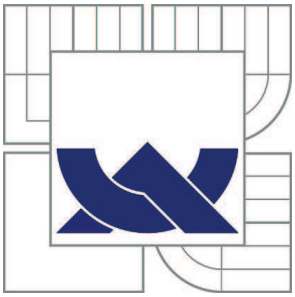
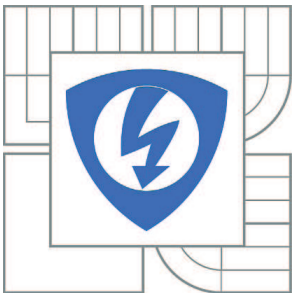


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

FUZZY MODELY MAP PRO POHYB MOBILNÍCH ROBOTŮ.

FUZZY MAP MODELS FOR MOBILE ROBOTS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

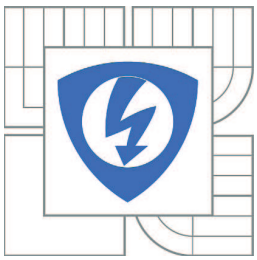
AUTOR PRÁCE — Bc. ONDŘEJ MACHEK

AUTHOR

VEDOUCÍ PRÁCE — prof. Ing. PAVEL JURA, CSc.

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Ondřej Machek	***ID:***	77773
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Fuzzy modely map pro pohyb mobilních robotů.

POKYNY PRO VYPRACOVÁNÍ:

Budování map pro robota pomocí fuzzy modelování a jeho navádění po mapě

DOPORUČENÁ LITERATURA:

1. Klir, G.J., Bo Juan: Fuzzy sets and fuzzy logic. Theory and application. Prentice Hall International, 1995. ISBN 0-13-101171-5

2. Babuška, R.: Fuzzy modeling for control. Kluwer Academic Publishers, 1998, ISBN 0-7923-8154-8.

Termín zadání:	7.2.2011	***Termín odevzdání:***	23.5.2011

Vedoucí práce: prof. Ing. Pavel Jura, CSc.

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

Abstrakt

Cílem diplomové práce bylo vytvoření algoritmu, který by pomocí neuronové sítě a neuro fuzzy sítě vytvářel topologickou mapu pro mobilního robota. V práci je navržen postup pro tvorbu topologické mapy pomocí tří klasifikátorů prostředí: dvě neuronové sítě a neuro fuzzy síť. Klasifikátory jsou mezi sebou navzájem porovnány. Dále je navržen řídící algoritmus pro prohledávání neznámého prostředí pro autonomního robota. Což vedlo ke zrychlení tvorby topologické mapy. Byl navržen simulační program v prostředí Matlab, který simuluje pohyb robota v neznámém prostředí.

Klíčová slova

Robotika, fuzzy, Kohonenova neuronová síť, neuro fuzzy síť, topologická mapa.

Abstract

This master thesis present a method for building topological maps for mobile robot navigation using neural network and neural fuzzy network. The master thesis concentrates on classification method. Neural fuzzy network is compared with two neural networks. It was also designed control algorithm exploration environment for autonomous mobile robot. This will rereduce the time to build the map. I developed simulation program in Matlab, which simulate move mobile robot in unknown environment.

Keywords

Robotic, fuzzy, Kohonen neural network, neuro fuzzy network, topological map.

Bibliografická citace:

MACHEK, O. FUZZY MODELY MAP PRO POHYB MOBILNÍCH ROBOTU. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 60s. Vedoucí diplomové práce byl prof. Ing. Pavel Jura, CSc.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma FUZZY MODELY MAP PRO POHYB MOBILNÍCH ROBOTŮ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **23. května 2011** ……
podpis autora

Poděkování

Velice bych chtěl poděkovat vedoucímu práce prof. Ing. Pavlu Jurovi, CSc za věnovaný čas, trpělivost, odbornou pomoc a další cenné rady při zpracování mé diplomové práce. Rád bych poděkoval svému bratrovi, Mgr. Ivu Machkovi za cenné rady a pomoc. Dále bych chtěl poděkovat své rodině a přítelkyni Tereze Vrbické za podporu a cenné rady.

V Brně dne: **23. května 2011** ……

podpis autora

Obsah

Seznam obrázků

1 ÚVOD

Jedním z nejdůležitějších problémů, které musí autonomní mobilní robot řešit je navigace v neznámém prostředí. V práci se zabývám tvorbou map pro mobilního robota ve vnitřním prostředí (dům, hala apod.). Zaměřuji se na tvorbu topologických map, které jsou reprezentovány formou grafu. Porovnávám různé přístupy pro tvorbu topologické mapy. Dále se zaměřuji na různé přístupy prohledávání neznámého prostředí a porovnání těchto metod. V práci se snažím optimalizovat prohledávání neznámého prostředí s přímou tvorbou topologické mapy a vytvářenou topologickou mapu optimalizovat tak, aby při následné cestě robota prostředím byla celková trasa co nejkratší. Dalším problémem který autonomní robot řeší při tvorbě mapy neznámého prostředí je určení, kdy je mapa již dokončena a robot může mapování zastavit.
Proto, aby si autonomní robot mohl nějakým způsobem plánovat cestu kudy pojede je zapotřebí, aby měl k dispozici informace o daném prostředí. K tomu účelu slouží mapy. Volba druhu mapy závisí především na prostředí ve kterém se robot pohybuje (rozloha, členitost).
Navrhuji simulační program pro autonomního robota, který si sám zmapuje pro něj neznámé prostředí a vytvoří topologickou mapu, podle které se bude poté schopen pohybovat. Porovnávám metody využívající neuronovou síť a neuro fuzzy síť.

2 FUZZY LOGIKA

Ostré (klasické) množiny. Teorie množin byla založena německým matematikem Georgem Cantorem (1845-1918)[7].

Klasické množiny můžeme chápat jakou soubor prvků. Každá množina může obsahovat buď konečný počet prvků $M = \{1,2,3\}$, nekonečný počet prvků $M = \{1,2,3, \dots\}$ nebo nemusí obsahovat žádný prvek prázdná množina $M = \emptyset$.

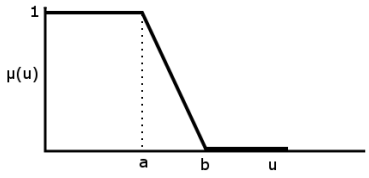
Za zakladatele fuzzy logiky (mlhavé logiky) se považuje Lotfim Zadeh, který roku 1965 zavedl teorii fuzzy množin.

Fuzzy logika nám umožňuje oprostit se od používání přesných údajů (např. hmotnost 1,536 kg), ale umožní nám zavést místo toho námi definované vágní pojmy (malá, střední, velká). V životě se většinou nerozhodujeme na základě precizních údajů, ale zavádíme vágní pojmy které podle našeho úsudku nejlépe vystihují danou situaci. Jako příklad můžeme uvést rychlost automobilu (0-30 km/h -> malá rychlost, 31-100-> střední rychlost, 100-200 km/h rychlost vysoká). Důvodem proč používáme vágní pojmy při rozhodování nebo popisu je také to, že nemáme většinou přesné informace.

Univerzum U je množina obsahující všechny prvky.

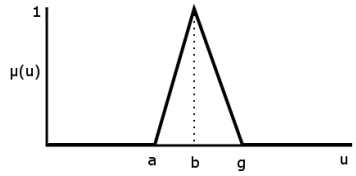
Funkce příslušnosti mapuje univerzum na celý interval $\langle 0,1 \rangle$. Každému bodu na univerzu U je přiřazena hodnota, která vyjadřuje příslušnost k dané množině. Různé funkce příslušnosti ukazují obrázky 2-1, 2-2, 2-3, 2-4 a 2-5. Podle vzorců (1.1), (1.2), (1.3), (1.4) a (1.5) vypočítáme příslušné hodnoty funkcí příslušnosti.

$$L(u,a,b) = \begin{cases} 0 & u \langle a \\ \dfrac{(u-a)}{b-a} & a \le u \le b \\ 1 & u \rangle b \end{cases} \qquad (1.1.)$$



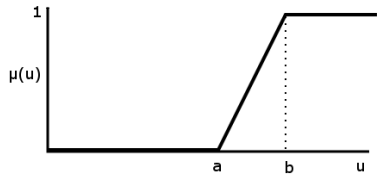
2-1 L funkce příslušnosti

$$\Lambda(u,a,b,g) = \begin{cases} 0 & u \langle a \\ \dfrac{(u-a)}{b-a} & a \le u \le b \\ \dfrac{g-y}{g-b} & b \le u \le g \\ 0 & u \rangle g \end{cases} \qquad (1.2.)$$



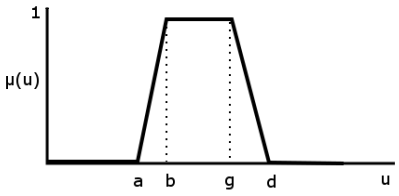
2-2 Λ funkce příslušnosti

$$\Gamma(u,a,b)=\begin{cases}0 & u\langle a\\ \dfrac{u-a}{b-a} & a\le u\le b\\ 1 & u\rangle b\end{cases} \qquad (1.3.)$$



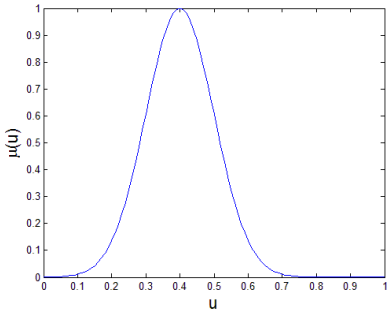
2-3 Γ funkce příslušnosti

$$\Pi(u,a,b,g,d)=\begin{cases}0 & u\langle a\\ \dfrac{(u-a)}{(b-a)} & a\le u\le b\\ 1 & b\le u\le g\\ \dfrac{(d-u)}{g-d} & g\le u\le d\\ 0 & u\rangle d\end{cases} \qquad (1.4.)$$



2-4 Π funkce příslušnosti

$$f(u,\sigma,c)=e^{\frac{-(x-c)^2}{2\sigma^2}} \qquad (1.5.)$$

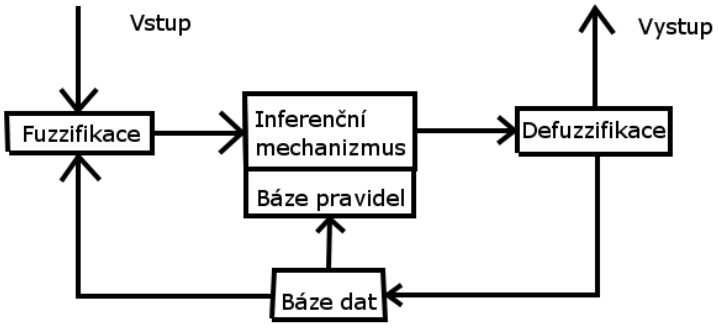


2-5 Gausian funkce příslušnosti

Na první pohled podle vzorců je vidět, že funkce Gausian je hladká funkce a bude výpočetně náročnější než předešlé funkce, proto se většinou používají funkce Λ, Γ a L.
Fuzzifikace je proces tvořený ze dvou částí. Normalizace a samotné fuzzifikace. Vstupní signál je ostrá hodnota, kterou musíme přepočítat do univerza většinou $< 0; 1 >$. Po normalizaci můžeme ostrý vstupní signál převést do fuzzy množin.
Defuzzifikace je proces skládající se z částí defuzzifikace a denormalizace. Defuzzifikace převede výstupní signál z fuzzy množiny na číslo. Denormalizace zajistí převedení výstupního signálu na zadanou fyzikální veličinu.
Báze pravidel definuje následující:

- *volba veličin, reprezentujících stav systémů a akční zásah do systému*
- *volba množin termů tj. hodnot jazykových proměnných*
- *volba obsahu antecedentů a konsekventů pravidel*
- *volba všech pravidel if-then[8]*

Inferenční mechanizmus vypočte výstupní veličiny na základě fuzzy pravidel.

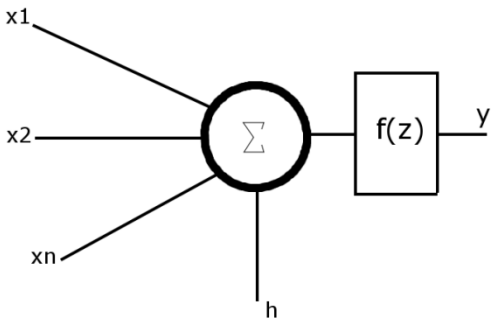


2-6 Schéma fuzzy řízení

3 NEURONOVÁ SÍŤ

Princip umělých neuronových sítí vychází z modelování biologických neuronů.
První umělé neurony byly vytvořeny Warrenem McCullochem a Walterem Pittsem v roce 1943. *Tyto neurony fungovaly tak, že byl jejich výstup 0 nebo 1 v závislosti na tom, jestli vážená suma vstupních signálů překročila prahovou hranici, nebo ne.[8]*

Na podobném principu fungují také dnešní neurony. Vstupy do neuronu jsou váhovány váhami w. Váhované vstupu jsou sumovány a následně je odečtena prahová hodnota h. Na výstup neuronu je připojena přenosová funkce, která určuje konečný výstup neuronu.

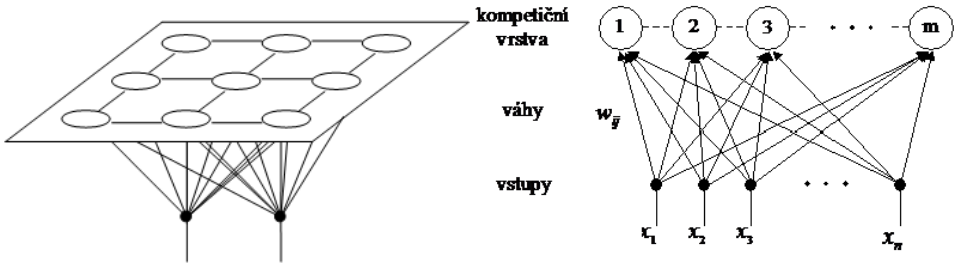


***3-1* Neuron**

$$y_{vys} = f\left(w_0 + \sum_{i=1}^{N} x_i w_i \right) \qquad (1.6.)$$

3.1 Umělá neuronová síť

Umělá neuronová síť je distribuovaný výpočetní systém sestávající z dílčích podsystémů (neuronů), který je inspirován neurofyziologickými poznatky o struktuře a činnosti neuronů a nervových systémů živých organizmů, a který je ve větší či menší míře realizuje.[1].



3-2 Kohonenova neuronová síť [1]

Kohonenova síť

Jde o typy sítí, které při učení nepotřebují učitele. Jsou založeny na algoritmu shlukové analýzy, tj. schopnost algoritmu, sítě, nalézt určité vlastnosti v závislosti přímo v překládaných trénovacích datech bez přítomnosti nějaké vnější informace.[21]

Jak můžeme vidět na obrázky 3-2, všechny neurony jsou ve stejné vrstvě. Do každého neuronu vedou všechny vstupy, které jsou váhovány.

Dále v textu uvažujeme neuronovou síť jako Kohonenovu neuronovou síť.

Aktivace (výstup) Kohonenovy sítě:

Výpočet vzdálenosti d_j udává vzdálenost jednotlivého vstupu x_i a příslušející váhy daného neuronu w_{ij} rovnice (1.7).

$$d_j = \sum_{i=1}^{N} \left[x_i - w_{ij} \right]^2 \qquad (1.7.)$$

j značí počet neuronů

i počet vstupů

Výsledek dostaneme pomocí rovnice (1.8), výsledkem pro nás není velikost vzdálenosti mezi vstupy a váhami, ale neuron, který má nejmenší hodnotu d_j (vítěz).

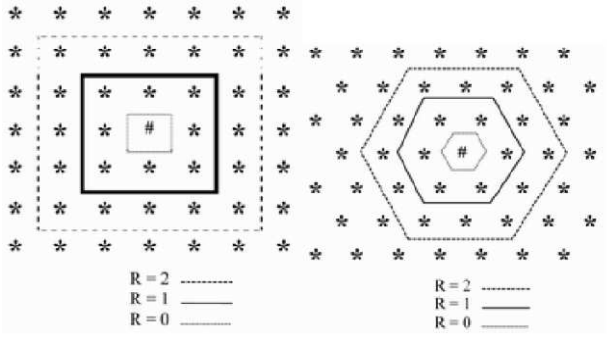
$$d_{j^*} = \min_j \left(d_j \right) \qquad (1.8.)$$

Učení Kohonenovy neuronové sítě

Kohonenova neuronová síť většinou využívá učení bez učitele. *Učení bez učitele se používá pro analýzu pozorování (nebo dat), když není k dispozici informace od učitele, tj. trénovací multimnožina.[22]*

$$w_j(t+1) = w_j(t) + \mu\left(x - w_j(t)\right) \qquad (1.9.)$$

Váhy vítězného neuronu adaptujeme podle rovnice (1.9). Parametr μ je koeficient učení, většinou se volí v rozsahu $\langle 0{,}2;\, 0{,}9 \rangle$. Jeho hodnota se může v průběhu učení měnit. Základem Kohonenovy sítě je uspořádaná množina neuronů obrázek 3-3. Při učení Kohonenovy sítě nemusíme učit pouze vítězný neuron, ale i neurony v jeho okolí. Okolí je na obrázku značeno 3-3 souvislými a přerušovanými čarami, neurony jsou znázorněny hvězdičkou.

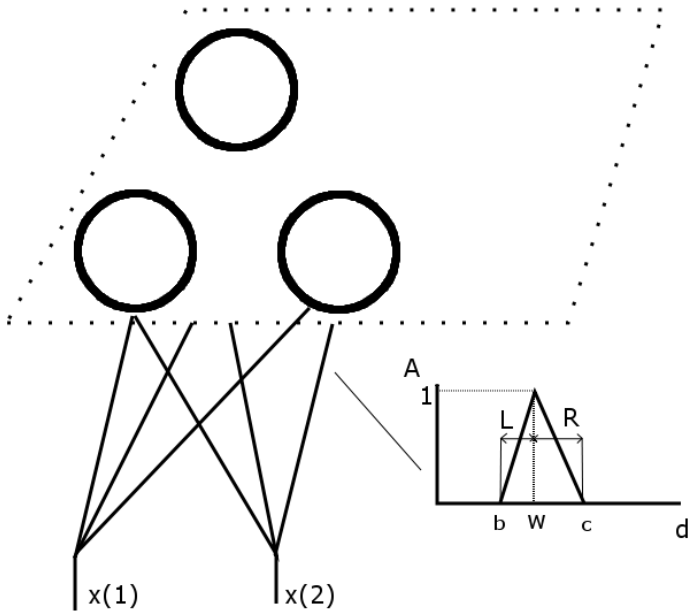


3-3 Struktura a uspořádání neuronů [23]

Dále budeme uvažovat okolí nastavené na nulu (učit se bude pouze vítězný neuron).

3.2 Neuro fuzzy sítě

Spojením neuronových sítí a fuzzy logiky může vzniknou velmi silný nástroj pro modelování a klasifikaci. Při popisu neuro fuzzy sítě jsem vycházel z článku [11]. Fuzzy logika vnáší do neuronových sítí zpracovávání vágních pojmů (neostrých hodnot). Fuzzy neuronovou síť popisuje obrázek 3-4.



3-4 Struktura neuro fuzzy

Síť připomíná svou topologií Kohonenovu neuronovou síť. Vstupy do jednotlivých neuronů nejsou tradičně váhovány, ale každý vstup je převeden, fuzzifikován. Jde o jiný způsob váhování, při kterém určujeme, jak moc aktuální vstup "sedí" k danému neuronu. Nastavení neuro fuzzy sítě určujeme pomocí trojúhelníkové funkce příslušnosti a její parametr w určuje polohu vrcholu funkce příslušnosti. Parametry b a c určují šířku (nosič) funkce příslušnosti.

Aktivace neuro fuzzy sítě:

Pro každý neuron spočítáme výstupní hodnoty podle rovnice (1.10).

$$A_i(d) = A_{i1}(d_1)A_{i2}(d_2)...A_{im}(d_m) \qquad (1.10.)$$

Za výstup považujeme neuron, který má největší hodnotu $A_i(d)$.

$$c = \max_c(A_i(d)) \qquad (1.11.)$$

Učení

Náhodně nastavíme středy w_{ij} fuzzy funkce příslušnosti v rozsahu $< 0,1 >$. Parametry R_{ij} a L_{ij} nastavíme na malé hodnoty.

Při učení používáme jako vstup ostré hodnoty, které před použitím normalizujeme do rozsahu $< 0,1 >$. Vypočítáme neuron, který budeme učit (u kterého budeme měnit parametry funkce příslušnosti w, b a a) podle vzorce (1.12) a (1.13).

$$O_i = \|d - w_i\| \tag{1.12.}$$

$$\|d - w_c\| = \min\|d - w_i\|, 1 \le i \le n \tag{1.13.}$$

Jednotlivé středy funkcí příslušnosti budeme měnit o Δw_c.

$$\Delta w_c = \eta(d - w_c) \tag{1.14.}$$

η je učící parametr, většinou volíme $< 0,2; 0,9 >$. Učící parametr se může v průběhu učení měnit.
Nové středy fuzzy množin upravíme podle rovnice (1.15).

$$w_c(t+1) = w(t) + \Delta w_c \tag{1.15.}$$

Dále upravíme *L* a *R* šířku fuzzy množiny. Pomocí rovnice (1.16) získáme absolutní diferenci mezi vstupem a středem fuzzy množin všech neuronů.

$$\Delta_{ij} = |d_j - w_{ij}| \tag{1.16.}$$

Suma absolutních diferencí

$$S_j = \sum_{i=1}^{n} \Delta_{ij} \tag{1.17.}$$

Výpočtem podle rovnice (1.18) defuzzifikuje vstupní hodnoty. Na obrázku 3-5 je graficky znázorněn postup výpočtu.

$$V_j = 1 - \frac{\Delta_{cj}}{S_j} \tag{1.18.}$$

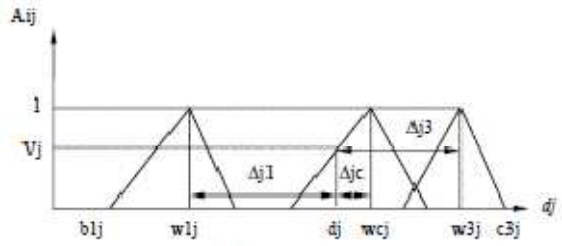


Fig. 3. Calculating the membership value V_j.

3-5 Učení neuro fuzzy síť

Určení levé a pravé šířky fuzzy množiny vzorce (1.19) a (1.20).

$$L_{cj}(t+1) = \begin{cases} L_{cj}(t), & d_j \ge w_{cj} \\ L_{cj}(t) + \eta\left(\left(\frac{w_{cj} - d_j}{1 - V_j}\right) - L_{cj}(t)\right), & d_j < w_{cj} \end{cases} \tag{1.19.}$$

$$R_{cj}(t+1)=\begin{cases} R_{cj}(t), & d_j \le w_{cj} \\ R_{cj}(t)+\eta\left(\left(\dfrac{d_j-w_{cj}}{1-V_j}\right)-R_{cj}(t)\right), & d_j > w_{cj} \end{cases} \qquad (1.20.)$$

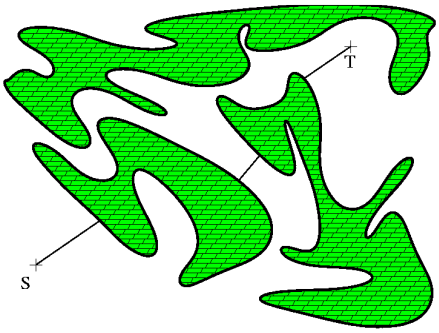
4 NAVIGACE MOBILNÍHO ROBOTA

4.1 Navigace

Navigaci můžeme rozdělit na lokální, globální a osobní [3].
a) Globální navigace je schopnost určit pozici robota v absolutních souřadnicích na zvolené mapě a plánovat trasu robota do zadaného cíle. Předpokládá se tedy už vytvořená mapa, na které se robot lokalizuje.
b) Lokální navigace je schopnost určit pozici robota vůči danému objektu například překážce. Na základě aktuálně zjištěných objektů upravuje svoje chování (směr jízdy, natočení apod.).
c) *Osobní navigace schopnost určit polohu vlastních komponent vůči sobě samému a okolí. [3]*

Mezi nejznámější globální navigační systémy patří GPS (Global Position System). Původně GPS vycházelo z projektu NAVSTAR GPS (Navigation Signal Timing and Ranging Global Positioning System). *Vývoj NAVSTAR GPS byl zahájen v roce 1973 sloučením dvou projektů určených pro určování polohy System 621B a pro přesné určování času Timation[4]*. Projekt do roku 1964 fungoval pouze pro vojenské účely, po roce 1964 byl uvolněn i pro civilní použití (zejména navigace jachet). V roce 1995 projekt GPS dosáhl plné operační způsobilosti. *Družice obíhají kolem země na šesti kruhových drahách se sklonem 55˚ ve výšce 20200 km na zemí[4]*.
Globální navigaci můžeme realizovat na vlastní mapě, kterou robotu dáme nebo si ji vytvoří. Příkladem může být mapa vytvořená pomocí neuronové sítě, kde dané neurony reprezentují jednotlivé oblasti.

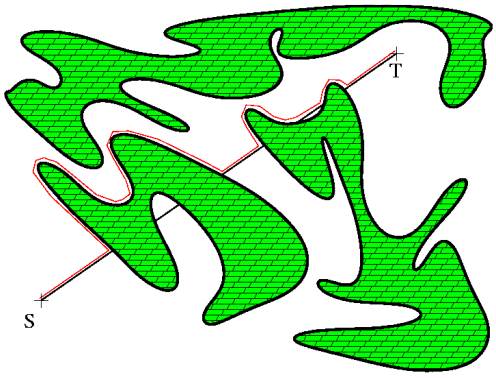
*Pod pojmem **lokální navigace** se skrývá problematika řešeni odhadu relativní pozice robota vůči okolním objektům a způsob interakce s blízkým prostředím.*[20] Mezi používané algoritmy pro lokální navigaci patří Bug1, Bug2, VisBug, fuzzy logika a neuro-fuzzy. Při popisu jsem vycházel z [6].
Bug1 je jednoduchý a snadno implementovatelný algoritmus, který robota dovede k cíli ve zcela neznámém prostředí.
Algoritmus:
1) vydej se směrem k cíli
2) pokud jsi v cíli KONEC - cesta nalezena
3) pokud narazíš na překážku označ si tento bod H (hit point) a obcházej překážku
4) monitoruj nejbližší bod k cíli (označ jej L = leave point) a ujetou vzdálenost
5) po návratu do H zvol kratší cestu zpět do L
6) pokud směr k cíli vede do překážky KONEC - cíl je nedosažitelný
7) GOTO 1 [6]
Algoritmus Bug1 ilustruje obrázek 4-1.



4-1 Algoritmus Bug1[6]

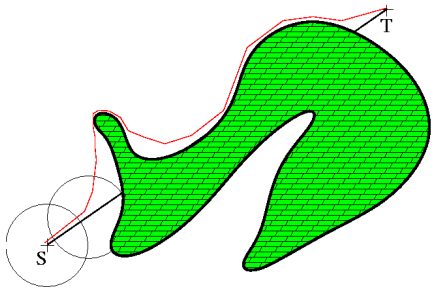
Robot se ze startovní pozice (S) vydá směrem k cíli (T), narazí na překážku, kterou celou objede a označí si bod s nejkratší vzdáleností od cíle (L). Vrátí se nejkratší cestou do bodu L a znovu pokračuje k cíli. Nevýhodou toho algoritmu je objíždění celé překážky a vracení se zpět do bodu L. Výhodou je, že pokud je to možné algoritmus by měl dospět vždy do cíle.

Algoritmus **Bug2** odstraňuje hlavní nevýhodu algoritmu Bug1. Postupuje podobně, ale na začátku si vytvoří úsečku ST. Pokud narazí na překážku začne ji obcházet, dokud nenarazí na úsečku u nebo se nevrátí do původního bodu, kdy začal překážku obcházet. Pokud při obcházení překážky narazí na úsečku ST, vydá ze znovu směrem po úsečce ST k cíli. Algoritmus Bug2 ilustruje obrázek 4-2.



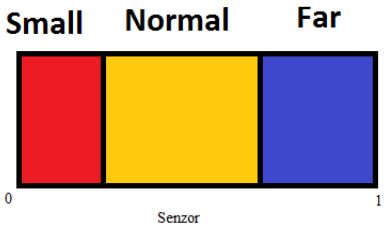
4-2 Algoritmus Bug2[6]

VisBug vychází z algoritmu Bug2. U VisBug musí být robot vybaven senzorem, který nám dává přehled o celkovém okolí robota. *Jako základ zase použijeme Bug2, ale jeho činnost budeme pouze simulovat do okamžiku, kdy se nám ztratí z dohledu a teprve pak se daným směrem vydáme[6].*
Situaci znázorňuje obrázek 4-3.

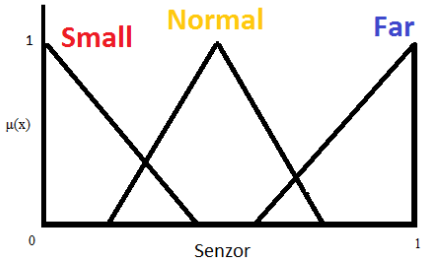


4-3 Algoritmus VisBug [6]

Pro lokální navigaci můžeme využít **fuzzy logiku**. Při použití fuzzy logiky nás nemusí zajímat přesná vzdálenost od překážky, ale vstupní hodnoty ze snímačů (metrická informace) fuzzifikujeme pro každý senzor zvlášť (small, medium, far) obrázek 4-5. Obrázky 4-4 a 4-5 ukazují rozdíl mezi klasickou ostrou množinou a fuzzy množinou. V případě použití senzorů, kde by data mohla být hodně zašuměná nebo nejednoznačná, je lepší zvolit klasifikace pomocí fuzzy.



4-4 Klasifikace do ostré množiny



4-5 Klasifikace do fuzzy množiny

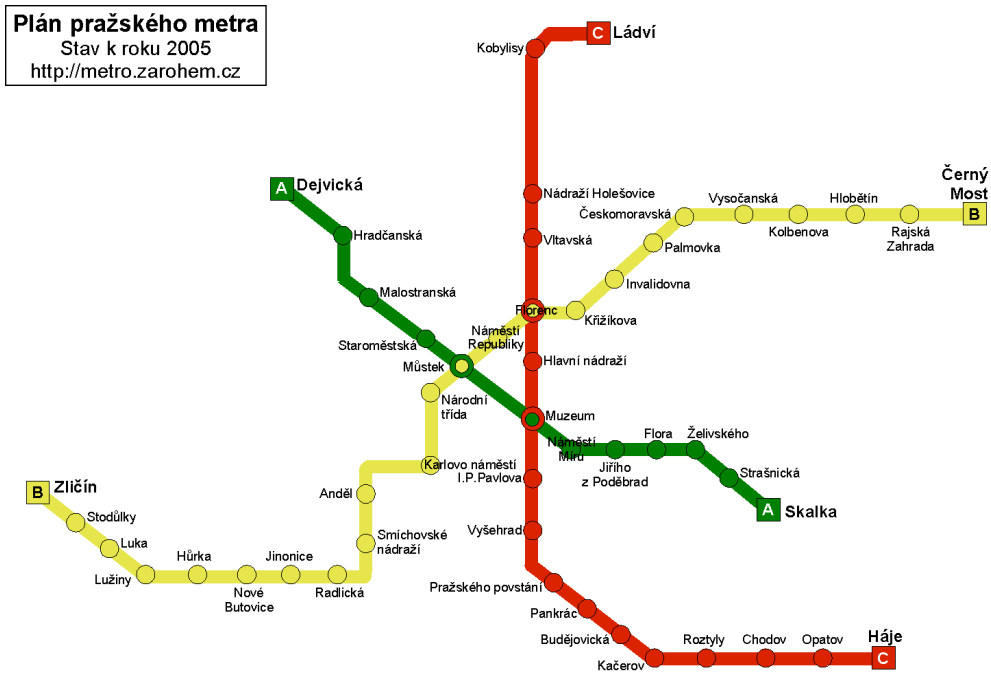
Podobně můžeme klasifikovat úhel robota a překážky (small , medium, large).
V nejjednodušším případě bychom si mohli fuzzy rozhodování představit takto:
if (úhel s překážkou small) and (senzor small) then (zatoč ostře).
Existují však mnohem sofistikovanější algoritmy, které využívají více parametrů (situace, minulá situace, požadovaný směr z nadřazené řídící jednotky) dále mají uloženy více strategií pro navigaci a umožňují mezi nimi libovolně přecházet.

4.2 Mapy

Mezi nejčastější mapy používané v robotice pro navigaci je mapa na mřížce (mřížková mapa), topologická mapa nebo kombinace obou.

Mřížková mapa nejčastěji využívá pravděpodobnosti nebo fuzzy logiky. Každé buňce je přiřazena určitá pravděpodobnost, že se v ní nachází překážka. Na podobném principu pracuje i mapa s fuzzy logikou, každé buňce přiřadí určitý stupeň příslušnosti (obsazeno/volno).

S topologickými mapami se setkáváme v běžném životě, i když si to zrovna nemusíme uvědomovat viz obrázek 4-6. Každé oblasti (regionu) přidělíme uzel a uzly se podle daných možností pospojují. Při kombinovaném přístupu topologické mapy a mapy na mřížce se využívá výhod obou map. Mapa na mřížce se může udržovat pouze částečná pro aktuální pozici robota a nemusí být tak rozsáhlá. Tím by například umožnila rychlejší pospojování bodů v topologické mapě.



4-6 Topologická mapa plán pražského metra [9]

Mezi výhody mřížkové mapy patří: znalost pozice překážky, snadné představení si modelu prostředí, dobrá navigace na mapě. Mezi nevýhody můžeme zařadit velkou paměťovou náročnost, při velkém prostředí pomalejší nalezení optimální cesty.

Mezi výhody topologické mapy patří především: malá paměťová náročnost, rychlé nalezení cesty i v rozsáhlém prostředí, hodí se pro rozsáhlé objekty. Mezi nevýhody můžeme zařadit: složitější algoritmus, nemusí vždy nalézt optimální cestu mezi startem a cílem, delší a náročnější proces tvorby mapy prostředí.

4.2.1 Mapa na mřížce

Při popisu jsem vycházel [10]. Pro sebelokalizaci robota a jeho pohyb v prostředí, ať už vnitřní (domy, haly) nebo venkovní, je důležité znát, kde se aktuálně nachází a mít

alespoň částečnou mapu prostředí, ve kterém se pohybuje. Robot si při své cestě může mapu také vytvářet za použití ultrazvukových senzorů nebo laserových proximitních senzorů.
Při budování mřížkové fuzzy mapy je mapa rozdělena pomocí mřížky na buňky. Každé buňce je přidělen stupeň příslušnosti OBSAZENO a NEOBSAZENO. Výsledná fuzzy mapa předává informaci o případné možné kolizi s objektem. Jak už bylo řečeno mapa na mřížce může využívat pravděpodobnostního přístupu (pro každou buňku je spočítaná pravděpodobnost, jestli se v dané buňce nachází překážka) nebo fuzzy přístupu, kde je vypočítána funkce příslušnosti (neobsazeno/obsazeno) pro každou buňku.
Další postup demonstruje použití fuzzy logiky při tvorbě mapy na mřížce. Před každým vyhodnocením bychom měli pro lepší přesnost naměřit n dat z dané oblasti a tyto výsledky agregovat podle vzorce (1.21).

$$\mathrm{E} = \cup_n \mathrm{E}_n$$
$$\mathrm{O} = \cup_n \mathrm{O}_n \tag{1.21.}$$

4-7 Postup určení obsazenosti buňky v mřížce [10]

Výpočet fuzzy mapy:
Rovnice (1.22) označuje stupeň rozporu mezi buňkami Empty a Occupy.

$$A = \mathrm{E} \cap O \tag{1.22.}$$

Pomocí rovnice (1.23) zvýrazníme buňky které jsou nejvíce nerozhodné.

$$I = \overline{E} \cap \overline{\mathrm{O}} \tag{1.23.}$$

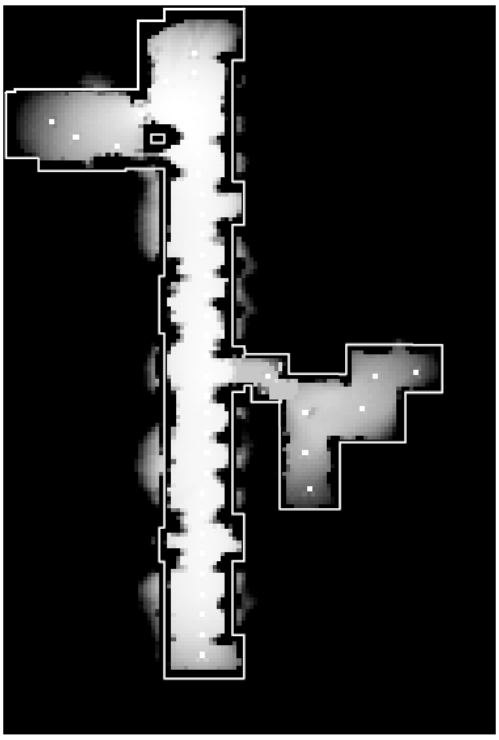
Celkovou mapu spočítáme podle vzorce (1.24) buňky s vyšší hodnotou představují překážku.

$$S = E^2 \cap \overline{O} \cap \overline{A} \cap \overline{I} \tag{1.24.}$$

Mapa pro pohyb robota, kde buňky s vyšší hodnotou představují volnou cestu rovnice (1.25.)

$$M = \overline{S} \tag{1.25.}$$

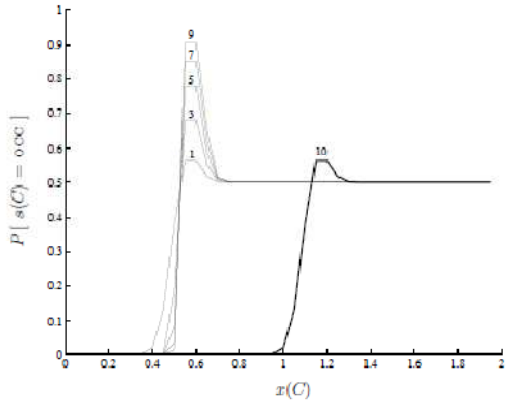
Vytvořenou fuzzy mapu můžeme vidět na obrázku 4-8.



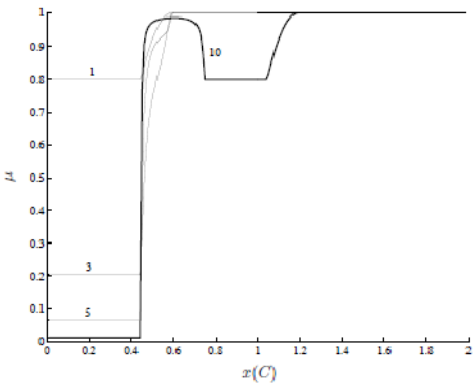
4-8 Fuzzy mapa [10]

Srovnání pravděpodobnostního přístupu a fuzzy mapy

Literatura [10] uvádí příklad, kde bylo provedeno deset měření ultrazvukovými senzory $r_1 = r_2 \ldots = r_9 = 0{,}6\ m$ a $r_{10} = 1{,}2\ m$ pro porovnání přístupu fuzzy a pravděpodobnostního. Rozdíly zobrazují obrázky 4-9 a 4-10. U pravděpodobnostního přístupu obrázek 4-9 je vidět že buňka ve vzdálenosti 0,6 m není označena jako obsazená i když zde byla devět krát naměřena překážka.



4-9 Výsledek deseti měření pravděpodobnostní přístup [10]

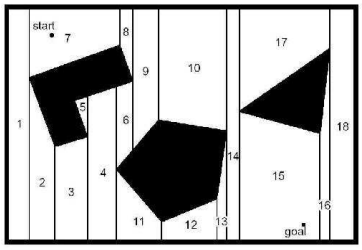


4-10 Výsledek deseti měření fuzzy přístup [10]

Naopak u fuzzy přístupu je buňka ve vzdálenosti $0{,}6\,[m]$ pokládána za obsazenou.

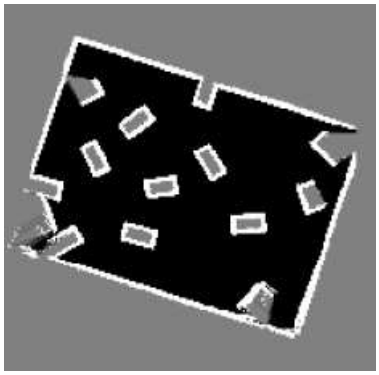
4.2.2 Geometrická mapa

Geometrická mapa reprezentuje prostředí pomocí vhodně zvolených geometrických entit.[2] Geometrické entity mohou být například přímka, křivky vyšších řádů, lichoběžníky apod.. Příkladem geometrické mapy může být obrázek 4-11, použili se zde různé geometrické tvary.

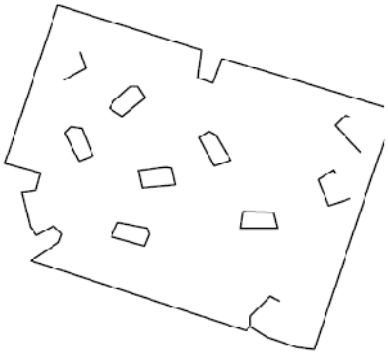


4-11 Geometrická mapa tvary lichoběžník, trojúhelník [2]

Geometrickou mapu můžeme tvořit buď přímo z dat ze senzorů nebo pomocí již vytvořené mapy (například mřížkové mapy), tuto variantu zobrazuje obrázek 4-12 a 4-13. Na obrázku 4-12 máme zobrazenou mřížkovou mapu (světlá místa značí překážky a tmavá volnou cestu), pomocí které vytvoříme mapu na obrázku 4-13. Pro získání geometrické mapy byly provedeny operace: *segmentace, dilatace a eroze[2]*.



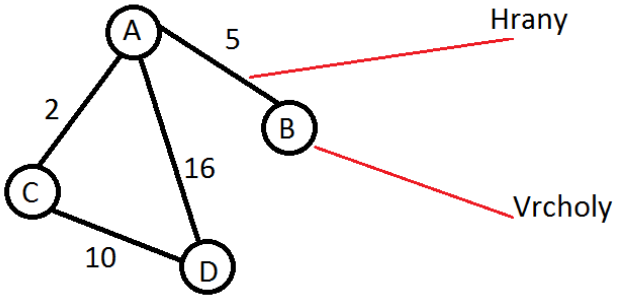
4-12 Mřížková mapa [2]



4-13 Geometrická mapa vytvořená z mřížkové [2]

4.2.3 Topologická mapa

Topologickou mapu si ve většině případů můžeme představit jako graf, který má několik uzlů a uzly jsou mezi sebou pospojovány. Na rozdíl od jiných metod (mřížka obsazenosti, geometrická mapa) topologická mapa většinou nezobrazuje na mapě překážky, ale uzly (vrcholy) reprezentují významné oblasti, tyto oblasti jsou spojeny pomocí hran. Uzly často nesou informaci o své poloze, případně informaci co daná oblast představuje (například roh). Hrany grafu (spojnice vrcholů) mohou být ohodnocené (metrickými údaji) například vzdáleností mezi jednotlivými uzly grafu (patnáct metrů) nebo neohodnocené představující pouze spoj. U navigace mobilních robotů budeme uvažovat hrany pouze ohodnocené metrickými údaji. Příklad ohodnoceného grafu je na obrázku 4-14.



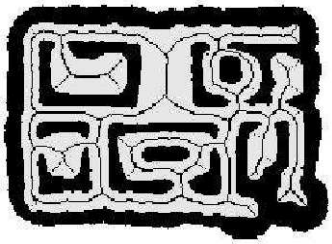
4-14 Ohodnocený graf

Stavba topologické mapy z mřížky obsazenosti:
Tvorba topologické mapy podle S. Thrun a A. Buckena:

a) *segmentace mřížkové mapy*
b) *vytvoření Voronoiho diagramu*
c) *nalezení rozdělujících bodů*
d) *vytvoření rozdělujících úseček*
e) *vytvoření topologické mapy[2]*

ad a) Nejdříve musíme nalézt hranici pro segmentaci mřížky obsazenosti na volno/obsazeno. Mřížkovou mapu segmentujeme do binárních čísel 0 a 1.

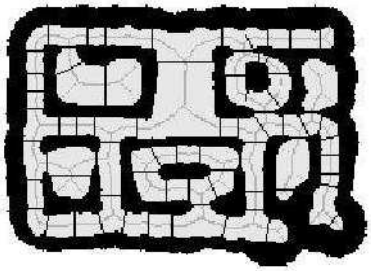
ad b) Pro každý bod volného prostoru definujeme množinu nejbližších bodů k překážce (pokud je bodů více musí být stejně vzdálené) obrázek 4-15.*[2]*



4-15 Voronoi diagram [2]

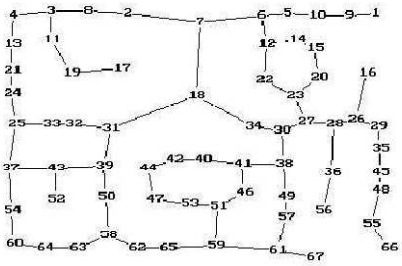
ad c) Nalezneme rozdělující body

Rozdělující úsečka je spojnice rozdělujícího bodu a nejbližšího obsazeného bodu v mřížce.[2]



4-16Nalezení rozdělujících bodů Voronoi diagram [2]

ad d) Vytvoření topologické mapy 4-17.



4-17 Topologická mapa pomocí Voronoi diagramu [2]

Tvorba topologické mapy z dat ze senzorů

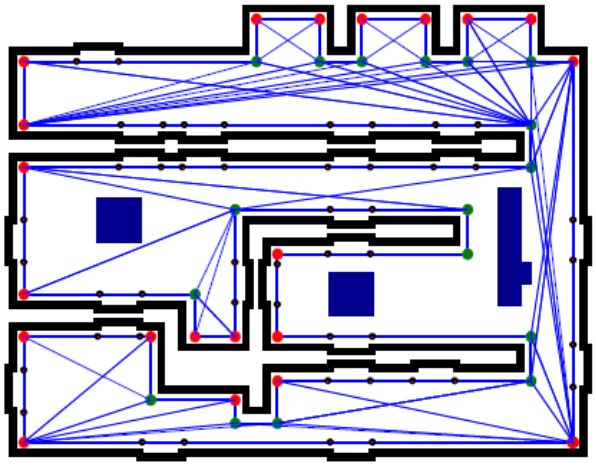
Topologická mapa může být tvořena přímo z dat senzorů. Abychom mohli takto mapu vytvořit, musíme být schopni data ze senzorů odpovídajícím způsobem klasifikovat do určitých shluků. Mezi nejčastěji používané klasifikátory patří neuronová síť konkrétně kohonenova síť (SOFM - Self Organizing Feature Map), můžeme se setkat i s různými variantami v podobě neuro-fuzzy klasifikátoru. Množina stavů, které bude robot schopen klasifikovat, může být předem daná, a to buď předem naučená (klasifikace rohů, zdí apod.) nebo nemusí být vůbec známa a robot se na dané prostředí naučí sám a sám bude rozhodovat, co jak bude klasifikovat.

4.2.4 Symbolická mapa

Symbolická mapa je nástavbou pro již vytvořenou mapu. Přiřazuje za pomoci operátora (člověka) významným místům na mapě symboly-jména (místnost A, židle, dveře apod.). Jako příklad můžeme uvést: robot se má dostat z laboratoře 216 do místnosti 516 a přitom projet přes vstupní halu. Mezi symboly mohou vznikat relace (např. místnost A má dveře, dveře jsou cestou z místnosti A).

4.3 Prohledávání prostoru

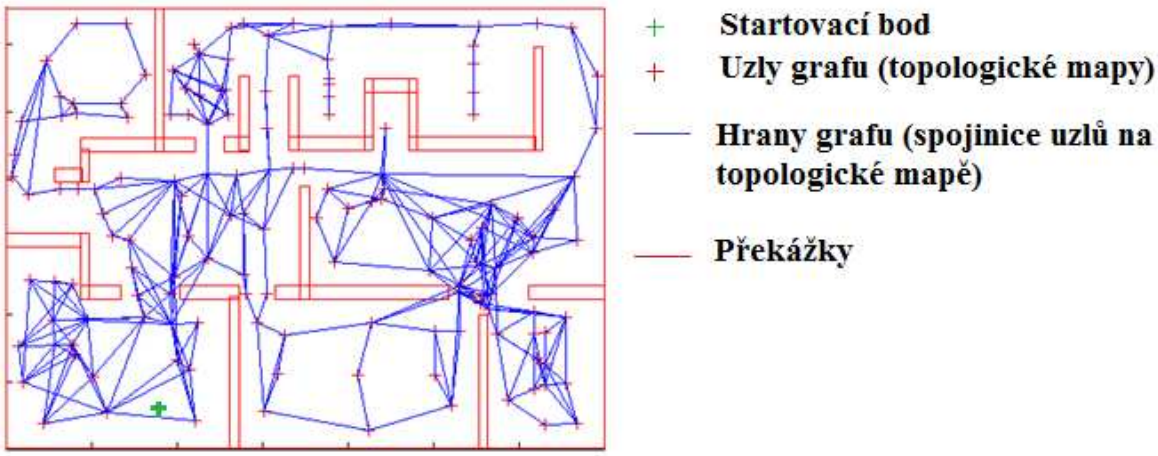
Důležitou volbou při tvorbě map je zvolený způsob prohledávání prostoru. Prohledávání prostoru ovlivňuje to, jak bude celá mapa vypadat. Algoritmus prohledávání neznámého prostoru by měl zajistit prohledání všech oblastí a ukončit prohledávání po vytvoření kompletní mapy.
Můžeme rozlišit dva základní typy prohledávání prostoru:
Náhodné prohledávání, algoritmus volí náhodný směr robota. Výhodou je jednoduchý algoritmus. Nevýhodou je, že algoritmus nemusí vždy prozkoumat všechny oblasti a dlouhá doba prohledávání prostoru.
V případě implementace **řídícího algoritmu** můžeme výrazně snížit dobu tvorby mapy, její kvalitu a měli bychom dosáhnout prohledání všech dosažitelných prostorů.
Mezi algoritmy pro prohledávání neznámého prostoru můžeme zařadit algoritmus prohledávání prostoru podél překážek. Robot se vydá k nejbližší překážce a začne ji objíždět. Při zjištění významných oblastí (roh, konec stěny) ukládá jejich pozice a vždy nově vytvořený uzel spojí s předcházejícím. Významné oblasti mohou být rohy, volný prostor, výčnělky ve zdech. Situaci znázorňuje obrázek 4-18. Výhodou tohoto přístupu je dobrá znalost o překážkách a jejich tvarech.



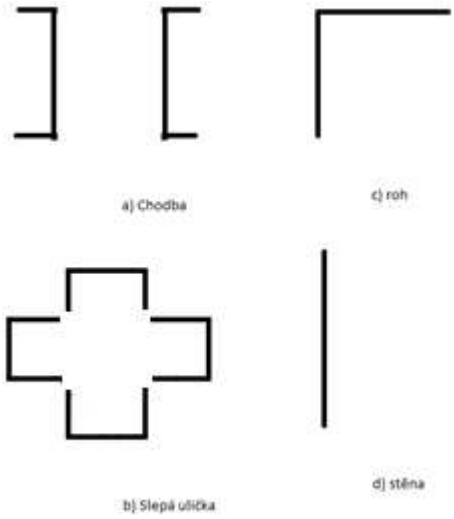
4-18 Prohledávání podél překážek [17]

Jakmile se robot dostane zpět do výchozího bodu, bude se snažit pospojovat všechny body mezi sebou, aby dosáhl co nejkratší cesty při navigaci (vytvořená mapa na obrázku 4-18).

Předchozí algoritmus se pohyboval pouze podél překážek. Můžeme ale použít algoritmus, který se bude pohybovat v prostoru, takto vytvořenou mapu ukazuje obrázek 4-19 který byl vytvořen v simulaci v rámci diplomové práce. Robot se řídí podle oblasti ve které se pohybuje a podle oblasti minulé. Při přechodu z jedné oblasti do druhé se rozhoduje jakou akci vykoná (udržuj směr, spoj uzly apod.). Robot při prozkoumávání oblasti neměl žádné apriorní informace o mapě nebo prostředí ve kterém se pohybuje k dispozici měl pouze údaje ze čtyř senzorů. Na obrázku 4-19 je vytvořená topologická mapa, kde zelený křížek zobrazuje startovní místo robota, červené křížky zobrazují uzly topologické mapy a modré spoje představují ohodnocené propojení uzlů na topologické mapě. Příklady oblastí, které by robot mohl klasifikovat jsou na obrázku 4-20.



4-19 Topologická mapa

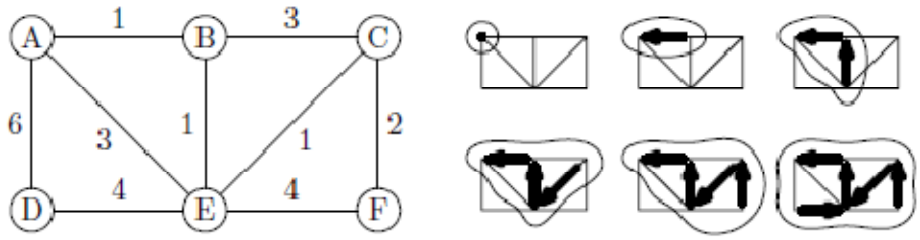


4-20 Klasifikace oblastí

5 DIJKSTRŮV ALGORITMUS

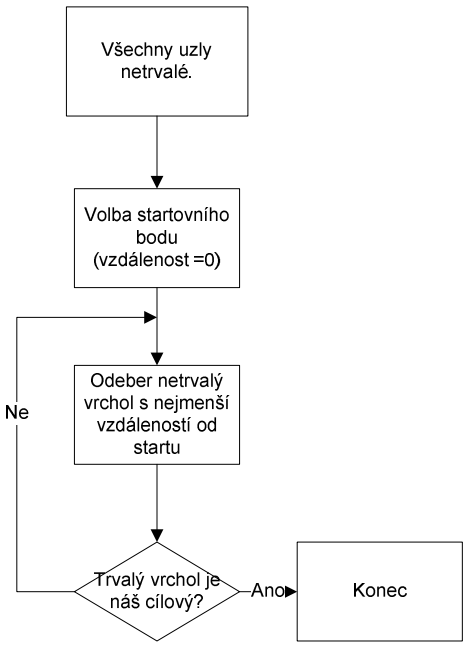
Vycházel jsem z článku [18]. Dijkstrův algoritmus slouží k hledání nejkratší cesty v ohodnoceném i neohodnoceném grafu. V případě neohodnoceného grafu ohodnotíme všechny hrany stejnou konstantou. Jedná se o algoritmus prohledávání grafu do šířky. Podmínkou je, že hrany grafu nesmí být záporně ohodnocené.

Algoritmus: Všechny uzly označíme za netrvalé a vzdálenosti nastavíme na nekonečno (maximální hodnota). Startovacímu uzlu nastavíme vzdálenost 0. Hledáme v grafu netrvalý vrchol s nejmenší vzdáleností (nalezneme startovní bod). Startovní bod označíme za trvalý. Přepočítáme vzdálenosti sousedů u nově trvalého uzlu. Projdeme opět netrvalé vrcholy a najdeme ten s nejkratší vzdáleností od startu. Uzel prohlásíme za trvalý a přepočítáme vzdálenosti u jeho sousedů. Celý postup opakujeme dokud neprohlásíme hledaný uzel za trvalý.



5-1 Dijkstrův algoritmus průběh [18]

Obrázek 5-1 znázorňuje průběh Dijkstrova algoritmu. Zakrouškované vrcholy jsou prohlášeny za trvalé. Šipky znázorňují strom nejkratší cesty na trvalých vrcholech. [18]



5-2 Dijkstrův algoritmus

Dijkstrův algoritmus se používá pro nalezení nejkratší cesty na topologické mapě, která je formou grafu s ohodnocenými spoji. Spoje mohou být ohodnoceny buď metricky nebo mohou označovat obtížnost terénu, případně kombinaci obou metod.

Dijkstrův algoritmus najde nejkratší cestu v grafu, může ale nastat situace, kdy bychom měli dvě stejně ohodnocené cesty v grafu, potom by pro nás byla nejžádanější cesta přes nejméně uzlů. Tento problém řeší Floyd-Warshallùv algoritmus. Pracuje s ohodnoceným grafem s nezáporným ohodnocením. Floyd-Warshallùv algoritmus najde nejkratší cestu v grafu, ale v případě dvou různých cest se stejným ohodnocením, vybere tu s nejmenším počtem hran.

6 TOPOLOGICKÁ MAPA

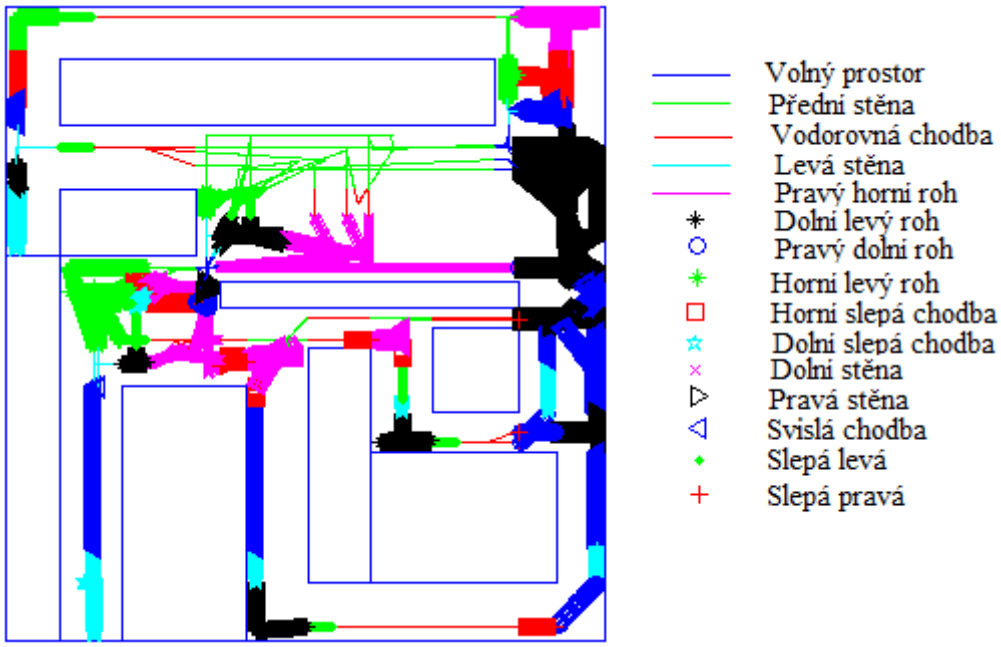
6.1 Neuronová síť neuro fuzzy

Při řešení tvorby topologické mapy pomocí neuronové sítě jsem vycházel z článku A.Kurze [13] a při řešení pomocí neuro fuzzy sítě jsem vycházel z článku [12]. Metodu popsanou A.Kurzem [13] jsem se snažil navíc vylepšit pomocí přednaučení neuronové sítě na určité situace, tak abych urychlil proces tvorby mapy a robot nemusel projíždět neznámé prostředí dvakrát. Jako další klasifikátor byla použita neuro fuzzy síť. Dále byl vytvořen algoritmus pro autonomní prohledávání neznámé oblasti.

6.1.1 Princip metody

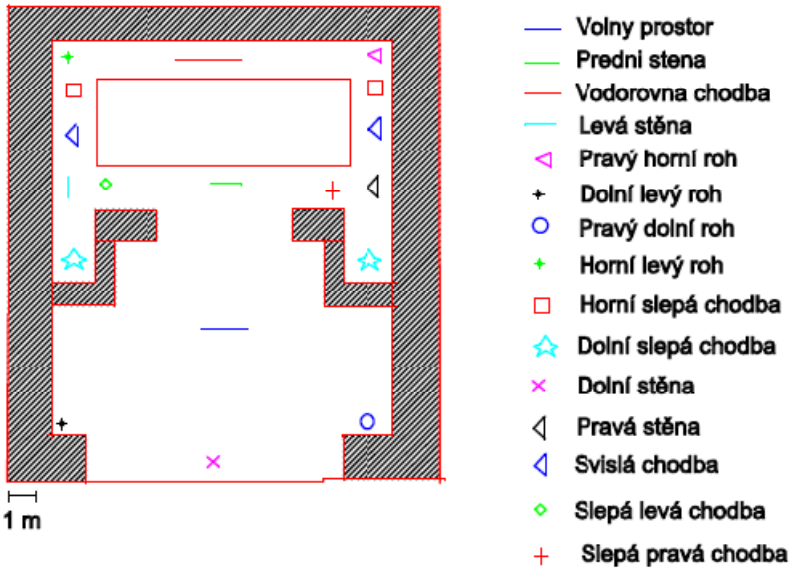
Pro klasifikaci různých oblastí a tvorbu topologické mapy lze použít Kohonenovu neuronovou síť popsaná v kapitole (3.1) nebo neuro fuzzy síť kapitola (3.2). V metodě uvedené podle A.Kurze [13] se na začátku vytvoří neuronová síť s náhodnými váhami. Robota necháme náhodně projíždět v neznámém prostředí, přitom se učí klasifikovat různé situace (mění váhy). Po ukončení učení, které je dáno počtem cyklů (jeden cyklus trvá 2,5 s), začne robot prozkoumávat prostředí za účelem tvorby topologické mapy (klasifikuje prostředí). Robot se v neznámém prostředí pohybuje náhodně (náhodně mění směr jízdy tak, aby nenarazil do překážky). Ukončení prohledávání prostředí je dáno počtem cyklů. Díky tomu, že se robot pohybuje náhodně, není zaručeno prohledání celé mapy. Robot popsaný v článku [13] je vybaven celkem 24 ultrazvukovými senzory s nastaveným prahováním na 2 m. Robot je postavený na diferenciálním podvozku.

Vylepšení navrhované metody spočívá v přednaučení neuronové sítě na situace, které chci klasifikovat. Tyto situace jsou: přední stěna, vodorovná chodba, levá stěna, pravý horní roh, dolní levý roh, pravý dolní roh, horní levý roh, horní slepá chodba, dolní slepá chodba, dolní stěna, pravá stěna, svislá chodba, slepá levá chodba, slepá pravá chodba. Na obrázku 6-1 vidíme trasu robota a jeho klasifikaci prostředí do 15 oblastí. V obrázku 6-1 jsou body hodně blízko u sebe a některé symboly nejdou rozpoznat. Na obrázku 6-2 jsou umístěny pouze jednotlivé symboly pro dané oblasti . Jako další klasifikátory pro srovnání byly použity neuronová síť (bez přednaučení), a neuro fuzzy síť (bez přednaučení). V případě kdy robotu přednaučíme neuronovou síť pro klasifikaci prostředí, odpadá nám nutnost prohledávání prostředí dvakrát. Síť neuro fuzzy by mohla být také přednaučena na klasifikaci předem určených oblastí, ale výsledky by byly stejné. Kvůli větší výpočetní náročnosti neuro fuzzy sítě byla před učena pouze neuronová síť.



6-1 Klasifikace do oblastí

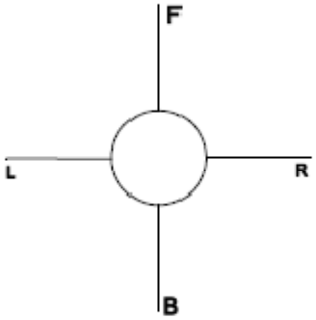
Při prozkoumávání prostředí pomocí řídícího algoritmu kapitola 6.2.1.1 robot přímo při jízdě tvoří topologickou mapu, kontroluje její úplnost a vzájemnou polohu všech uzlů na mapě tak, aby uzly vůči sobě zaujímaly pozice podle daných pravidel.



6-2 Klasifikace a dané situace

6.1.2 Robot

Při návrhu je robot pokládán za hmotný bod. Je vybaven čtyřmi proximitními senzory pro měření vzdálenosti obrázek 6-3. Podvozek byl zvolen synchronní, hlavně kvůli jeho výhodám při odometrii, další výhodou je, že při své cestě nedochází ke změně orientace robota. Při popisu synchronního podvozku jsem vycházel z článku [20].

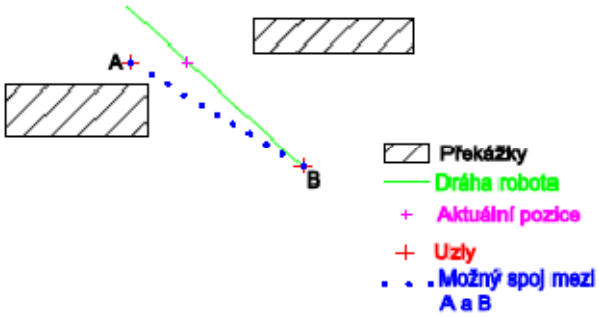


6-3 Robot

U synchronních podvozků se nejčastěji vyskytují konstrukce se třemi nebo čtyřmi koly obrázek 6-5. Oproti diferenciálnímu podvozku, kde se zatáčí pomocí rozdílu rychlostí kol na obou stranách, dosahuje robot se synchronním podvozkem lepších výsledků v odometrii [15]. Při používání odometrie mohou vznikat chyby například prokluzem kol, chybou snímače natočení kol. Nevýhodou odometrie je integrace chyby.
U synchronního podvozku uvažujeme robota kde nedochází ke změně orientace. Stejného natočení kol může dosáhnout buď elektronicky (poloha snímána senzory) nebo mechanickým spřažením kol.

6.1.3 Vytvoření a spojení uzlů topologické mapy

Při **přímé tvorbě** topologické mapy se uzel vytvoří vždy, když robot přejede z jedné oblasti do druhé. Pozice uzlu (souřadnice x a y) je spočítána jako průměr všech zaznamenaných pozic v předešlé oblasti. Při vytvoření uzlu si robot pamatuje, jaký typ situace uzel znázorňuje. V případě, že robot projel v blízkosti již vytvořeného uzlu, který znázorňuje stejnou oblast, jsou všechny uzly shlukovány do jednoho uzlu. Sousedi všech uzlů, které byly shlukovány, budou přeneseny do nového uzlu. Při vytvoření nového uzlu je nový uzel spojen s uzlem minulým mimo vytvoření prvního uzlu. Uzly mohou být spojovány i během jízdy. Robot pro tento případ zaznamenává pozice nejbližších překážek ze všech čtyř směrů. V případě, že robot při své cestě narazí na uzel A, který by mohl spojit s uzlem B, který byl vytvořen jako poslední, spojí oba uzly. Aby k tomu mohlo dojít, musí být uzel A v aktuálním dosahu senzorů a senzor na něj musí přímo mířit, dále žádná ze zaznamenaných překážek nesmí bránit spojení obou. Uzlů situaci znázorňuje obrázek 6-4.

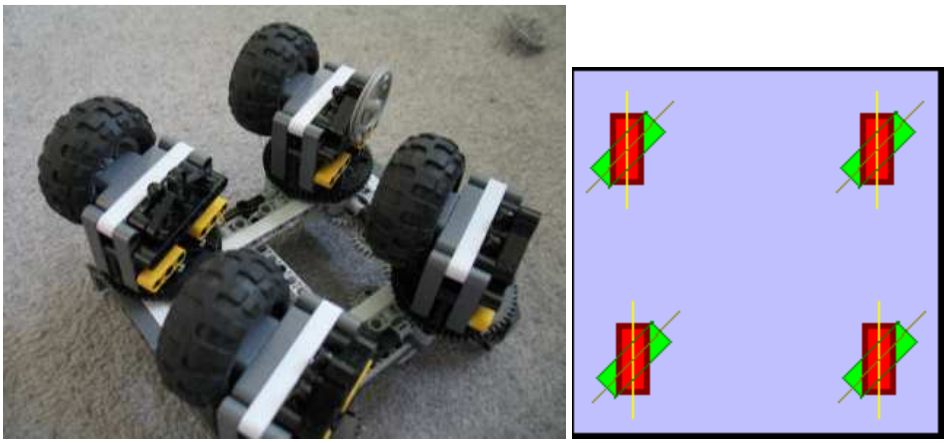


6-4 Spojení uzlů

V případě použití klasifikátorů, které nebyly přednaučeny je použita **nepřímá tvorba** mapy. Robot si v průběhu cesty zaznamenává souřadnice (x,y), kde se nacházel a klasifikovanou oblast. Po ukončení prohledávání mapy zpracuje postupně zaznamenané pozice, vytvoří uzly (průměr souřadnic x a y) a spojení je mezi oblastmi které na sebe navazovali.

6.1.4 Vnímání prostředí

Robot musí být vybaven senzory pro měření vzdáleností od překážek, aby mohl klasifikovat prostředí. Data ze senzorů musí být nejdříve předzpracována. Prvním důležitým krokem předzpracování je prahování. Volbu prahu si musíme zvolit, závisí na daném prostředí. Práh je maximální hodnota, kterou bude senzor ukazovat. Kdybychom měli senzor, který bude měřit do dva a půl metrů, ale nastavená hodnota prahu by byla dva metry, dostávali bychom od senzorů údaje vždy jenom maximálně dva metry. Práh byl zvolen 2 m stejně jako v článku [12], pro lepší srovnání dosažených výsledků.
Dalším důležitým krokem v předzpracování dat je nastavení a udržování jednoho předem definovaného směru. To je velice důležitý, krok aby byla data ze snímačů nezávislá na daném prostředí a natočení robota v prostředí. Nezávislosti dat ze senzorů můžeme dosáhnout pomocí velkého množství senzorů, kde má robot kolem sebe kruh ze senzorů a při natočení robota můžeme virtuálně v kruhu snímače posouvat obrázek 6-6. Dalším možným řešením je použití takové konstrukce robota, kde nebude docházet ke změně orientace. Takovým příkladem může být robot se synchronním podvozkem obrázek 6-5. Synchronní podvozek má dva stupně volnosti, kola se natáčejí do stejného úhlu a otáčí se stejnou rychlostí.



6-5 Synchronní podvozek [15]

6-6 Senzorický systém [14]

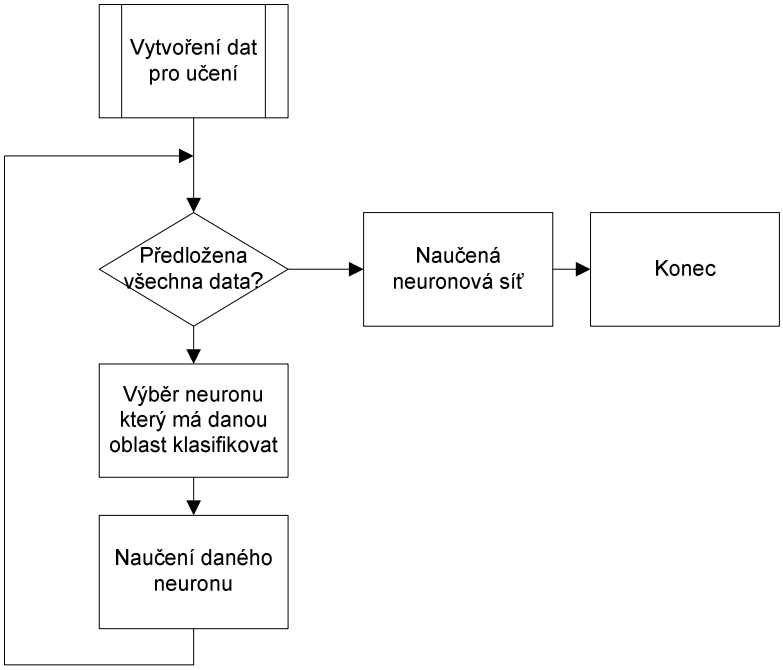
6.2 Klasifikátory

Při návrhu byly použity tři klasifikátory: neuronová síť přednaučená, neuronová síť, která se učila během projíždění neznámého prostředí a neuro fuzzy síť, která se také učila během projíždění neznámého prostředí. Důvodem, proč nebyla neuro fuzzy síť přednaučena také, je, že dávala stejné výsledky a samotná neuronová síť je výpočetně méně náročnější.

6.2.1 Neuronová síť přednaučená

Vycházel jsem z článku A.Kurze[13], kde neuronová síť nebyla nijak naučena (byla nastavena náhodně). Robot projížděl náhodně neznámé prostředí a přitom se učil klasifikovat oblasti. To znamenalo prostředí projíždět dvakrát. Jednou když se učil a podruhé tvořil topologickou mapu prostředí. Vylepšením této metody bylo přednaučení

neuronové sítě na předem definované oblasti: přední stěna, vodorovná stěna, levá stěna, pravý horní roh, dolní levý roh, pravý horní roh, horní levý roh, horní slepá chodba, dolní slepá chodba, dolní stěna, pravá stěna, svislá chodba, slepá levá chodba, slepá pravá chodba. Rozdělení do více oblastí už nepřinášelo výrazná zlepšení. Ačkoliv to není obvyklé pro Kohonenovu síť, učení bylo s učitelem. Obrázek 6-7 zobrazuje algoritmus přednaučení neuronové sítě.



6-7 Přednaučení Kohonenovy neuronové sítě

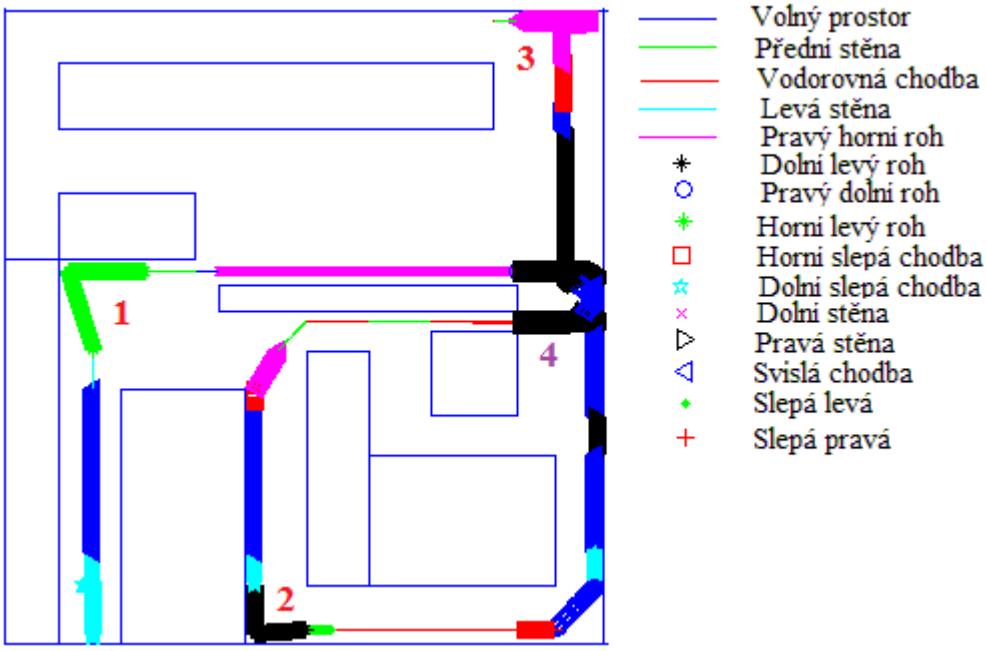
Blok **Vytvoření dat pro učení** Kohonenovy sítě pro každou oblast vytvoří 5000 různých vzorů. Každé dané oblasti je předem přidělen neuron který by ji měl klasifikovat. Učení bylo popsáno v kapitole (3.1), jediný rozdíl je v tom, že se neurčuje, který neuron zvítězí, ale vítěz je předem daný.

Při volbě klasifikátorů, které se učí během jízdy, kdy projíždějí neznámé prostředí, rozlišujeme dva druhy prohledávání prostředí pomocí: řídícího algoritmu kapitola (6.2.1.1) nebo náhodné projíždění neznámého prostředí.

6.2.1.1 Řídící algoritmus prohledávání prostředí

Autonomním řízením robota v tomto případě rozumíme, autonomní prohledání prostředí a tvorbu topologické mapy bez zásahu operátora. Robota umístíme na definovanou pozici a spustíme prohledávání. Robot se na začátku rozjede určeným směrem a dále se řídí podle pravidel. Pravidla se vždy aplikují na základě posledně identifikované oblasti a současně identifikované oblasti. Na obrázku 6-8 vidíme, jak robot projíždí neznámou trasu. V případě, kdy zaznamená přechod z oblasti levá stěna do levý horní roh (situace 1), zkontroluje nejdříve, jestli takový uzel ve své blízkosti již nemá a pokud nemá, vydá se směrem do daného rohu. Jakmile dorazí do středu rohu

kde by neměl kam dál jet, rozhodne se na základě posledně vytvořeného uzlu ,jakým směrem pojede. Posledním uzlem byla levá stěna, robot se nachází v rohu, proto pojede směrem od daného rohu doprava. Podobně robot řeší situace 2 a 3. Může nastat situace, kdy robot pojede do prostředí, které již prozkoumal a kde již uložil bod (situace 4). V tomto případě robot bude směřovat k danému uzlu, aby propojil posledně uložený uzel s uzlem již vytvořeným. Jakmile oba uzly propojí, zkontroluje, jestli, jsou všechny vytvořené uzly dostatečně a správně propojeny. V případě, že existují uzly, které potřebují upravit pozici svých sousedů robot pojede k uzlu, který je vzdušnou čarou nejblíže od místa, kde se nachází a upraví daný uzel. K cestě do zadaného uzlu je již navigován pomocí vytvářené topologické mapy. Vyzkoušena byla i varianta, kdy nebyl hledán nejbližší uzel vzdušnou čarou, ale byla spočítána délka cesty ke každému bodu, který potřeboval upravit pozici sousedů pomocí údajů z vytvářené topologické mapy. Následně byl vybrán uzel, který byl cestou nejblíže. Tato varianta však vykazovala delší dobu nutnou k prohledání a vytvoření mapy prostředí. Mohl by nastat případ, že robot bude pomocí topologické mapy navigován přes překážku (špatná klasifikace oblastí, uzel se posunul), proto je imlementován algoritmus pro lokální navigaci Bug2 popsaný v kapitole 4.1.

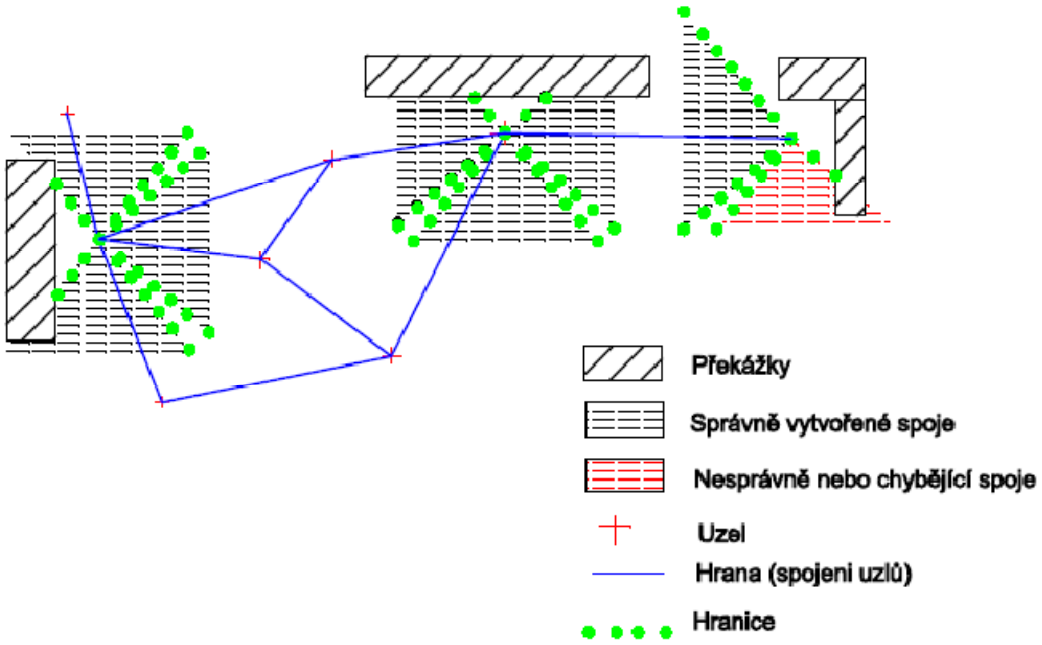


6-8 Autonomní řízení rozhodování na základě oblastí

Kontrola uzlů na mapě

Pro každý typ uzlu je definováno, kde se bude nacházet jeho soused, resp. jaký úhel mezi sebou musí svírat. Situaci znázorňuje obrázek 6-9. Robot vytvořil několik uzlů a některé mezi sebou spojil. Na obrázku 6-9 jsou tři uzly, dva mají spoje vytvořené správně, třetímu chybí spojení s jiným uzlem níž. První uzel z leva na obrázku 6-9 je

uzel definující oblast levá stěna, tento uzel by měl mít souseda nacházející se ho nad sebou, pod sebou a vpravo od něj. Druhý uzel zleva je uzel přední stěna, tento uzel by měl mít sousedy vpravo, vlevo a pod sebou. Třetí uzel je typu pravý horní roh, tento uzel by měl mít sousedy vlevo od sebe a pod sebou. Analogicky uvažujeme další uzly: uzel typu dolní stěna by měl mít sousedy vpravo a vlevo od sebe a nad sebou, uzel typu dolní levý roh by měl mít sousedy nad sebou a vpravo od sebe. Sousedi by tedy měli být vždycky na stranách kde se nevyskytuje překážka.

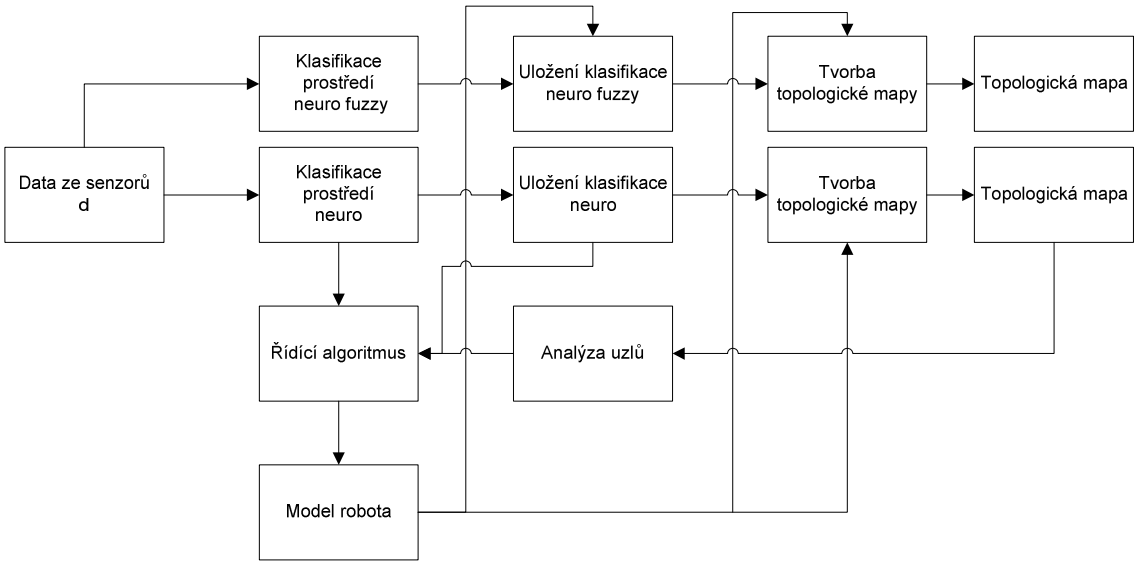


6-9 Analýza uzlu

Velikost úhlu, kdy je ještě hrana (spoj) klasifikovaná za správnou, byla nastavena empiricky. Obrázek 6-10 ukazuje, jaké meze byly testovány pro kontrolu spojení mezi uzly.

6-10 Nastavení uhlu při analýze uzlů

Nakonec byla zvolena hodnota 80˚, mapa byla dobře propojena a rychlost tvorby mapy vysoká. Při nastavení 40˚ byla tvorba mapy velice pomalá a byla hodně citlivá na změnu polohy uzlů. Uzly během prozkoumávání oblastí mohou měnit svoji pozici a proto může nastat situace kdy uzel, který měl všechny sousedy správně, se posune tak, že jeho vzájemná poloha uzlu a jeho sousedů již nebude vyhovovat. V takovém případě algoritmus dá pokyn robotu, aby jel k uzlu, který nemá správně sousedy a pokusil se upravit jeho pozici případně jeho sousedů, aby vyhovovala vzájemná poloha uzlu jeho sousedů. Uzel je dál uložen do seznamu uzlů které se již nebudou upravovat.
Na obrázku 6-11 je zobrazeno schéma řízení robota pomocí řídícího algoritmu.



6-11 Struktura prohledávání pomocí řídícího algoritmu

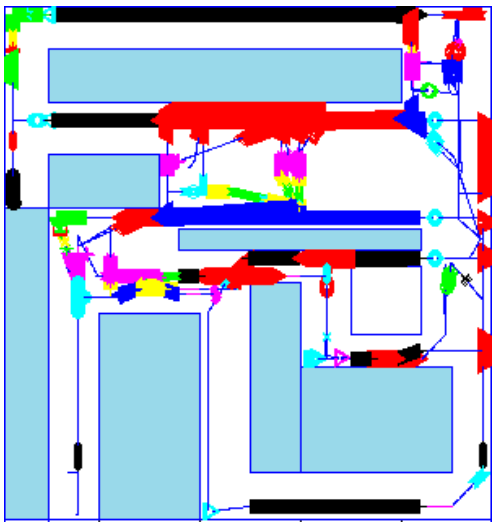
Data d ze čtyř senzorů slouží nejdříve ke klasifikaci dané situace. Klasifikaci prostředí provádí všechny tři klasifikátory: neuronová síť (přednaučená), neuro fuzzy síť (bez přednaučení) a neuronová síť (bez přednaučení). Blok, řídící algoritmus na základě aktuálně klasifikované situace, minule klasifikované situace a bloku analýza uzlů, rozhodne o směru robota. V bloku tvorba topologické mapy se počítají a spojují jednotlivé uzly topologické mapy kapitola (6.1.3). Blok topologická mapa obsahuje uloženou topologickou mapu. V bloku analýza uzlů se kontrolují jednotlivé uzly a jejich vytvořená propojení, blok dále rozhoduje o ukončení prohledávání prostředí (prostředí je zmapované).

6.2.2 Neuro fuzzy sítě

Při učení Kohonenovy sítě a neuro fuzzy sítě záleží na pořadí dat, jaká mu předkládáme k učení. V případě použití učení popsaného v kapitole (3.2) neuro fuzzy sítě a prohledávání neznámého prostředí pomocí řídícího algoritmu docházelo při klasifikaci k velké generalizaci. Většina klasifikovaných dat byla klasifikována jako

jedna stejná oblast. Z tohoto důvodu byly vytvořeny dva algoritmy pro učení neuro fuzzy sítě. Jeden pro učení při prohledávání neznámého prostředí pomocí řídícího algoritmu a druhý pro prohledávání neznámého prostředí náhodně. Výsledky klasifikace neuro fuzzy sítí můžeme vidět na obrázku 6-12, robot při učení neuro fuzzy sítě projížděl prostředí pomocí řídícího algoritmu a na obrázku 6-13 robot při učení projížděl neznámé prostředí náhodně. Robot v obou případech ujel stejnou vzdálenost a pak bylo učení zastaveno, počet trénovacích dat byl v obou případech 10000. Robot znovu prohledal neznámé prostředí, tentokrát se už neučil, ale prostředí klasifikoval. Nevýhodou při prohledávání prostředí pomocí řídícího algoritmu je velký počet vytvořených neuronů a některé neurony nejsou později při klasifikaci použity. Každá barva a symbol na obrázcích 6-12 a 6-13 představuje jinou oblast. Pouze celistvé oblasti vyplněné modrou barvou znázorňují překážky.

Algoritmus učení neuro fuzzy sítě při prohledávání prostředí, kdy se robot řídí pomocí řídícího algoritmu, je na obrázku 6-14. Na začátku tvoří neuro fuzzy síť jeden neuron. V průběhu učení se do sítě postupně přidávají neurony. Neuron je vždy přidán, pokud je splněna podmínka $\|d - w_c\| > \mathrm{k}$, kde k je námi zvolený parametr, určuje práh, kdy bude přidán nový neuron. Hodnota k byla určena empiricky na hodnotu 0,2. Při této hodnotě již nedocházelo k tomu, že jeden neuron klasifikoval všechny oblasti a zároveň se vytvořil nejmenší počet neuronů. Výpočet $\|d - w_c\|$ udává jak moc dobře se daný neuron hodí kde klasifikaci zadaných vstupních hodnot. V podmínce $\|d - w_n\| > \mathrm{b}$ určuje b míru naučení daného neuronu, w_n jsou váhy nového neuronu. Hodnota b byla nastavena na základě zkoušení na hodnotu 0,15.

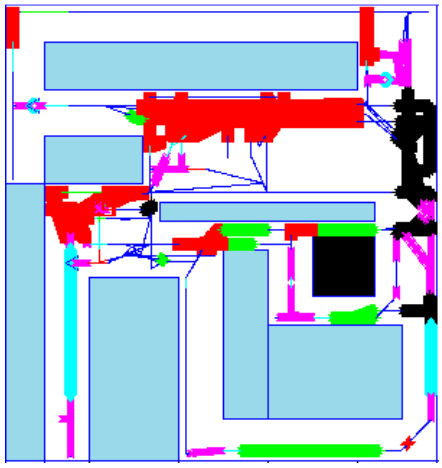


6-12 Klasifikace pomocí neuro fuzzy prohledávání prostředí pomocí řídícího algoritmu

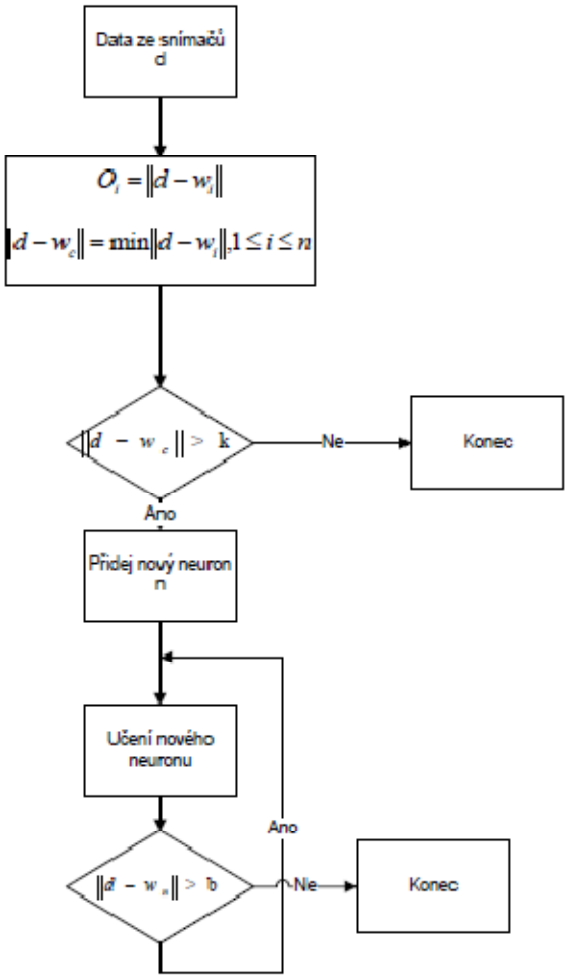
Při prohledávání neznámého prostředí náhodně (robot mění svůj směr náhodně, ale tak aby nenarazil do překážky) je vytvořena síť s pevným počtem neuronů a

náhodnými parametry. Výsledek klasifikace znázorňuje obrázek 6-13. Síť se učí přímo z dat, které dostává od senzorů, algoritmus učení byl popsán v kapitole (3.2).

Pro vytvoření topologické mapy se u neuro fuzzy používá nepřímá tvorba mapy kapitola (6.1.3).



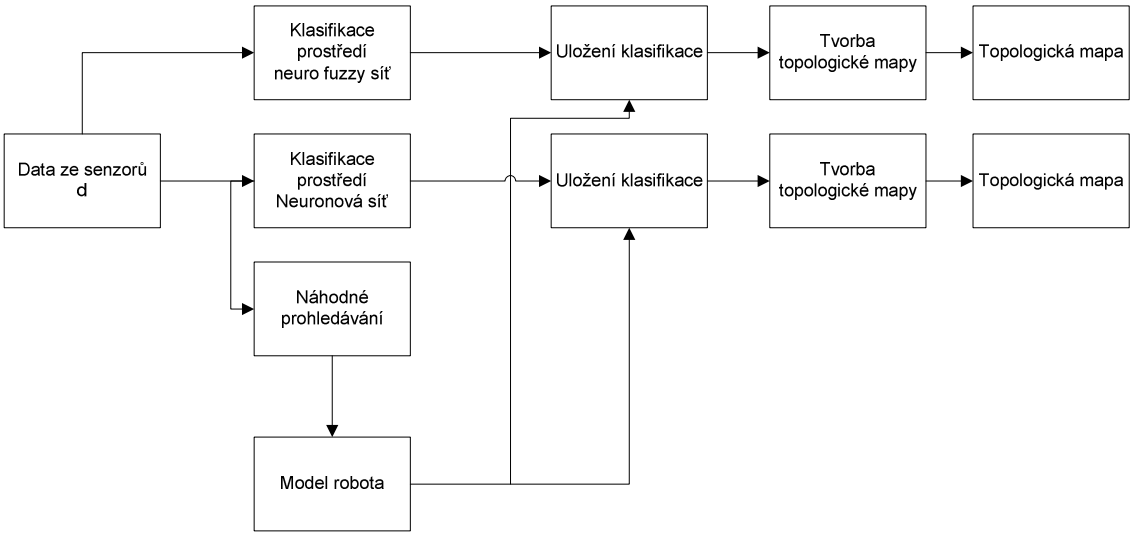
6-13 Neuro fuzzy klasifikace prohledávání prostředí náhodně



6-14 Algoritmus učení neuro fuzzy sítě při prohledávání prostředí řídícím algoritmem

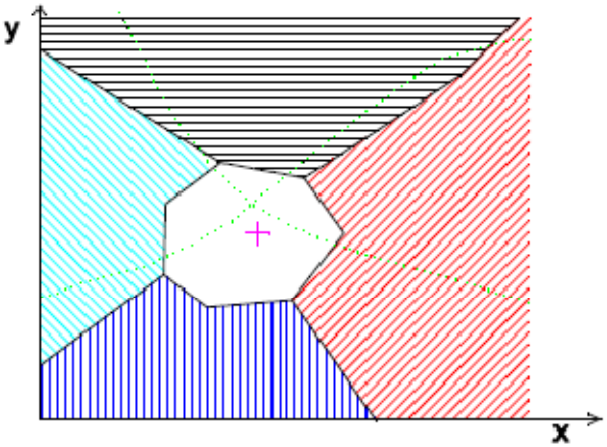
6.2.2.1 Náhodné prohledávání neznámého prostředí

V případě, že nepoužijeme řídící algoritmus, který by určoval směr robota, můžeme použít náhodné prohledávání, kdy robot mění svůj směr náhodně, ale zároveň tak, aby nenarazil do nějaké překážky.



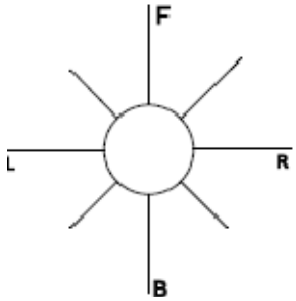
6-15 Struktura náhodného prohledávání neznámé oblasti

V tomto případě se nekontroluje vzájemná poloha uzlů ani jestli je mapa již dokončena a neznámé prostředí prohledáno. Ukončení prohledávání prostředí je dáno počtem cyklů (počet dat ze senzorů). Blok **Data ze senzorů** předává informace o naměřených hodnotách ze snímačů. Podle naměřených dat jsou data v blocích **Klasifikace prostředí neuro fuzzy síť** a **Klasifikace prostředí Neuronová síť** klasifikována do jednotlivých oblastí. Data jsou uložena v blocích **Uložení klasifikace** společně se souřadnicemi (x,y). Po ukončení prohledávání prostředí je z uložených klasifikovaných oblastí vytvořena topologická mapa. Uzly jsou vytvořeny jakou průměr souřadnic (x, y), v dané jedné oblasti. Situaci znázorňuje obrázek 6-16, robot v průběhu prozkoumávání projel dvakrát přes jednu oblast. Souřadnice z obou drah (zelená barva) jsou použity k tvorbě uzlu pro danou oblast.



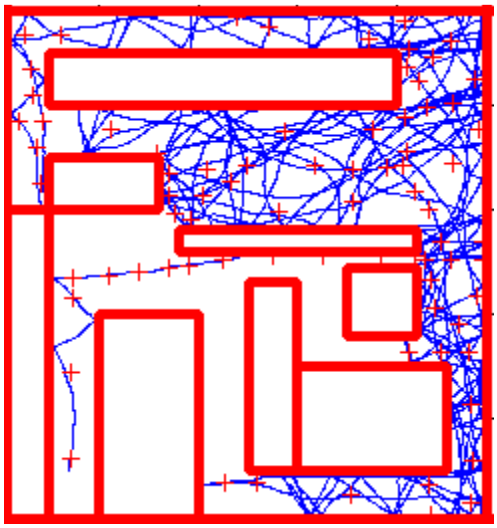
6-16 Vytvoření uzlu

Protože robot by při prohledávání oblasti mohl narazit do nějakého rohu, byli mu přidány senzory situaci znázorňuje obrázek 6-17.



6-17 Robot použitý při náhodném prohledávání

Obrázek 6-18 ukazuje cestu kterou jel robot při náhodném prohledávání oblasti. Robot ujel dvojnásobnou dráhu než při prohledávání pomocí řídícího algoritmu. Modré křivky ukazují cestu robota, červené křížky značí uzly na topologické mapě která byla tvořena přímo za běhu.

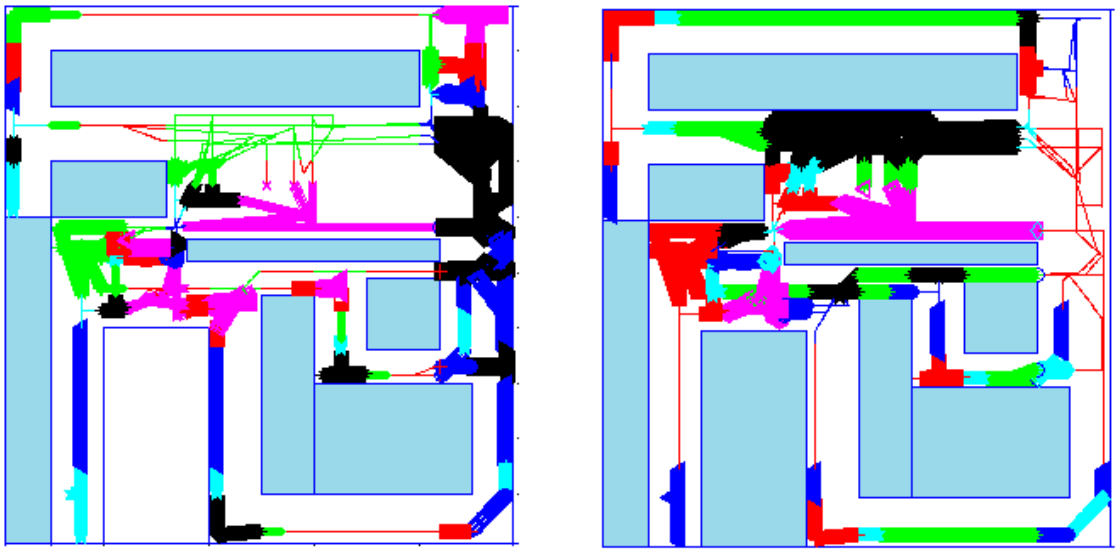


6-18 Náhodné prohledávání prostředí

6.2.3 Neuronová síť

Dalším klasifikátorem je Kohonenova neuronová síť, která nebyla přednaučena. Síť je popsána v kapitole (3.1). Pro porovnání s neuronovou sítí, která byla přednaučena byla vytvořena neuronová síť o stejném počtu neuronů, kde byly váhy nastaveny náhodně. Robot se nejdříve při prohledávání prostředí učil klasifikovat oblasti. Po ukončení učení, které bylo dáno počtem předložených vzorů, robot znovu prohledával prostředí, ale tentokrát oblasti klasifikoval. Na obrázku 6-19 a) je znázorněna klasifikace přednaučené neuronové sítě b) neuronové sítě, která byla naučena v průběhu

prohledávání neznámého prostředí. Na obrázku je patrné, že obě sítě takřka totožně klasifikují oblasti. Tato metoda byla popsána A.Kurzem v článku [13].



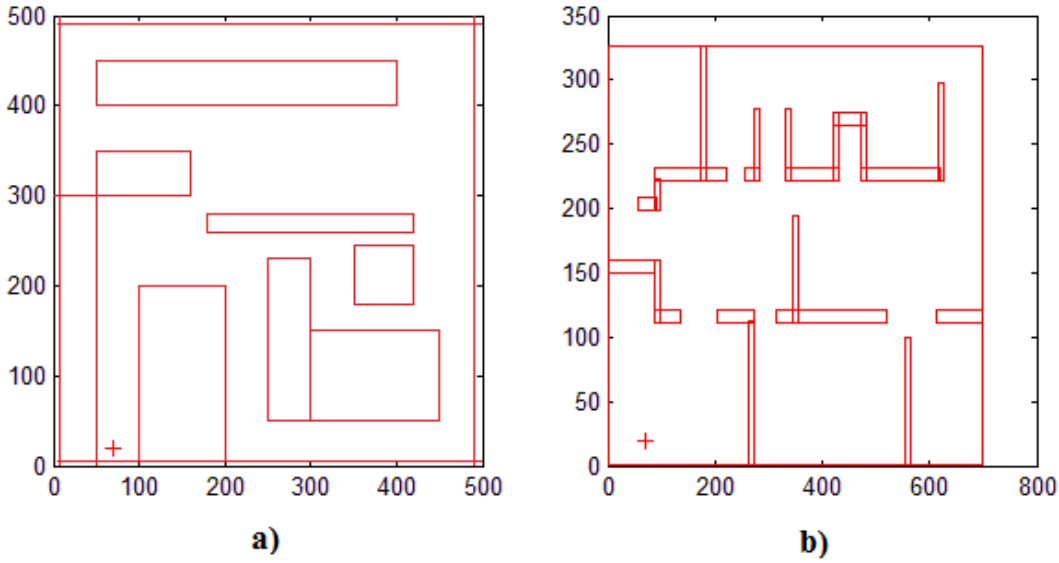
a) b)

6-19 Klasifikace oblastí a) neuronová síť přednaučená b) neuronová síť učená při prohledávání oblasti

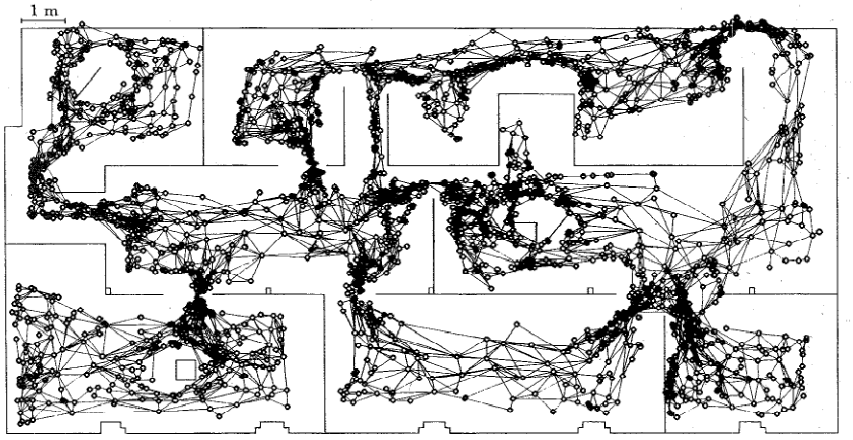
7 VÝSLEDKY

Výsledky se budou porovnávat pro všechny tři klasifikátory neuronovou síť přednaučenou, neuro fuzzy síť (nepřednaučenou) a neuronovou síť nepřednaučenou. Klasifikátory budou mít k dispozici stejná data. Výsledky budou porovnány na dvou mapách obrázek 7-1. Mapa b) na obrázku 7-1 byla použita v článku A. Kurze [19] a v článku [12] obrázek 7-2. Topologická mapa na obrázku 7-2 byla vytvořena pomocí neuronové sítě. Robot v tomto případě nejdříve prostředí náhodně projížděl a učil se klasifikovat oblasti. Po 10000 měření, bylo učení ukončeno. Robot znovu prostředí náhodně projížděl ale tentokrát už jenom klasifikoval. Ukončení prozkoumávání prostředí bylo dáno počtem iterací (10000).

Robot v simulaci před prozkoumáváním prostředí nemá žádnou informaci o daném prostředí, ví pouze, na jakých souřadnicích se nachází. Klasifikace prostředí probíhá pouze na základě dat ze senzorů.

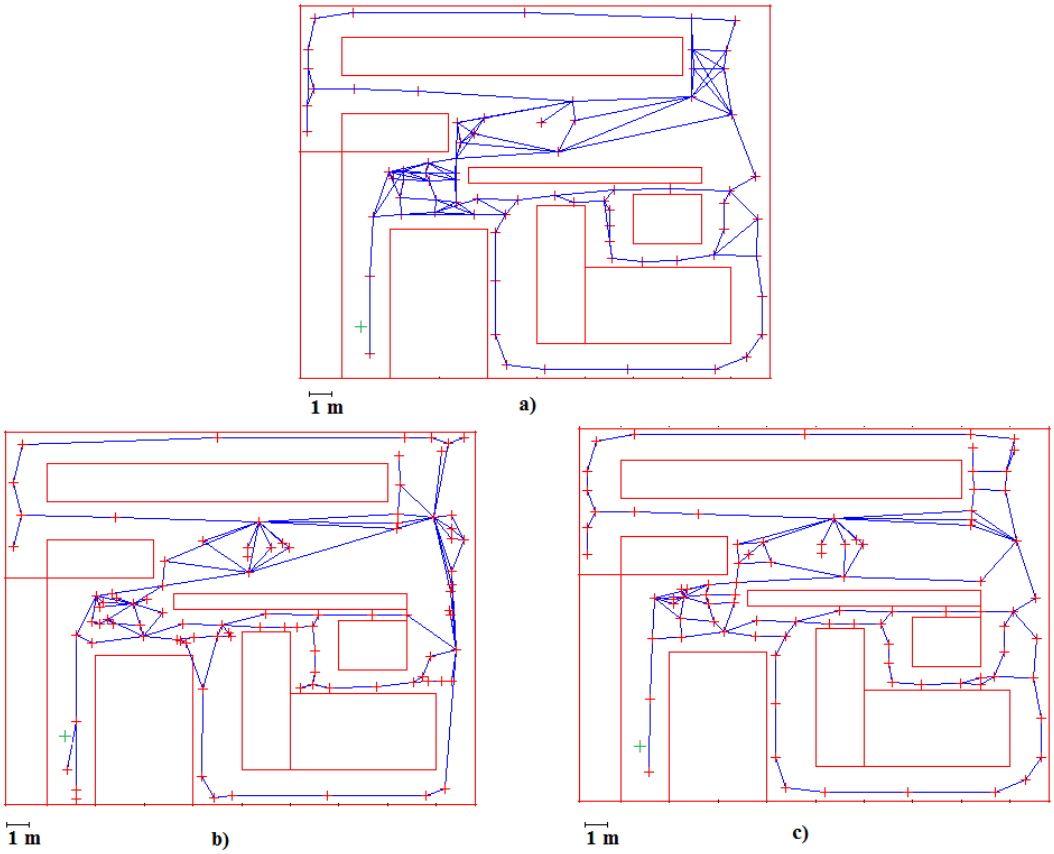


7-1 Testovací mapy v simulaci



7-2 Topologická mapa použitá v [19]

Jako měřítko pro srovnání vytvořených topologických map všech tří klasifikátorů se použije délka trasy z bodu A do bodu B. Bude zadáno náhodně 1000 měření (vygenerování 1000 startovních bodů a vygenerování 1000 cílových bodů). Délky tras na všech třech mapách budou sečteny. Minimální hodnota bude vyhodnocena jako nejlepší, protože robot při cestování v prostředí ujede nejmenší vzdálenost.



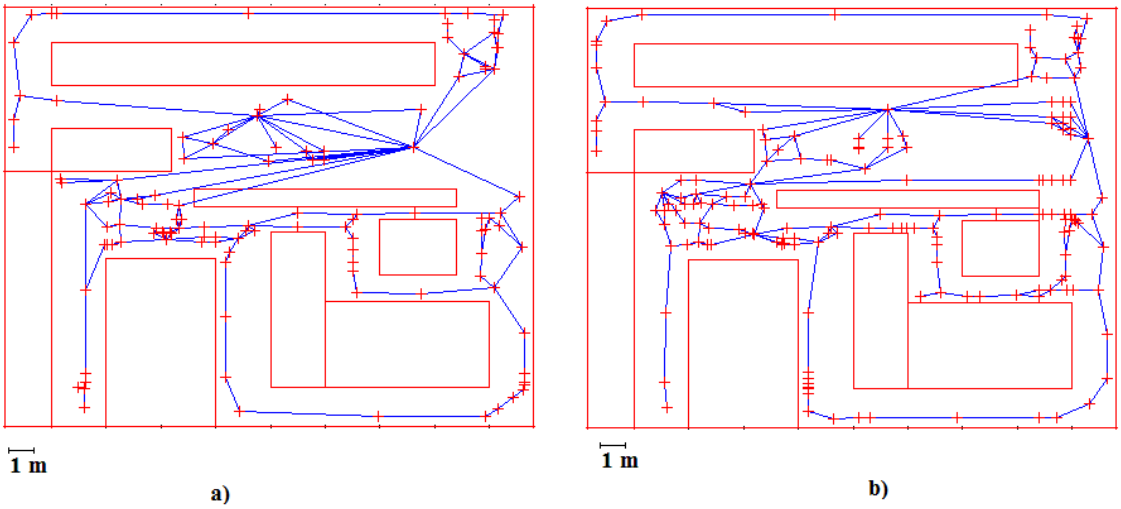
7-3 Topologické mapy simulace a) Přednaučený klasifikátor b) Neuro fuzzy c) Kohonenova neuronová síť

Na obrázku 7-3 jsou znázorněny topologické mapy, které vznikly v simulačním programu v rámci diplomové práce. Na obrázku 7-3 jsou uzly topologické mapy znázorněny červeným křížkem, modré úsečky tvoří spojení mezi jednotlivými uzly, červeně jsou vyznačeny překážky, zelený křížek značí startovní pozici robota. Robot prozkoumal prostředí pomocí řídícího algoritmu, který po prohledání všech oblastí a správné vzájemné poloze všech uzlů na mapě ukončil prohledávání. Celková ujetá dráha robot při klasifikaci prostředí byla 214 m. Mapa na obrázku 7-3 a) se tvořila v průběhu prozkoumávání neznámé oblasti (přímá tvorba mapy), mapy b) a c) byly vytvořeny po ukončení prohledávání (nepřímá tvorba mapy). Konfigurace všech tří klasifikátorů jsou v tabulce 1. Neuro fuzzy síť obrázek 7-3 b) a Neuronová síť c) byly učeny při náhodném projíždění neznámého prostředí celkem 10 000 vzorky.

	Neuronová síť přednaučená	Neuro fuzzy síť (nepřednaučená)	Neuronová síť (nepřednaučená)
Počet neuronů	15	16	16

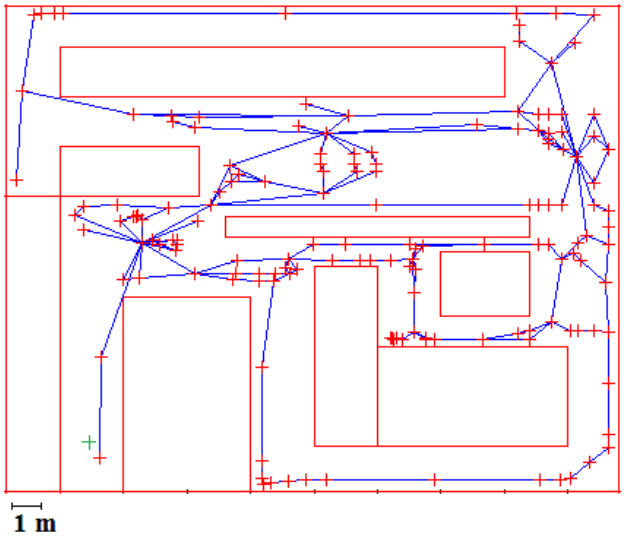
Tabulka 1 Konfigurace klasifikátorů

Na obrázku 7-4 je vidět topologická mapa vytvořená pomocí neuro fuzzy klasifikátoru s 25 a 50 neurony. Neuro fuzzy síť se učila při náhodném projíždění neznámého prostředí, počet učících dat byl 10 000.



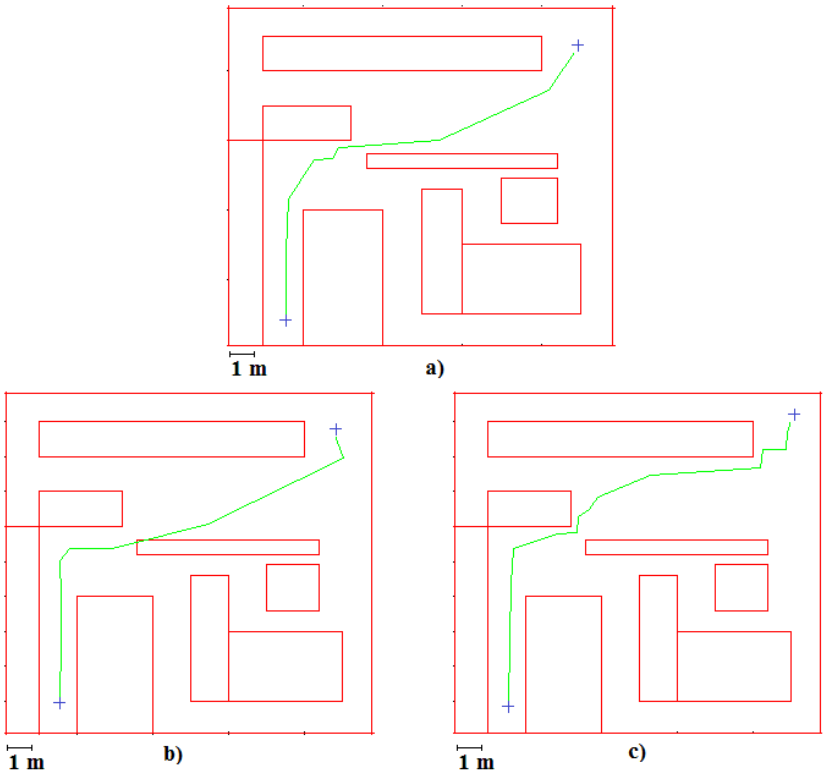
7-4 Topologická mapa neuro (fuzzy síť) a) 25 neuronů b) 50 neuronů

Na obrázku 7-5 je zobrazena topologická mapa, kde byl použit neuro fuzzy klasifikátor. Neuro fuzzy klasifikátor se učil v průběhu prozkoumávání neznámého prostředí. Prostředí prozkoumával pomocí řídícího algoritmu kapitola 6.2.2. V průběhu učení neuro fuzzy sítě bylo vytvořeno celkem 79 neuronů.



7-5 Topologická mapa neuro fuzzy učení řidící algoritmus

Na obrázku 7-6 a) je navigace na topologické mapě vytvořené pomocí přednaučené neuronové sítě, obrázek 7-6 b) je navigace na topologické mapě vytvořené pomocí neuro fuzzy sítě (16 neuronů) a obrázek 7-6 c) navigace na topologické mapě vytvořené pomocí neuronové sítě (16 neuronů). Zelená křivka na obrázku značí cestu robota, kudy by měl jet. Cesta je vytvořena ze spojnic uzlů na topologické mapě tak, aby cesta byla co nejkratší. Modré křížky značí startovní a cílový bod.



7-6 Navigace na topologické mapě

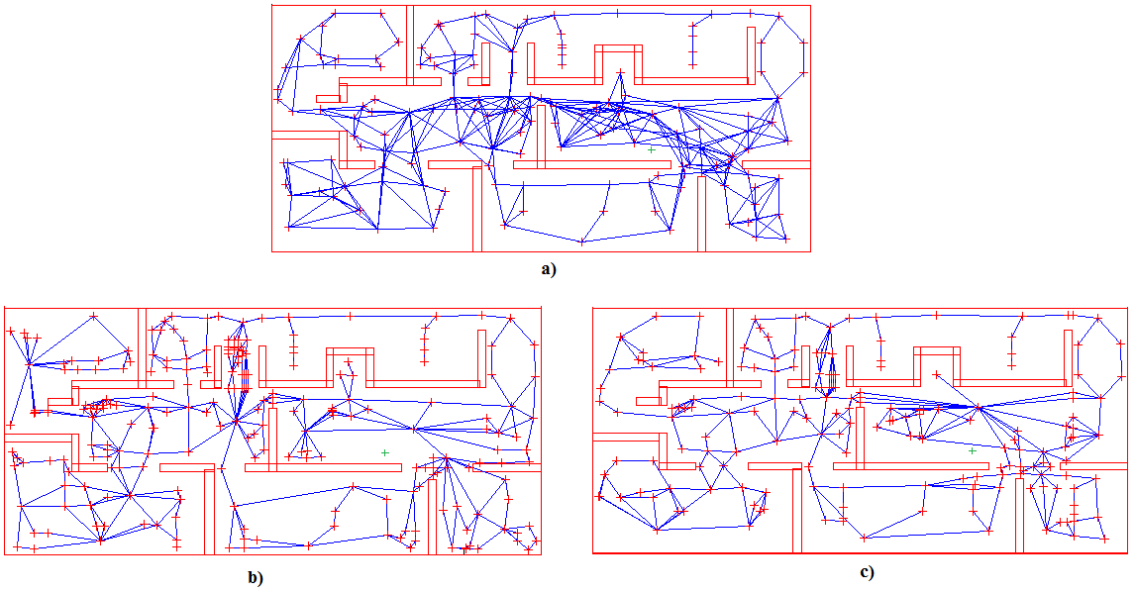
V simulacích bylo odzkoušeno několik různých startovních pozic pro tvorbu topologické mapy. Mapy byly takřka vždy totožné nebo podobné, právě z důvodu použitého řídícího algoritmu prohledávání neznámého prostředí.

Pro porovnání map byla použita vzdálenost, kterou by musel robot podle jednotlivých map ujet, aby se dostal k cíli. Náhodně bylo zadáno 1000 startovních a 1000 cílových bodů. Na topologických mapách se našla nejkratší vzdálenost v grafu pomocí dijkstrova algoritmu. Výsledky pro různé startovní pozice (x;y) při tvorbě mapy ukazuje tabulka 2

	Neuronová síť přednaučená	Neuro fuzzy síť nepřednaučená	Neuronová síť nepřednaučená
Délka cesty [m] (70;50)	26075	26706	26389
Délka cesty [m] (450;450)	24903	25346	25320

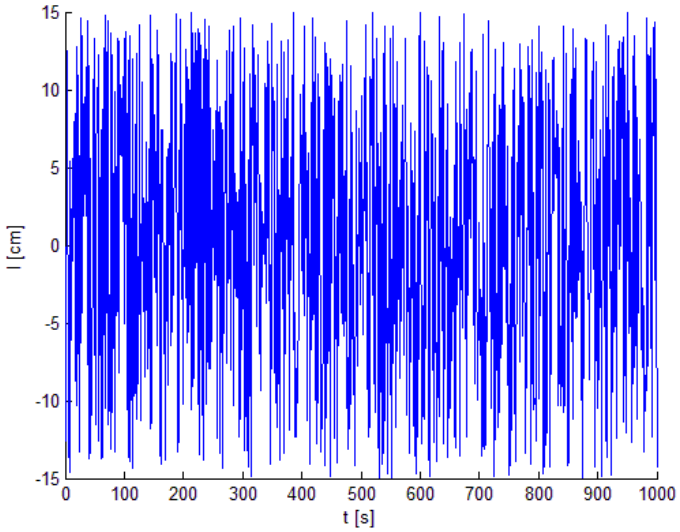
Tabulka 2 Výsledky ujetá vzdálenost při 1000 náhodných bodech

Na obrázku 7-7 je mapa překážek, která byla použita v článku [19]. Topologická mapa a) byla vytvořena pomocí neuronové sítě, která byla přednaučena, tvořila se přímo při prozkoumávání neznámé oblasti, mapa b) byla vytvořena pomocí neuro fuzzy sítě šestnácti neurony, síť byla učena při náhodném projíždění neznámé oblasti. Mapa c) byla vytvořena pomocí neuronové sítě, síť je tvořena šestnácti neurony, učení sítě probíhalo při náhodném projíždění neznámé oblasti. Náhodné projíždění neznámé oblasti za účelem učení neuro fuzzy sítě a neuronové sítě (nepřednaučené) bylo vždy provedeno před samotnou tvorbou topologické mapy a klasifikací prostředí. Zelený křížek značí pozici, kde robot začal prozkoumávat neznámé prostředí a tvořit topologickou mapu. Červené křížky značí uzly topologické mapy, modré křivky jsou spoje mezi jednotlivými uzly mapy. Červeně jsou označeny překážky.



7-7 Mapa prostředí převzatá z [19]

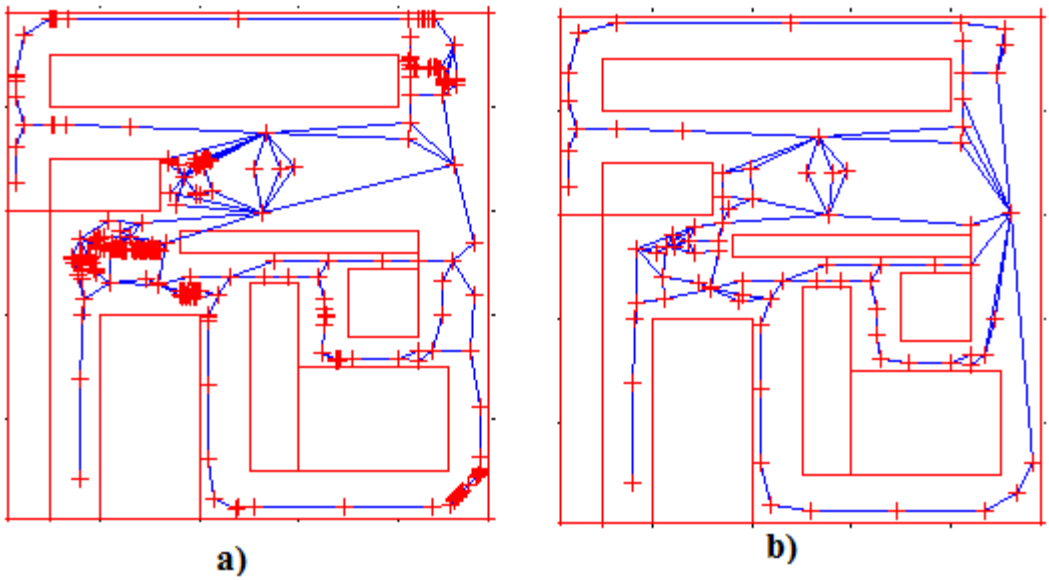
Při klasifikaci a učení byl vyzkoušen vliv **zašumění senzorů**. Data ze snímačů byla náhodně zašuměna $\pm 7{,}5$ % od své hodnoty obrázek 7-8.



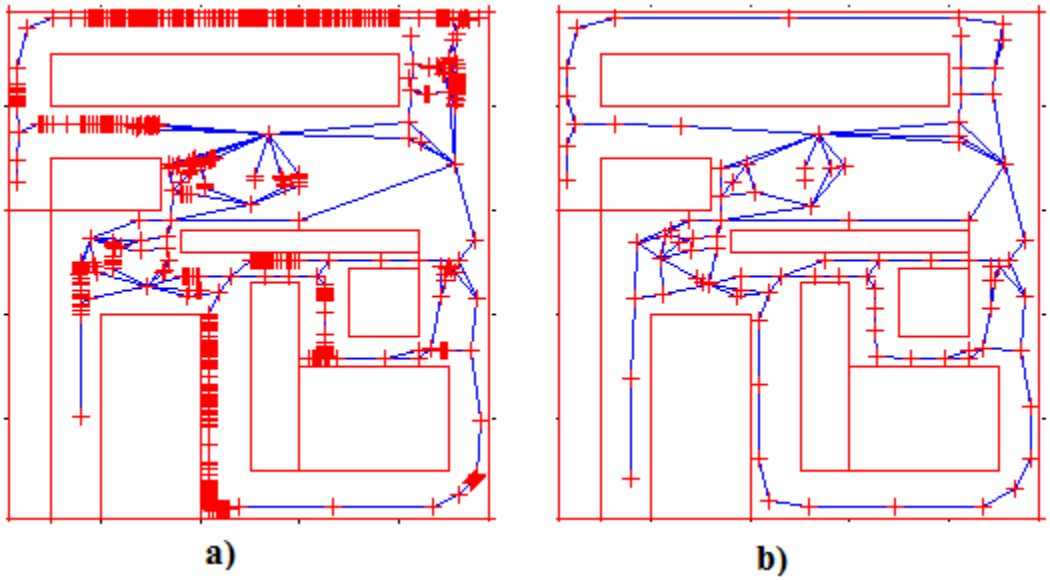
7-8 Zašumění snímačů pro 2 m

Zašuměná data byla testována na neuro fuzzy síti a na neuronové síti přednaučené. Obrázek 7-9 ukazuje vytvořené topologické mapy a) pomocí neuro fuzzy b) neuronové sítě nepřednaučené. Neuro fuzzy síť i neuronová síť nepřednaučená byly trénovány 20000 daty při náhodném projíždění neznámého prostředí.

Zde se ukázal vliv počtu trénovacích dat na naučení neuro fuzzy sítě. Na obrázku 7-9 a) je topologická mapa vytvořená pomocí neuro fuzzy klasifikátoru, který byl učen 20000 daty. Na obrázku 7-10 a) je neuronová síť učená pouze 10000 daty. Na topologické mapě vzniká hodně uzlů vlivem zašumění dat ze snímačů.



7-9 Topologická mapa počet trénovacích dat 20000 a) neuro fuzzy b) neronová síť nepřednaučená



7-10 Topologická mapa počet trénovacích dat 1000 a) neuro fuzzy b) neuronová síť nepřednaučená

8 SOFTWARE

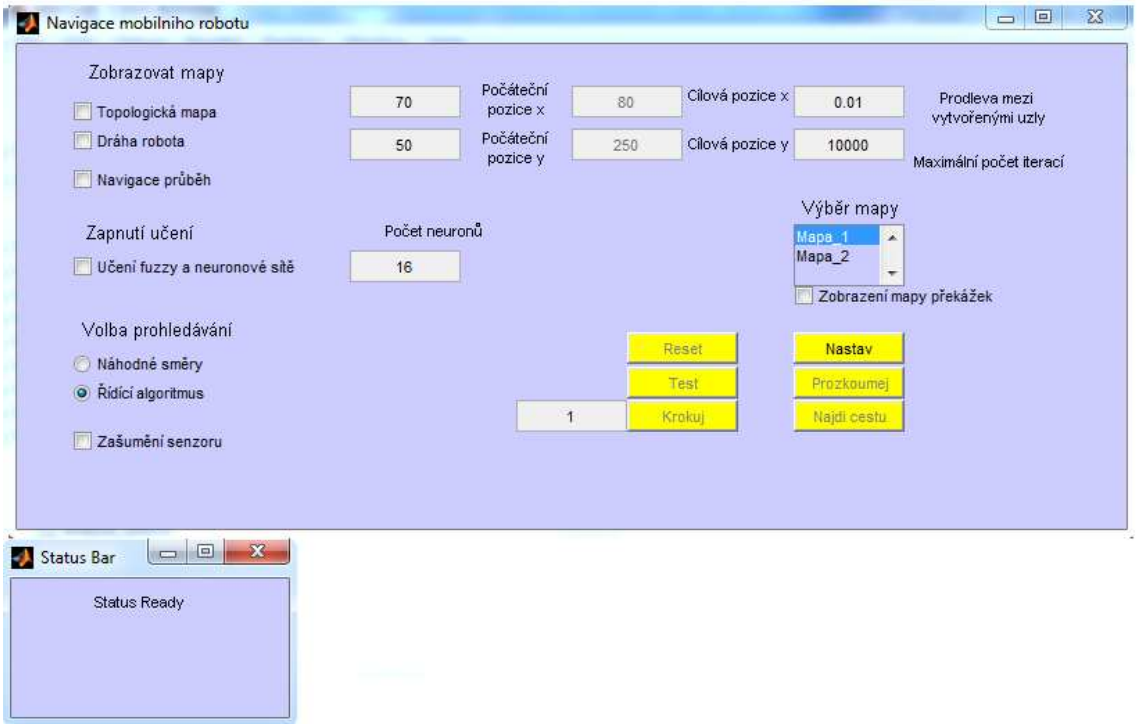
Program byl vytvořen v prostředí Matlab 2010a. Byl testovaný na 64-bit verzi i na 32-bit verzi. Pro spuštění napíšeme příkaz: *diplomova_prace();*

Simulační program obsahuje dvě základní okna: **Navigace mobilního robota** a **Status bar**. V okně Navigace mobilního robota nastavujeme parametry: počáteční pozice, maximální počet iterací, maximální počet operací s uzly, typ prohledávání oblastí, zašumění senzorů, dále si zde můžeme zvolit mapu, kterou bude robot prozkoumávat. Informace o tom co robot právě dělá zobrazuje Status bar.

V průběhu prozkoumávání prostředí si můžeme nechat zobrazovat: dráhu robota, přímo tvořenou topologickou. Před spuštěním simulace musíme ještě zvolit způsob prohledávání prostředí: pomocí řídícího algoritmu nebo náhodné prohledávání). Při volbě náhodné prohledávání robot ukončí prohledávání buď po dosažení maximálního počtu iterací nebo po dosažení maximálního zadaného počtu kroků (operace s uzly).

Za předpokladu, že budeme chtít robota učit v průběhu prozkoumávání prostředí zaškrtneme **učení fuzzy neuronové sítě**, dále závisí na jakou volbu prohledávání neznámé oblasti zvolíme: náhodné, pomocí řídícího algoritmu. Kdybychom zvolili náhodné prohledávání neznámé oblasti zvolíme i pevně počet neuronů, který bude mít neuro fuzzy síť, ale kdyby jsme zvolili prohledávání pomocí řídícího algoritmu, neuronová síť by začínala s jedním neuronem a další by se přidávali v průběhu učení. kapitola 6.2.1.1.

Program si můžeme buď krokovat (tlačítko Krokuj) po jednotlivých krocích (jeden krok je jedna operace s uzle - přidání uzlu, odebrání), nebo nechat robota rovnou prozkoumat celé prostředí (tlačítko Prozkoumej). V případě, že bychom chtěli sledovat průběh vytváření topologické mapy, můžeme si průběh zrychlit nebo zpomalit nastavením položky **Prodleva mezi vytvořenými uzly**.



8-1 Simulační program

Po prozkoumání celého prostředí dojde ke zpracování ostatních topologických map z uložených údajů klasifikátorů neuro fuzzy a neuronové sítě (nepřenadučené). Pro cesty kudy by robot jel podle jednotlivých topologických map zadáme počáteční a cílové pozice a stiskneme tlačítko **Najdi cestu**.

9 ZÁVĚR

Cílem diplomové práce bylo seznámit se s možnostmi využití fuzzy logiky při tvorbě mapy pro mobilního robota a jeho navigace na mapě. Největší pozornost byla soustředěna na topologické mapy a na různé metody tvorby těchto map. Navržené metody vycházejí z neuronových sítí a neuro fuzzy sítí. Dále byl v práci navržen algoritmus pro prohledání neznámého prostředí pro autonomního mobilního robota. V prostředí Matlab byl navržen simulátor, který simuluje pohyb robota v neznámém prostředí.

Navržené metody pro tvorbu topologické mapy, byli rozděleny do třech částí. První navržená metoda byla klasifikace oblastí pomocí neuronové sítě. Tato metoda byla vylepšena o přednaučení neuronové sítě, čímž se ušetřila doba nutná pro tvorbu topologické mapy. Dále byl k této metodě navržen řídící algoritmus pro autonomní prohledávání neznámého prostředí, který posuzoval mapu podle vzájemné polohy uzlů topologické mapy. Po zmapování prostředí a dosažení správné vzájemné polohy všech uzlů sám ukončí mapování. Použitá neuronová síť spolu s řídícím algoritmem pro prohledávání neznámého prostředí má výhodu v malé paměťové náročnosti a rychlosti zmapování neznámého prostředí. Druhá metoda využívala neuro fuzzy síť, k tvorbě topologické mapy. U neuro fuzzy sítě bylo vyzkoušeno přednaučení také, ale výsledky byly hodně podobné. Z důvodu vyšší výpočetní náročnosti neuro fuzzy sítě nebyla nakonec neuro fuzzy síť přednaučena. Učení pro neuro fuzzy síť bylo rozděleno do dvou částí: robot projíždí prostředí náhodně, robot projíždí prostředí pomocí řídícího algoritmu. Učení při projíždění prostředí pomocí řídícího algoritmu nedávalo dobré výsledky, protože data ze senzorů byla málo různorodá. Algoritmus učení byl upraven tak, že síť začíná s jedním neuronem a v průběhu projíždění jsou dynamicky neurony přidávány a učeny na aktuální prostředí. Tento způsob má nevýhodu ve vytvoření velkého množství neuronů, některé neurony nebyly použity pro klasifikaci. Učení při projíždění neznámé oblasti náhodně se neupravovalo, data při tomto projíždění byla dostatečně různorodá a neuro fuzzy síť se dobře učila. Třetí metoda využívá neuronovou síť o pevném počtu neuronů, která se učí při projíždění neznámého prostředí, pro oba způsoby prohledávání prostředí se učí stejně. Po naučení neuronové sítě se ukázalo, že přednaučená neuronová síť a neuronová síť, která se učila během prohledávání neznámého prostředí, klasifikují obě sítě takřka shodně. Síť se projížděním neznámého prostředí, naučila podobně jako přednaučená neuronová síť.

V simulaci bylo vyzkoušeno zašumění dat ze senzorů. Zde se ukázal vliv počtu učících dat při učení neuro fuzzy sítě. Při použití 10000 dat pro učení, byla neuro fuzzy síť málo odolná vůči šumu, na topologické mapě se vytvářelo velké množství uzlů. Při

použití dvojnásobného množství dat 20000 pro učení, se na topologické mapě takřka nevyskytovali žádné uzly navíc.

Kritériem pro vybrání nejlepší topologické mapy byla ujetá dráha robota při navigaci na mapách. Bylo zadáno 1000 startovní a cílových bodů a délka drah na jednotlivých mapách sečtena. Výsledky pro všechny tři metody byli hodně podobné. Nejlepší výsledek měla vždy topologická mapa vytvořená pomocí přednaučené neuronové sítě.

Literatura

[1] **Jiřina, Marcel.** České vysoké učení technické v Praze Fakulta Biomedicínského inženýrstvý. *Neuronové sítě.* [Online] [Citace: 29. 04 2011.] www.fbmi.cvut.cz/e/prezentace-o-neuronovych-sitich/1565.ppt.

[2] **RNDr. Petr Štěpán, Ph.D.** Mobilní robotika ČVUT. *Modely prostředí II.* [Online] 11. Březen 2008. [Citace: 3. Duben 2011.] https://cw.felk.cvut.cz/lib/exe/fetch.php/mkr/lessons/mapovani-ii.pdf.

[3] **Honathan Dixon, Oliver Henlich.** Mobile robot Navigation. [Online] 10. Červen 1997. [Citace: 13. Duben 2011.] http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol4/jmd/.

[4] wikipedia. *Global Positioning System.* [Online] 10. Květen 2011. [Citace: 11. Květen 2011.] http://cs.wikipedia.org/wiki/Global_Positioning_System.

[5] **bergmann.** CE4YOU. *Co to je GPS? Historie a úvod do problematiky.* [Online] 12. Prosinec 2005. [Citace: bergmann13. Leden 2011.] http://www.ce4you.cz/articles/detail.asp?a=244.

[6] **Dlouhý, Martin.** Robotika. *Bug algoritmy.* [Online] 07. Listopad 2005. [Citace: 20. Březen 2011.] http://robotika.cz/guide/bug-alg/cs.

[7] **prof. Ing Pavel Jura, CSc.** *Základy FUZZY logiky pro řízení a modelování.* Brno : PC-DIR spol. s.r.o.

[8] wikipedia. *Perceptron.* [Online] 21. Duben 2011. [Citace: 24. Leden 2011.] http://cs.wikipedia.org/wiki/Perceptron.

[9] Pražské metro. *Plán pražského metra.* [Online] 31. Června 2006. [Citace: 25. Listopad 2010.] http://metro.zarohem.cz/mapa.html.

[10] **Giuseppe Oriolo, Giovanni Ulivi, Marilena Vendittelli.** CiteSeerX. *Fuzzy Maps: A new tool for mobile robot perception and planning.* [Online] 1997. [Citace: 1. Květen 2010.] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.40.4703&rep=rep1&type=pdf.

[11] **Omar M. Al-Jarrah, Omar Q. Bani-Melhem.** IOS Press. *Building maps for mobile robot navigation using fuzzy classification of ultrasonic range.* [Online] Listopad 2001. [Citace: 21. Květen 2010.] http://iospress.metapress.com/content/yaffmab1fmedtg6g/.

[12] *Building maps based on a learned classification of ultrasonic range data.* **A.Kurz.** Southampton, UK : International Federation of automatic control, 1993. IFAC Workshop.

[13] MIT Conputer science and artificial intelligence laboratory. *Robo-Rats Locomotion: Synchro Drive.* [Online] 4. Duben 2001. [Citace: 18. Duben 2011.] http://groups.csail.mit.edu/drl/courses/cs54-2001s/synchro.html.

[14] Robotic workshop. *Synchro Drive.* [Online] [Citace: 2. Květen 2011.] http://www.convict.lu/Jeunes/SynchroDrive.htm.
[15] **ŠVOMA, STANISLAV.** Bakalářská práce VUT v Brně Fakulta strojního inženýrství. *AD INVERTOR - 3D MODELOVÁNÍ PODVOZKY ROBOTU O3-OSNERA.* Brno : autor neznámý, 2010.
[16] **Wesley H. Huang, Kristopher R. Beevers.** CiteSeerx. *Topological Mapping with Sensing-Limited Robots.* [Online] 2006. [Citace: 25. Září 2010.] http://www.cs.rpi.edu/~beevek/research/mapping_wafr04.pdf.
[17] **Černý, Jakub.** Základní grafové algoritmy. *Nejkratší cesta v grafu.* [Online] 24. Listopad 2010. [Citace: 19. Duben 2011.] http://kam.mff.cuni.cz/~kuba/ka/.
[18] **Kurz, Andreas.** IEEE. *Constructing Maps for Mobile Robot Navigation Based on Ultrasonic Range Data.* [Online] 2. Duben 1996. [Citace: 25. Březen 2010.] http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=00485835.
[19] **Miller, David P. a Koushik, Prabhakar M.** Virginia Polytechnic Institute and State University. *Low Error Path Planning for a Synchro Drive Mobile Robot.* [Online] [Citace: 12. Březen 2011.] http://eprints.cs.vt.edu/archive/00000035/01/TR-86-28.pdf.
[20] **Al-Jarrah, Omar M. a Bani-Melhem, Omar Q.** ACM Digital Library. *Journal of Intellignet & Fuzzy System.* [Online] 2001. [Citace: 2. 12 2010.] http://portal.acm.org/citation.cfm?id=1314138.
[21] ČVUT. *Neuronové sítě.* [Online] [Citace: 5. 15 2011.] http://cyber.felk.cvut.cz/gerstner/biolab/bio_web/teach/FunBio/neuron/neursite.html.
[22] **Hlavác, Václav.** Fakulta elektrotechnická ČVUT v Praze. *UČENÍ BEZ UČITELE.* [Online] [Citace: 10. Květen 2011.] http://cmp.felk.cvut.cz/~hlavac/Public/TeachingLectures/UceniBezUcitele.pdf.
[23] **Vojáček, Antonín.** Západočeská univerzita v Plzni. *Samoučící se neuronová síť - SOM.* [Online] 14. Květen 2006. [Citace: 25. Duben 2011.] http://www.kiv.zcu.cz/studies/predmety/uir/NS/Samouc_NN2.pdf.

Seznam použitých zkratek a symbolů

μ koeficient učení

Seznam příloh

Příloha 1

Část kódu z řídícího algoritmu

```
case 10 %dolni slepa chodba
        if(length(x_main) <= 10)%prvni bod?
            smer = 0;  %natvrdo hozeny smer protoze je to prvni bod a my vime
jenom ze jsme nekde v chodbe
        else
            switch(neuron_main(end - 1))
                case 6    %dolni levy roh
                    if(((pozice(3) >= 0) || (pozice(3) >= 2*pi)) && (pozice(3)
<= pi))
                        smer =pi/2;
                    end
                case 7    % pravy dolni roh
                    smer = pi/2; %zmena smeru
                case 3
                    smer = pozice(3); %zachovani pozice
                case 13   % svila chodba
                    smer = pozice(3);
                case 10   % dolni slepa chodba
                    smer = pozice(3);

                    if((pozice(3) > 0) && (pozice(3) < pi))
                        smer = pi/2;
                    else
                        smer = 3*pi/2;
                    end

                    ma = find((senzor <= 5) & (senzor ~= 0));
                    [~, n] = size(ma);
                    if(n > 0)    %jsem blizko prekazky?
                        switch ma(1)   % v jakém smeru je prekazka
                            case 1
                                smer = pi +(2*pi-pi)*rand(1,1);
                            case 2
                                smer = 0 + (pi - 0)*rand(1,1);
                            case 3
                                smer = pi/2 + (3*pi/2 - pi/2)*rand(1,1);
                            case 4
                                if((pozice(3) > 0) && (pozice(3) < pi))
                                    smer = 0 + (pi/2 - 0)*rand(1,1);
                                else
                                    smer = 3*pi/2 + (2*pi - 3*pi/2)*rand(1,1);
                                end
                        end
                    end
                otherwise %neznama oblast
                    ma = find((senzor <= 5) & (senzor ~= 0));
                    [m n] = size(ma);
                    if(n > 0)
                        switch ma(1)
                            case 1
                                smer = pi +(2*pi-pi)*rand(1,1);
                            case 2
                                smer = 0 + (pi - 0)*rand(1,1);
                            case 3
                                smer = pi/2 + (3*pi/2 - pi/2)*rand(1,1);
                            case 4
                                smer = -pi/2 + (pi/2 - (-pi/2))*rand(1,1);
                        end
                    end
            end
        end
```