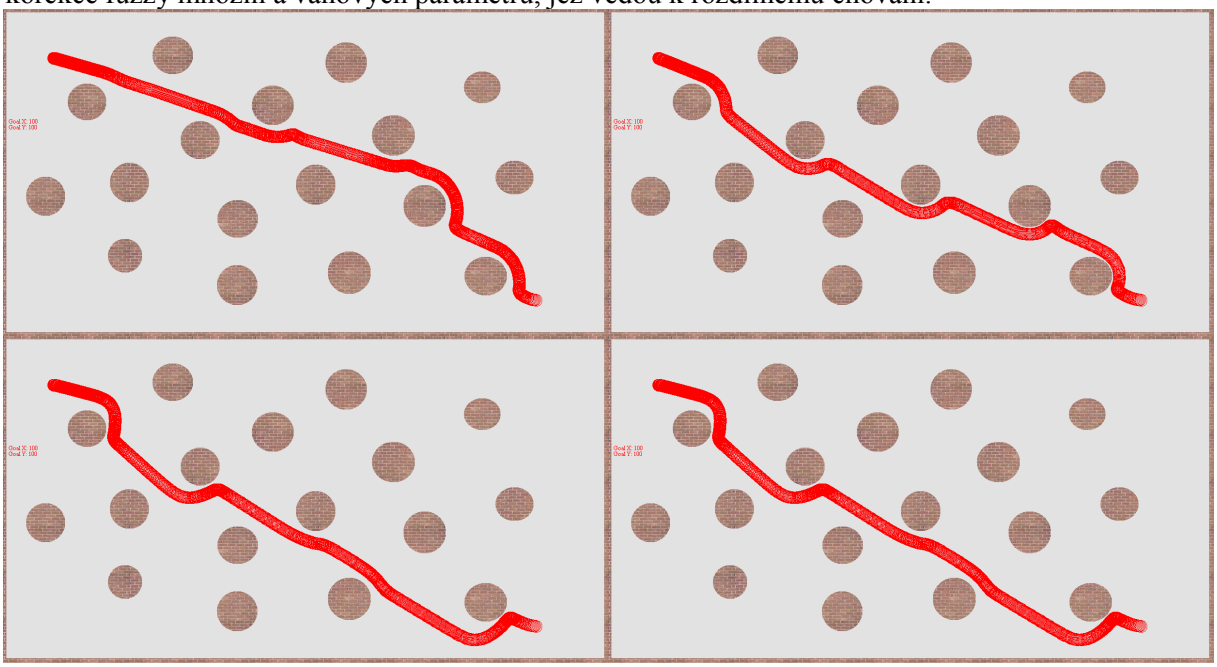
Tab. 1 *Porovnání navržených systémů řízení.*

7.2 Kvalitativní porovnání výsledků navržených systémů řízení

K tomu, aby bylo možné jednotlivé čtyři typy systémů řízení porovnat, je třeba stanovit parametry, na jejichž základě bude vyhodnocení provedeno. Základním parametrem je zajisté čas, který je potřeba k přemístění robotu z počáteční do cílové polohy. Dále je možné porovnávat vzdálenost, již robot urazil a také průměrnou rychlost, se kterou se pohyboval. Pro výkonové testování byly navrhnuty dvě mapy s konvexními překážkami a tři s výskytem konkávních překážek. V případě prostředí s konvexními překážkami je test proveden pro více počátečních a cílových pozic robotu.

1. testovací prostředí (konvexní)

Prostředí je tvořeno kruhovými náhodně rozmístěnými překážkami. Z obrázků je patrná především zkutečnost, že přístup využívající virtuální cíle je založen na dekompozičním přístupu, tudíž chování robotu je naprosto stejné. Dále bude tedy v prostředí konvexních překážek testována pouze základní dekompoziční varianta. Řídicí systém založený na využívání hodnot z matice je také postaven na dekompozici, ovšem v tomto případě byly pro zakomponování třetího chování provedeny korekce fuzzy množin a váhových parametrů, jež vedou k rozdílnému chování.



Obr. 68 První testované prostředí s konvexními překážkami. Shodné výsledky dekompozičních metod (dole). Základní přístup(vlevo nahoře). Přístup s maticí (vpravo nahoře).

Z porovnání je patrné, že základní přístup volí směr objíždění převážně vpravo. To je dáno pravidly, jež v situacích, kdy není směr dán jednoznačně senzory, volí směr vpravo. Naproti tomu u ostatních metod, které kombinují chování, je směr objíždění určen chováním GS. V tomto prostředí bylo dále provedeno testování pro další dvě varianty počáteční a cílové polohy robotu. Výsledky experimentů jsou patrné z tabulky.

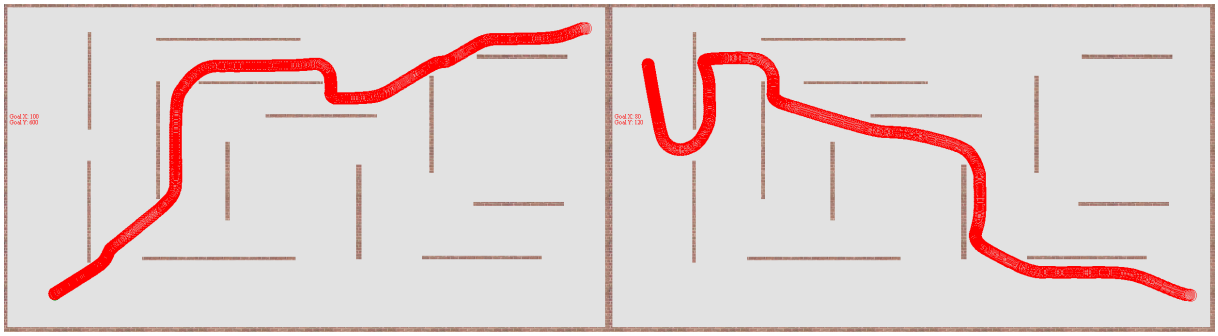
Řídicí systém	Varianta prostředí	Čas [s]	Délka dráhy [m]	Průměrná rychlost [m/s]
základní řídicí systém s dvěma regulátory	1.	107,2	24,06	0,22
	2.	95,5	23,18	0,24
	3.	112,4	26,92	0,24
řídicí systém založený na dekompozici (dvě chování)	1.	99,1	24,55	0,25
	2.	94,7	23,9	0,25
	3.	107,1	26,47	0,25
řídicí systém založený na dekompozici (tři chování) + matice	1.	92,3	24,44	0,26
	2.	82,9	23,12	0,28
	3.	87,7	26,92	0,31

Tab. 2 *Výkonnostní porovnání řídicích systémů v 1. typu prostředí.*

Hodnoty uvedené v tabulce vypovídají o skutečnosti, že řídicí systém využívající matici vykazuje již v prostředí s konvexními překážkami nejlepších výsledků. Výhody, které plynou z využití matice, nejsou v tomto prostředí uplatnitelné. Avšak změny, uskutečněné za účelem sloučení jednotlivých chování, vedou především ke zvýšení rychlosti robotu, jež zásadně ovlivňuje i ostatní testované parametry.

2. testovací prostředí (konvexní)

Prostředí je tvořeno konvexními obdélníkovými překážkami. Test je v tomto případě proveden pro dvě varianty počáteční a cílové polohy robotu.



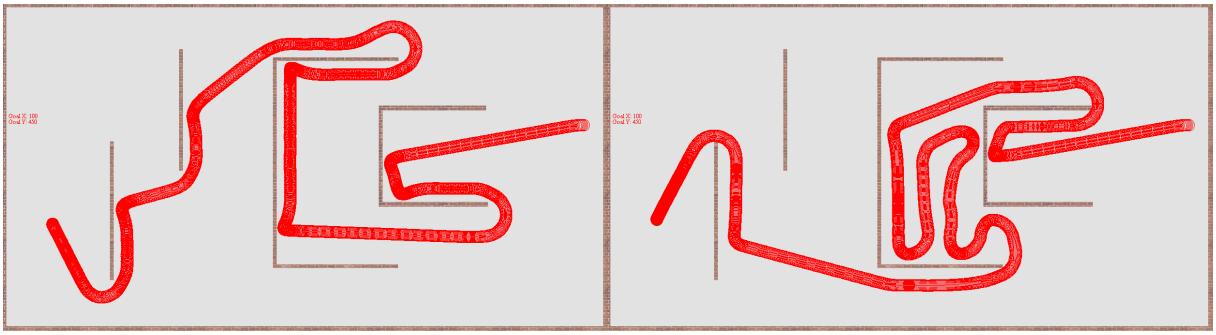
Obr. 69 Základní řídicí systém v první variantě prostředí (vlevo). Dekompoziční přístup ve druhé variantě prostředí (vpravo).

Řídicí systém	Varianta prostředí	Čas [s]	Délka dráhy [m]	Průměrná rychlost [m/s]
základní řídicí systém s dvěma regulátory	1.	117,8	28,8	0,24
	2.	125,9	29,56	0,24
řídicí systém založený na dekompozici (dvě chování)	1.	132,7	33,96	0,26
	2.	105,1	25,82	0,25
řídicí systém založený na dekompozici (tři chování) + matice	1.	105,9	29,59	0,28
	2.	99,2	29,02	0,29

Tab. 3 *Výkonnostní porovnání řídicích systémů v 2. typu prostředí.*

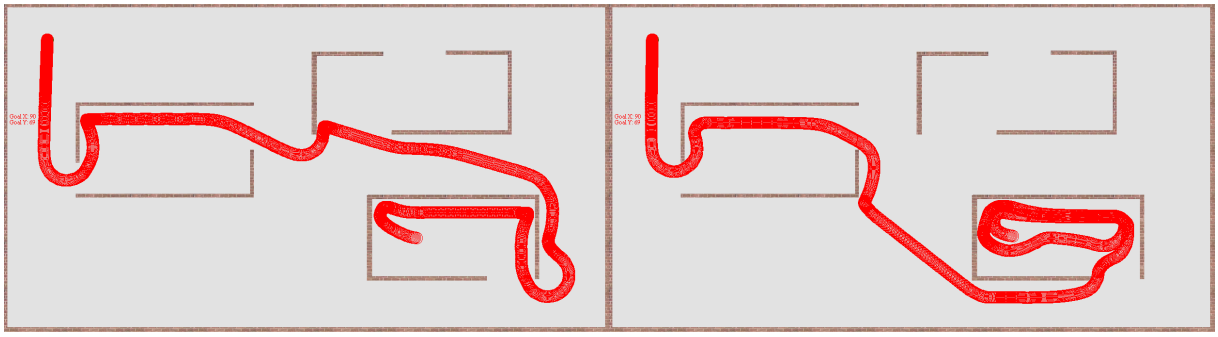
3. testovací prostředí (konkávní)

Ve zbylé části jsou již prezentovány výsledky dosažené při experimentech v prostředí, kde se vyskytují konkávní překážky. Následně je tedy test zúžen pouze na přístup využívající virtuální cíle a přístup využívající matici. Základní přístupy v tomto prostředí již selhávají a jsou uvězněny v lokálním minimu.



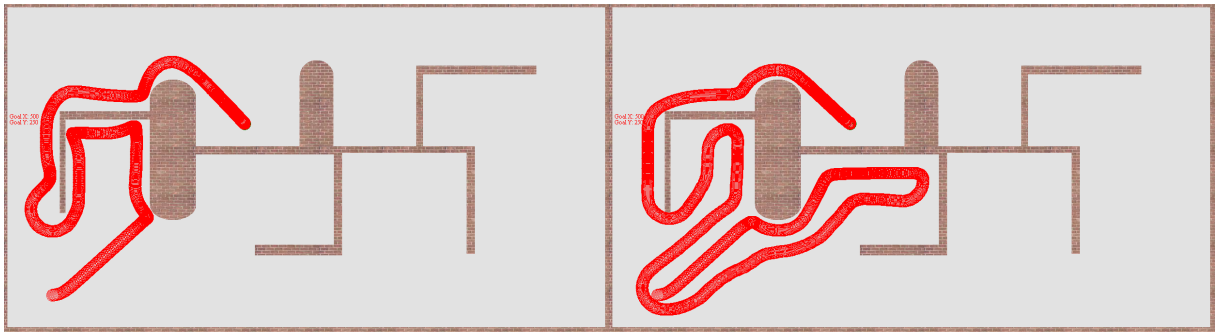
Obr. 70 Metoda využívající virtuální cíle (vlevo). Metoda využívající matici (vpravo).

4. testovací prostředí (konkávní)



Obr. 71 Metoda využívající virtuální cíle (vlevo). Metoda využívající matici (vpravo).

5. testovací prostředí (konkávní)



Obr. 72 Metoda využívající virtuální cíle (vlevo). Metoda využívající matici (vpravo).

Řídicí systém	Testovací prostředí	Čas [s]	Délka dráhy [m]	Průměrná rychlost [m/s]
řídicí systém založený na dekompozici (dvě chování) využívající virtuální cíle	3.	220,6	61,5	0,28
	4.	184,2	47,47	0,26
	5.	130,7	32,19	0,25
řídicí systém založený na dekompozici (tři chování) + matice	3.	199,7	66,49	0,33
	4.	174,3	52,66	0,30
	5.	163	55,24	0,34

Tab. 4 *Výkonnostní porovnání řídicích systémů v prostředí s konkávními překážkami.*

8 ZÁVĚR

Úvod diplomové práce je věnován popisu některých významných metod navigace mobilních robotů. Tato oblast robotiky se neustále zdokonaluje. Jednotlivé přístupy je možné rozdělit do několika směrů, přičemž některé z nich jsou s úspěchem využívány již desítky let.

V práci je využito fuzzy logiky, pomocí které je možné řešit problémy, jež je velice těžké či nemožné matematicky popsat. Tyto problémy ovšem dokáže řešit expert na danou oblast. A právě zkušenosti experta jsou ve formě pravidel využity. Další nespornou výhodou fuzzy logiky je schopnost pracovat i na základě nepřesných informací. Tím se jeví jako velice výhodná právě pro použití v robotice, kde je třeba, aby byl robot schopen reagovat i na základě možných nedokonalých údajů ze senzorického systému.

K návrhu systému řízení robotu je využito několika přístupů. Je to jednak čistě reaktivní systém, jenž nedisponuje informacemi o prostředí robotu z předchozích kroků řízení. Dále pak to jsou kombinované řídicí systémy, jež jsou již schopné tyto informace nějakým způsobem uchovávat a následně využívat. Také je využit přístup, který rozděluje chování robotu dle jednotlivých situací, což vede především ke snadnějšímu návrhu pravidel.

Významnou částí diplomové práce byla jednoznačně tvorba aplikace, jejíž účelem je testování jednotlivých systémů řízení robotu. Tato aplikace umožňuje efektivní správu fuzzy množin a fuzzy pravidel, což vede ke snadnějšímu ladění a ověřování testovaných systémů.

Z výsledků provedených experimentů je zřejmé, že reaktivní systémy řízení je možné použít samostatně pouze v prostředí, kde se vyskytují jen konvexní překážky. V případě konkávních překážek je veliké riziko, že robot uvízne v lokálním minimu. Pokud robot má schopnost uchovávat informace o prostředí, zvyšuje se jeho šance na úspěšné řešení těchto problémů. Není ovšem zaručené, že pokud si robot dokáže poradit s jedním typem konkávní překážky, dokáže pak řešit všechny situace stejně úspěšně. Prostředí robotu může být velice různorodé a proto je velice obtížné navrhnout systém řízení, který by dokázal reagovat na všechny nástrahy, které mohou robot potkat.

Aby byly patrné jednotlivé problémy, byla snaha navrhnout systémy řízení postupně od základních po komplexnější. Tento cíl se podařilo splnit a z experimentů jsou zřejmé pokroky, které přinášejí vylepšení jednotlivých přístupů. V této problematice existuje ovšem ještě velký prostor pro vylepšení. Ovšem pokud robot nebude mít kompletní informace o prostředí, je zcela nemožné dosáhnout vždy ve všech možných prostředích optimální trajektorie z počáteční do cílové polohy.

SEZNAM POUŽITÉ LITERATURY

[1] NOVÁK, V.:*Základy fuzzy modelování.* Praha: Ben - technická literatura, 2003. ISBN 80-7300-009-1.

[2] ŠVARC, I.; ŠEDA, M.; VÍTEČKOVÁ, M.: *Automatické řízení.* Brno: CERM, 2007. ISBN 978-80-214-3491-2.

[3] MODRLÁK, O.: *Fuzzy řízení a regulace* [PDF dokument].Dostupné z: http://www.fm.vslib.cz/~krtsub/fm/modrlak/pdf/tar2_fuz.pdf .

[4] NAVARA, M.; OLŠÁK, P.: *Základy fuzzy množin* [PDF dokument]. Dostupné z: http://ftp://math.feld.cvut.cz/pub/olsak/fuzzy/fuzzy.pdf.

[5] ŠOLC, F.; ŽALUD, L.: *Robotika* [PDF dokument].Dostupné z: http://matescb.skvorsmalt.cz/robotika.../VUT_Brno_Robotika.pdf .

[6] BŘEZINA, T.:*Řízení mechatronických soustav* [PDF dokument]. Dostupné z: http://autnt.fme.vutbr.cz/brezina/Rizeni_Mech_Soustav.pdf .

[7] PEŇÁZOVÁ, M.: *Když se řekne robot.* Automatizace. [online]. 2008, ročník 7 - 8. 488 - 493. [cit. 2010-6-24] Dostupné z: http://www.automatizace.cz/article.php?a=2194.

[8] MASEHIAN, E.; SEDIGHIZADEH, D.: *Classic and heuristic approaches in robot motion planning - A chronological review* [PDF dokument]. Dostupné z: http://www.waset.org/journals/waset/v29/v29-19.pdf .

[9] WANG, M.; LIU, J.: *Fuzzy logic-based real-time robot navigation in unknown environment with dead ends.* Robotics and Autonomous Systems 56 (2008) 625–643.

[10] MOTLAGH, O.; HONG, T.; ISMAIL, N.: *Development of a new minimum avoidance system for a behavior-based mobile robot.* Fuzzy Sets and Systems 160 (2009) 1929–1946.

[11] CHATTERJEE, R.; MATSUNO, F.: *Use of single side reflex for autonomous navigation of mobile robots in unknown environments*. Robotics and Autonomous Systems 35 (2001) 77–96.

[12] LUH, G.; LIU, W.: *An immunological approach to mobile robot reactive navigation.* Applied Soft Computing 8 (2008) 30–45.

[13] RAM, A.; SANTAMARÍA, J.: *Continuous case-based reasoning.* Artifical Intelligence 90 (1997) 25-77.

[14] HUI, N.; PRATIHAR, D.: *A comparative study on some navigation schemes of a real robot tackling moving obstacles.* Robotics and Computer-Integrated Manufacturing 25 (2009) 810-828.

[15] ŠTER, B.: *An integrated learning approach to environment modelling in mobile robot navigation.* Neurocomputing 57 (2004) 215-238.

[16] DUBRAWSKI, A.; CROWLEY, J.; *Learning locomotion reflexes: A self-supervised neural system for a mobile robot.* Robotics and Autonomous Systems 12 (1994) 133-142.

[17] MAAREF, F.; BARRET, C.; *Sensor-based fuzzy navigation of an autonomous mobile robot in an indoor environment.* Control Engineering Practise 8 (2000) 757-768.

[18] KRČEK, P.: *Využití expertního systému pro plánování dráhy robota*. Diplomová práce. Brno: FSI VUT Brno, Ústav automatizace a informatiky, 2003. Vedoucí diplomové práce RNDr. Jiří Dvořák, CSc.

[19] ŠÍMA, M.: *Plánování cesty robota ve spojitém prostředí*. Diplomová práce. Brno: FSI VUT Brno, Ústav automatizace a informatiky, 2006. Vedoucí diplomové práce RNDr. Jiří Dvořák, CSc.

[20] GROSS, T.: *Plánování cesty mobilního robota*. Diplomová práce. Brno: FSI VUT Brno, Ústav automatizace a informatiky, 2007. Vedoucí diplomové práce RNDr. Jiří Dvořák, CSc.

[21] SEDLÁK, V.: *Plánování cesty mobilního robota pomocí mravenčích systémů*. Bakalářská práce. Brno: FSI VUT Brno, Ústav automatizace a informatiky, 2009. Vedoucí diplomové práce RNDr. Jiří Dvořák, CSc.

[22] NEUWIRTH, D.: *Umělé imunitní výpočetní systémy.* Bakalářská práce. Brno: FIT VUT Brno, Ústav počítačových systémů, 2007. Vedoucí diplomové práce Doc. Ing. Lukáš Sekanina, Ph.D.

[23] WINKLER, Z.: *Plánování na mřížce (Robotika > Průvodce)* [online]. 2003, 12. 3. 2003 [cit. 2010-6-24]. Dostupné z WWW: <http://robotika.cz/guide/gridplan/cs>.

[24] DLOUHÝ, M.: *Exaktní plánování (Robotika > Průvodce)* [online]. 2003, 12. 3. 2003 [cit. 2010-6-24]. Dostupné z WWW: <http://robotika.cz/guide/exactplan/cs>.

[25]DLOUHÝ, M.: *Pravděpodobnostní plánování (Robotika > Průvodce)* [online]. 2003, 12. 5. 2003 [cit. 2010-6-24]. Dostupné z WWW: <http://robotika.cz/guide/probplan/cs>.
[26]JURIŠICA, L.; MURÁR, R.: *Reaktívné riadenie mobilného robota.* AT&P JOURNAL 2003-05 36-38.
[27]WIKIPEDIA, otevřená encyklopedie. Poslední revise 23. 5. 2010. http://wikipedia.org.
[28]LOCHMAN, R.: *Implementace neuronové sítě ART.* Diplomová práce. Praha: ČVUT FEL, katedra počítačů, 2006. Vedoucí diplomové práce Ing. Jan Koutník Ph.D.
[29]DVOŘÁK, J.: *Expertní systémy.* Elektronický studijní materiál, VUT v Brně, FSI, 2004, http://www.zam.fme.vutbr.cz/~jdvorak/Opory/ExpertniSystemy.pdf.
[30]HUANG, H.; LEE, P.: *A real-time algorithm for obstacle avoidance of autonomous mobile robots.* Robotica 10 (1992) 217-227.
[31]KRISHNA, K.; KALRA, P.: *Perception and remembrance of the environment during real-time navigation of a mobile robot.* Robotics and Autonomous Systems 37 (2001) 25-51.
[32]JURA, P.: *Základy fuzzy logiky pro řízení a modelování.* Brno, VUTIUM, 2003, ISBN 80-214-2261-0.
[33]PUŠ, P.: *Poznáváme C# a Microsoft .NET.* [online]. [cit. 18.4.2010]. Dostupné z: http://www.zive.cz/clanky/poznavame-c-a-microsoft-net--1dil/sc-3-a-120978/default.aspx.

SEZNAM PŘÍLOH

Příloha 1. Výpis fuzzy pravidel a parametrů fuzzy množin.
Příloha 2. Disk CD-ROM s obsahem:

- elektronická podoba práce
- simulační software
- .net framework

PŘÍLOHA 1. VÝPIS FUZZY PRAVIDEL A PARAMETRŮ FUZZY MNOŽIN

Základní navržený systém řízení s dvěma regulátory.
1. regulátor (θ)
Parametry fuzzy množin:
target angle: *left [-180, -180, -20, 0], zero [-20, 0, 0, 20], right [0, 20, 180, 180]*.
front sensor, right sensor, left sensor: *obstacle [0, 0, 0.1, 0.55], obstacle near [0.1, 0.55, 0.55, 1], obstacle far [0.55, 1, 1, 1]*.
robot angle: *left [-5, -5, -0.5, 0], zero [-0.5, 0, 0, 0.5], right [0, 0.5, 5, 5]*.
Pravidla:
IF target angle is *left* AND front sensor is *obstacle far* AND left sensor is *obstacle far* AND right sensor is *obstacle far* THEN robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle far* AND left sensor is *obstacle far* AND right sensor is *obstacle near* THEN robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle far* AND left sensor is *obstacle far* AND right sensor is *obstacle* THEN robot angle is *left*
IF target angle is *left* AND front sensor is obstacle near AND left sensor is *obstacle far* AND right sensor is *obstacle far* THEN robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle near* AND left sensor is *obstacle far*AND right sensor is *obstacle near* THEN robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle near* AND left sensor is *obstacle far* AND right sensor is *obstacle* THEN robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle near* AND left sensor is *obstacle near* AND right sensor is *obstacle far* THEN robot angle is *right*
IF target angle is *left* AND front sensor is *obstacle near* AND left sensor is *obstacle near* AND right sensor is *obstacle near* THEN robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle near* AND left sensor is *obstacle near* AND right sensor is *obstacle* THEN robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle near* AND left sensor is *obstacle* AND right sensor is *obstacle far* THEN robot angle is *right*
IF target angle is *left* AND front sensor is *obstacle near* AND left sensor is *obstacle* AND right sensor is *obstacle near* THEN robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle near* AND left sensor is *obstacle* AND right sensor is *obstacle* AND robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle* AND left sensor is *obstacle far* AND right sensor is *obstacle far* THEN robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle* AND left sensor is *obstacle far* AND right sensor is *obstacle near* THEN robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle* AND left sensor is *obstacle far* AND right sensor is *obstacle* THEN robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle* AND left sensor is *obstacle near* AND right sensor is *obstacle far* THEN robot angle is *right*
IF target angle is *left* AND front sensor is *obstacle* AND left sensor is *obstacle near* AND right sensor is *obstacle near* THEN robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle* AND left sensor is *obstacle near* AND right sensor is *obstacle* THEN robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle* AND left sensor is *obstacle* AND right sensor is *obstacle far* THEN robot angle is *right*
IF target angle is *left* AND front sensor is *obstacle* AND left sensor is *obstacle* AND right sensor is *obstacle near* THEN robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle* AND left sensor is *obstacle* AND right sensor is *obstacle* THEN robot angle is *left*

IF target angle is *zero* AND front sensor is *obstacle far* AND left sensor is *obstacle far* AND right sensor is *obstacle far* THEN robot angle is *zero*
IF target angle is *zero* AND front sensor is *obstacle far* AND left sensor is *obstacle far* AND right sensor is *obstacle near* THEN robot angle is *zero*
IF target angle is *zero* AND front sensor is *obstacle far* AND left sensor is *obstacle far* AND right sensor is *obstacle* THEN robot angle is *zero*
IF target angle is *zero* AND front sensor is *obstacle far* AND left sensor is *obstacle near* AND right sensor is *obstacle far* THEN robot angle is *zero*
IF target angle is *zero* AND front sensor is *obstacle far* AND left sensor is *obstacle near* AND right sensor is *obstacle near* THEN robot angle is *zero*
IF target angle is *zero* AND front sensor is *obstacle far* AND left sensor is *obstacle near* AND right sensor is *obstacle* THEN robot angle is *zero*
IF target angle is *zero* AND front sensor is *obstacle far* AND left sensor is *obstacle* AND right sensor is *obstacle far* THEN robot angle is *zero*
IF target angle is *zero* AND front sensor is *obstacle far* AND left sensor is *obstacle* AND right sensor is *obstacle near* THEN robot angle is *zero*
IF target angle is *zero* AND front sensor is *obstacle far* AND left sensor is *obstacle* AND right sensor is *obstacle* THEN robot angle is *zero*
IF target angle is *zero* AND front sensor is *obstacle near* AND left sensor is *obstacle far* AND right sensor is *obstacle far* THEN robot angle is *right*
IF target angle is *zero* AND front sensor is *obstacle near* AND left sensor is *obstacle far* AND right sensor is *obstacle near* THEN robot *angle is left*
IF target angle is *zero* AND front sensor is *obstacle near* AND left sensor is *obstacle far* AND right sensor is *obstacle* THEN robot angle is *left*
IF target angle is *zero* AND front sensor is *obstacle near* AND left sensor is *obstacle near* AND right sensor is *obstacle far* THEN robot angle is *right*
IF target angle is *zero* AND front sensor is *obstacle near* AND left sensor is *obstacle near* AND right sensor is *obstacle near* THEN robot angle is *right*
IF target angle is *zero* AND front sensor is *obstacle near* AND left sensor is *obstacle near* AND right sensor is *obstacle* THEN robot angle is *right*
IF target angle is *zero* AND front sensor is *obstacle near* AND left sensor is *obstacle* AND right sensor is *obstacle far* THEN robot angle is *right*
IF target angle is *zero* AND front sensor is *obstacle near* AND left sensor is *obstacle* AND right sensor is *obstacle near* THEN robot angle is *right*
IF target angle is *zero* AND front sensor is *obstacle near* AND left sensor is *obstacle* AND right sensor is *obstacle* THEN robot angle is *right*
IF target angle is *zero* AND front sensor is *obstacle* AND left sensor is *obstacle far* AND right sensor is *obstacle far* THEN robot angle is *right*
IF target angle is *zero* AND front sensor is *obstacle* AND left sensor is *obstacle far* AND right sensor is *obstacle near* THEN robot angle is *left*
IF target angle is *zero* AND front sensor is *obstacle* AND left sensor is *obstacle far* AND right sensor is *obstacle* THEN robot angle is *left*
IF target angle is *zero* AND front sensor is *obstacle* AND left sensor is *obstacle near* AND right sensor is *obstacle far* THEN robot angle is *right*
IF target angle is *zero* AND front sensor is *obstacle* AND left sensor is *obstacle near* AND right sensor is *obstacle near* THEN robot angle is *right*
IF target angle is *zero* AND front sensor is *obstacle* AND left sensor is *obstacle near* AND right sensor is *obstacle* THEN robot angle is *right*
IF target angle is *zero* AND front sensor is *obstacle* AND left sensor is *obstacle* AND right sensor is *obstacle far* THEN robot angle is *right*
IF target angle is *zero* AND front sensor is *obstacle* AND left sensor is *obstacle* AND right sensor is *obstacle near* THEN robot angle is *right*
IF target angle is *zero* AND front sensor is *obstacle* AND left sensor is *obstacle* AND right sensor is *obstacle* THEN robot angle is *right*

Příloha 1. Výpis fuzzy pravidel a parametrů fuzzy množin

IF target angle is *right* AND front sensor is *obstacle far* AND left sensor is *obstacle far* AND right sensor is *obstacle far* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle far* AND left sensor is *obstacle far* AND right sensor is *obstacle near* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle far* AND left sensor is *obstacle far* AND right sensor is *obstacle* THEN robot angle is *zero*

IF target angle is *right* AND front sensor is *obstacle far* AND left sensor is *obstacle near* AND right sensor is *obstacle far* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle far* AND left sensor is *obstacle near* AND right sensor is *obstacle near* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle far* AND left sensor is *obstacle near* AND right sensor is *obstacle* THEN robot angle is *zero*

IF target angle is *right* AND front sensor is *obstacle far* AND left sensor is *obstacle* AND right sensor is *obstacle far* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle far* AND left sensor is *obstacle* AND right sensor is *obstacle near* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle far* AND left sensor is *obstacle* AND right sensor is *obstacle* THEN robot angle is *zero*

IF target angle is *right* AND front sensor is *obstacle near* AND left sensor is *obstacle far* AND right sensor is *obstacle far* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle near* AND left sensor is *obstacle far* AND right sensor is *obstacle near* THEN robot angle is *left*

IF target angle is *right* AND front sensor is *obstacle near* AND left sensor is *obstacle far* AND right sensor is *obstacle* THEN robot angle is *left*

IF target angle is *right* AND front sensor is *obstacle near* AND left sensor is *obstacle near* AND right sensor is *obstacle far* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle near* AND left sensor is *obstacle near* AND right sensor is *obstacle near* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle near* AND left sensor is *obstacle near* AND right sensor is *obstacle* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle near* AND left sensor is *obstacle* AND right sensor is *obstacle far* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle near* AND left sensor is *obstacle* AND right sensor is *obstacle near* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle near* AND left sensor is *obstacle* AND right sensor is *obstacle* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle* AND left sensor is *obstacle far* AND right sensor is *obstacle far* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle* AND left sensor is *obstacle far* AND right sensor is *obstacle near* THEN robot angle is *left*

IF target angle is *right* AND front sensor is *obstacle* AND left sensor is *obstacle far* AND right sensor is *obstacle* THEN robot angle is *left*

IF target angle is *right* AND front sensor is *obstacle* AND left sensor is *obstacle near* AND right sensor is *obstacle far* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle* AND left sensor is *obstacle near* AND right sensor is *obstacle near* THEN robot angle is *right*

IF target angle is *right* AND front sensor is obstacle AND left sensor is obstacle near AND right sensor is *obstacle* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle* AND left sensor is *obstacle* AND right sensor is *obstacle far* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle* AND left sensor is *obstacle* AND right sensor is *obstacle near* THEN robot angle is *right*

IF target angle is *right* AND front sensor is *obstacle* AND left sensor is *obstacle* AND right sensor is *obstacle* THEN robot angle is *right*

IF target angle is *left* AND front sensor is *obstacle far* AND left sensor is *obstacle near* AND right sensor is *obstacle far* THEN robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle far* AND left sensor is *obstacle near* AND right sensor is *obstacle near* THEN robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle far* AND left sensor is *obstacle near* AND right sensor is *obstacle* THEN robot angle is *left*
IF target angle is *left* AND front sensor is *obstacle far* AND left sensor is *obstacle* AND right sensor is *obstacle far* THEN robot angle is *zero*
IF target angle is *left* AND front sensor is *obstacle far* AND left sensor is *obstacle* AND right sensor is *obstacle near* THEN robot angle is *zero*
IF target angle is *left* AND front sensor is *obstacle far* AND left sensor is *obstacle* AND right sensor is *obstacle* THEN robot angle is *zero*

2.regulátor (v)
Parametry fuzzy množin:
target distance: *zero [0, 0, 0, 1], near [0, 1, 1, 2], far [1, 2, 3, 3].*
robot velocity: *zero [0, 0, 0, 0.1], small [0, 0.1, 0.25, 0.35], large [0.25, 0.35, 0.5, 0.5].*
Pravidla:
IF target distance is *zero* OR front sensor is *obstacle* THEN robot velocity is *zero*
IF target distance is *near* AND NOT front sensot is *obstacle* THEN robot velocity is *small*
IF NOT tatget distance is *zero* AND front sensor is *obstacle near* THEN robot velocity is *small*
IF NOT target distance is *zero* AND NOT front sensor is *obstacle* AND right sensor is *obstacle* THEN robot velocity is *small*
IF NOT target distance is *zero* AND NOT front sensor is *obstacle* AND left sensor is *obstacle* THEN robot velocity is *small*
IF target distance is *large* AND front sensor is *obstacle far* AND NOT left sensor is *obstacle* AND NOT right sensor is *obstacle* THEN robot velocity is *large*

Dekompozice (OA + GS)
OA - 1. regulátor (θ)
Parametry fuzzy množin:
front sensor, right sensor, left sensor: *obstacle [0, 0, 0.1, 0.55], obstacle near [0.1, 0.55, 0.55, 1], obstacle far [0.55, 1, 1, 1].*
robot angle: *left [-5, -5, -0.5, 0], zero [-0.5, 0, 0, 0.5], right [0, 0.5, 5, 5].*
Pravidla:
IF front sensor is *obstacle* AND left sensor is *obstacle* AND right sensor is *obstacle* THEN robot angle is zero
IF front sensor is *obstacle* AND left sensor is *obstacle* AND right sensor is *obstacle near* THEN robot angle is *right*
IF ront sensor is *obstacle* AND left sensor is *obstacle* AND right sensor is *obstacle far* THEN robot angle is *right*
IF front sensor is *obstacle* AND left sensor is *obstacle near* AND right sensor is *obstacle* THEN robot angle is *left*
IF front sensor is *obstacle* AND left sensor is *obstacle near* AND right sensor is *obstacle near* THEN robot angle is *zero*
IF front sensor is *obstacle* AND left sensor is *obstacle near* AND right sensor is *obstacle far* THEN robot angle is *right*
IF front sensor is *obstacle* AND left sensor is *obstacle far* AND right sensor is *obstacle* THEN robot angle is *left*
IF front sensor is *obstacle* AND left sensor is *obstacle far* AND right sensor is *obstacle near* THEN robot angle is *left*
IF front sensor is *obstacle* AND left sensor is *obstacle far* AND right sensor is *obstacle far* THEN robot angle is *zero*
IF front sensor is *obstacle near* AND left sensor is *obstacle* AND right sensor is *obstacle* THEN robot

angle is *zero*
IF front sensor is *obstacle near* AND left sensor is *obstacle* AND right sensor is *obstacle near* THEN robot angle is *right*
IF front sensor is obstacle near AND left sensor is *obstacle* AND right sensor is *obstacle far* AND robot angle is *right*
IF front sensor is *obstacle near* AND left sensor is *obstacle near* AND right sensor is *obstacle* THEN robot angle is *left*
IF front sensor is *obstacle near* AND left sensor is *obstacle near* AND right sensor is *obstacle near* THEN robot angle is *zero*
IF front sensor is *obstacle near* AND left sensor is *obstacle near* AND right sensor is *obstacle far* THEN robot angle is *right*
IF front sensor is *obstacle near* AND left sensor is *obstacle far* AND right sensor is *obstacle* THEN robot angle is *left*
IF front sensor is *obstacle near* AND left sensor is *obstacle far* AND right sensor is *obstacle near* THEN robot angle is *left*
IF front sensor is *obstacle near* AND left sensor is *obstacle far* AND right sensor is *obstacle far* THEN robot angle is *zero*
IF front sensor is *obstacle far* AND left sensor is *obstacle* AND right sensor is *obstacle* THEN robot angle is *zero*
IF front sensor is *obstacle far* AND left sensor is *obstacle* AND right sensor is *obstacle near* THEN robot angle is *zero*
IF front sensor is *obstacle far* AND left sensor is *obstacle* AND right sensor is *obstacle far* THEN robot angle is *zero*
IF front sensor is *obstacle far* AND left sensor is *obstacle near* AND right sensor is *obstacle* THEN robot angle is *zero*
IF front sensor is *obstacle far* AND left sensor is *obstacle near* AND right sensor is *obstacle near* THEN robot angle is *zero*
IF front sensor is *obstacle far* AND left sensor is *obstacle near* AND right sensor is *obstacle far* THEN robot angle is *zero*
IF front sensor is *obstacle far* AND left sensor is *obstacle far* AND right sensor is *obstacle* THEN robot angle is *zero*
IF front sensor is *obstacle far* AND left sensor is *obstacle far* AND right sensor is *obstacle near* THEN robot angle is *zero*
IF front sensor is *obstacle far* AND left sensor is *obstacle far* AND right sensor is *obstacle far* THEN robot angle is *zero*

OA - 2. regulátor (*v*)
Parametry fuzzy množin:
front sensor, right sensor, left sensor: *obstacle [0, 0, 0.1, 0.55], obstacle near [0.1, 0.55, 0.55, 1], obstacle far [0.55, 1, 1, 1].*
robot velocity: *zero [0, 0, 0, 0.1], small [0, 0.1, 0.25, 0.35], large [0.25, 0.35, 0.5, 0.5].*
Pravidla:
IF front sensor is *obstacle* THEN robot velocity is *zero*
IF NOT front sensor is *obstacle* AND right sensor is *obstacle* THEN robot velocity is *small*
IF NOT front sensor is *obstacle* AND left sensor is *obstacle* THEN robot velocity is *small*
IF front sensor is *obstacle far* AND NOT left sensor is *obstacle* AND NOT right sensor is *obstacle* THEN robot velocity is large
IF front sensor is *obstacle near* AND NOT left sensor is *obstacle* AND NOT right sensor is *obstacle* THEN robot velocity is *small*

OA - 3. regulátor (*w*)
Parametry fuzzy množin:
front sensor, right sensor, left sensor: *obstacle [0, 0, 0.1, 0.55], obstacle near [0.1, 0.55, 0.55, 1], obstacle far [0.55, 1, 1, 1].*

oa weight: *zero [0, 0, 0, 0.2], small [0, 0.2, 0.8, 1], large [0.8, 1, 1, 1].*
Pravidla:
IF front sensor is *obstacle* THEN oa weight is *zero*
IF NOT front sensor is *obstacle* AND right sensor is *obstacle* THEN oa weight is *small*
IF NOT front sensor is *obstacle* AND left sensor is *obstacle* THEN oa weight is *small*
IF front sensor is *obstacle far* AND NOT left sensor is *obstacle* AND NOT right sensor is *obstacle* THEN oa weight is *learge*
IF front sensor is *obstacle near* AND NOT left sensor is *obstacle* AND NOT right sensor is *obstacle* THEN oa weight is *small*

GS - 1. regulátor (θ)
Parametry fuzzy množin:
target angle: *left [-180, -180, -20, 0], zero [-20, 0, 0, 20], right [0, 20, 180, 180].*
robot angle: *left [-5, -5, -0.5, 0], zero [-0.5, 0, 0, 0.5], right [0, 0.5, 5, 5].*
Pravidla:
IF target angle is *left* THEN robot angle is *left*
IF target angle is *zero* THEN robot angle is *zero*
IF target angle is *right* THEN robot angle is *right*

GS - 2. regulátor (v)
Parametry fuzzy množin:
target distance: *zero [0, 0, 0, 1], near [0, 1, 1, 2], far [1, 2, 3, 3].*
robot velocity: *zero [0, 0, 0, 0.1], small [0, 0.1, 0.25, 0.35], large [0.25, 0.35, 0.5, 0.5].*
Pravidla:
IF target distance is *large* THEN robot velocity is *large*
IF target distance is *small* THEN robot velocity is *small*
IF target distance is *zero* THEN robot velocity is *zero*

Dekompozice s využitím matice (OA + GS + PS)
OA - 1. regulátor (θ)
shodné s příslušným regulátorem základní dekompozice
OA - 2. regulátor (*v*)
shodné s příslušným regulátorem základní dekompozice
OA - 3. regulátor (*w*)
Parametry fuzzy množin:
front sensor, right sensor, left sensor: *obstacle [0, 0, 0.1, 0.55], obstacle near [0.1, 0.55, 0.55, 1], obstacle far [0.55, 1, 1, 1]*
oa weight: *zero [0, 0, 0, 0.2], small [0, 0.2, 0.8, 1], large [0.8, 1, 1, 1]*
Pravidla:
IF front sensor is *zero* THEN oa weight is *zero*
IF front sensor is *small* THEN oa weight is *small*
IF front sensor is *large* THEN oa weight is *large*

GS - 1. regulátor (θ)
shodné s příslušným regulátorem základní dekompozice
GS - 2. regulátor (*v*)
shodné s příslušným regulátorem základní dekompozice
GS - 3. regulátor (*w*)
Parametry fuzzy množin:
oa weight: *zero[0, 0, 0.2, 0.2], small[0, 0.2, 0.4, 0.6], large[0.4, 0.6, 1, 1].*
ps weight: *zero[0, 0, 0.2, 0.2], small[0, 0.2, 0.4, 0.6], large[0.4, 0.6, 1, 1].*
gs weight: *zero[0, 0, 0.2, 0.2], small[0, 0.2, 0.4, 0.6], large[0.4, 0.6, 1, 1].*
Pravidla:
IF ps weight is *zero* AND oa weight is *zero* THEN gs weight is *large*

Příloha 1. Výpis fuzzy pravidel a parametrů fuzzy množin

IF NOT ps weight is *zero* OR NOT oa weight is *zero* THEN gs weight is *zero*
IF oa weight is *small* AND ps weight is *zero* THEN gs weight is *small*

PS - 1. regulátor (θ)
Parametry fuzzy množin:
front sector, left sector, right sector: *zero [0, 0, 0.1, 0.55], small [0.1, 0.55, 0.55, 1], large [0.55, 1, 1, 1].*
robot angle: *left [-5, -5, -0.5, 0], zero [-0.5, 0, 0, 0.5], right [0, 0.5, 5, 5]*
Pravidla:
IF front sector is *large* AND left sector is *large* AND right sector is *large* THEN robot angle is *zero*
IF front sector is *large* AND left sector is *large* AND right sector is *small* THEN robot angle is *right*
IF ront sector is *large* AND left sector is *large* AND right sector is *zero* THEN robot angle is *right*
IF front sector is *large* AND left sector is *small* AND right sector is *large* THEN robot angle is *left*
IF front sector is *large* AND left sector is *small* AND right sector is *small* THEN robot angle is *zero*
IF front sector is *large* AND left sector is *small* AND right sector is *zero* THEN robot angle is *right*
IF front sector is *large* AND left sector is *zero* AND right sector is *large* THEN robot angle is *left*
IF front sector is *large* AND left sector is *zero* AND right sector is *small* THEN robot angle is *left*
IF front sector is *large* AND left sector is *zero* AND right sector is *zero* THEN robot angle is *zero*
IF front sector is *small* AND left sector is *large* AND right sector is *large* THEN robot angle is *zero*
IF front sector is *small* AND left sector is *large* AND right sector is *small* THEN robot angle is *right*
IF front sector is *small* AND left sector is *large* AND right sector is *zero* AND robot angle is *right*
IF front sector is *small* AND left sector is *small* AND right sector is *large* THEN robot angle is *left*
IF front sector is *small* AND left sector is *small* AND right sector is *small* THEN robot angle is *zero*
IF front sector is *small* AND left sector is *small* AND right sector is *zero* THEN robot angle is *right*
IF front sector is *small* AND left sector is *zero* AND right sector is *large* THEN robot angle is *left*
IF front sector is *small* AND left sector is *zero* AND right sector is *small* THEN robot angle is *left*
IF front sector is *small* AND left sector is *zero* AND right sector is *zero* THEN robot angle is *zero*
IF front sector is *zero* AND left sector is *large* AND right sector is *large* THEN robot angle is *zero*
IF front sector is *zero* AND left sector is *large* AND right sector is *small* THEN robot angle is *zero*
IF front sector is *zero* AND left sector is *large* AND right sector is *zero* THEN robot angle is *zero*
IF front sector is *zero* AND left sector is *small* AND right sector is *large* THEN robot angle is *zero*
IF front sector is *zero* AND left sector is *small* AND right sector is *small* THENrobot angle is *zero*
IF front sector is *zero* AND left sector is *small* AND right sector is *zero* THEN robot angle is *zero*
IF front sector is *zero* AND left sector is *zero* AND right sector is *large* THEN robot angle is *zero*
IF front sector is *zero* AND left sector is *zero* AND right sector is *small* THEN robot angle is *zero*
IF front sector is *zero* AND left sector is *zero* AND right sector is *zero* THEN robot angle is *zero*

PS - 2. regulátor (w)
Parametry fuzzy množin:
front sector, left sector, right sector: *zero [0, 0, 0.1, 0.55], small [0.1, 0.55, 0.55, 1], large [0.55, 1, 1, 1].*
ps weight: *zero [0, 0, 0, 0.2], small [0, 0.2, 0.8, 1], large [0.8, 1, 1, 1].*
Pravidla:
IF front sector is *zero* THEN ps weight is *zero*
IF front sector is *small* THEN ps weight is *small*
IF front sector is *large* AND left sector is *large* THEN ps weight is *small*
IF front sector is *large* AND right sector is *large* THEN ps weight is *small*
IF front sector is *large* AND NOT left sector is *large* THEN ps weight is *large*
IF front sector is *large* AND NOT right sector is *large* THEN ps weight is *large*