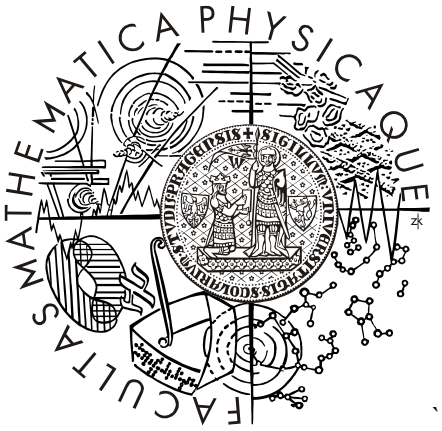
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Pavel Jiroutek

Porovnávání p irozeného a um lého u ení

Kabinet software a výuky informatiky
Vedoucí diplomové práce: RNDr. Franti ek Mráz, CSc.
Studijní obor: informatika – softwarové systémy

M j dík pat í v em, kte í mi pomohli s realizací této diplomové práce. P edev ím d kuji mému vedoucímu RNDr. Franti ku Mrázovi, CSc. za ochotu, cenné rady a p ipomínky. Tato práce by také nevznikla bez podpory Mgr. Jana Svobody z Fyziologického ústavu Akademie v d R, který mi poskytl data z oblasti u ení potkan . Dále bych rád pod koval v em ostatním kolem mne za trp livost a p ínosné konzultace.

Prohla uji, že jsem svou diplomovou práci napsal samostatn a výhradn s použitím citovaných pramen . Souhlasím se zap j ováním práce.

V Praze dne 7. 8. 2005

Pavel Jiroutek

Obsah

Název práce: Porovnávání p irozeného a um lého u ení

Autor: Pavel Jiroutek

Katedra: Kabinet software a výuky informatiky

Vedoucí diplomové práce: RNDr. Franti ek Mráz, CSc.

e-mail vedoucího: Frantisek.Mraz@mff.cuni.cz

Abstrakt:

Práce se zabývá návrhem a implementací evolu ního systému pro ízení autonomního mobilního robota. Tento systém by mu m l umož ovat adaptaci na t ídu úloh, kterou lze srovnateln definovat i pro skute ného živo icha, který je rovn ž schopen se úlohy z této t ídy nau it. Pro tyto ú ely jsou využity výsledky reálných pokus s u ením laboratorních potkan . Systém pro ízení robota je kombinací metod mobilní robotiky a um lé inteligence. Adaptabilní ást jeho ízení je založena na principu genetických algoritm a neuronových sítí. Krom r zných aspekt jednotlivých prvk systému, vlastního ízení robota a jeho realizace, jsou zde rozebrány i dal í nutné komponenty testovacího prost edí, ve kterém lze provád t evolu ní experimenty s reálnými roboty. Sou ástí práce jsou i výsledky praktických experiment s vytvo eným systémem a jejich porovnání s výsledky reálných test s potkany na podobných úlohách.

Klí ová slova: evolu ní robotika, neuronové sít , genetické algoritmy, reálné pokusy

Title: Comparison of natural and artificial learning

Author: Pavel Jiroutek

Department: Department of Software and Computer Science Education

Supervisor: RNDr. Franti ek Mráz, CSc.

Supervisor's e-mail address: Frantisek.Mraz@mff.cuni.cz

Abstract:

This paper deals with design and implementation of an evolutionary system for control of an autonomous mobile robot. This system should make possible an adaptation to a group of tasks, that can be similarly defined for a living being. We use results of real experiments with laboratory rats and tasks, which these rats are able to learn. The robot control system is a combination of several methods of mobile robotics and artificial intelligence. The adaptable part of the control system is based on genetic algorithms and neural networks. This work covers a wide range of problems related with this subject — various elements of the control system, the robot control and implementation, and also components of the test environment, which can be used to execute evolutionary experiments with real robots. The final part of this work includes results of our practical experiments with the robot and their comparison with real testing of rats on similar tasks.

Keywords: evolutionary robotics, neural networks, genetic algorithms, real experiments

1 Úvod

Mobilní robotika je moderní obor propojující n kolik v dních disciplín — informatiku, elektrotechniku, biologii i filozofii. Mobilní roboti se asto pohybují v dynamickém, i neznámém prost edí, což klade velké nároky na komplexnost jak návrhu, tak hlavn ízení. Pro ízení robot existují a vyvíjí se rozli né metody, li ící se svou výpo etní složitostí, p enositelností mezi r znými úlohami i úsp ností e ení zadání.

Mezi t mito jsou i metody založené na evoluci a u ení. Genetické algoritmy, jako metoda evoluce um lých systém , se v této oblasti hodí pro vývoj komplexn j ího chování bez p ímé informace o ohodnocení každého z jeho krok . Tedy nap íklad pro vývoj strategie e ící úkol, na který robot není p ímo naprogramován. Neuronové sít , tedy metodu pro strojové u ení, lze nap íklad použít pro postihnutí složitých závislostí mezi vstupy motor a výstupy senzor . Široké pole nabízí i r zná propojení t chto metod s ostatními, i spolu navzájem, nap íklad p i genetickém vývoji vah, nebo architektury neuronové sít , která pak generuje p íkazy strategie. Je t o úrove vý e je problém vývoje kooperativních strategií pro skupiny robot , i autonomní jedince ve skupin , což je dal í velice zajímavá a stále je t otev ená oblast evolu ní robotiky.

Jak genetické algoritmy, tak i neuronové sít jsou metody, jejichž vznik byl motivován zkoumáním vývoje a u ení živých tvor v p írod a následné poznatky byly aplikovány ve sv t um lé inteligence. Zde se uplat ují ve stále v t ím m ítku i v oblastech, které s p vodními úlohami, pro které se tyto postupy uplat ují v p írod , souvisí asto velice vzdálen .

Cílem této diplomové práce je navrhnout úlohu i t ídu úloh, kterou lze dob e definovat jak pro jednoduché živo ichy (konkrétn laboratorní potkany), tak i pro mobilní roboty. Poté navrhnout a implementovat ízení robota za pomoci metod evolu ní robotiky tak, aby reálný robot tuto úlohu e il s informacemi o úloze takovými, o nichž se dá p edpokládat, že je má i skute ný živo ich. S tím souvisí i návrh a implementace softwarového a hardwarového prost edí, ve kterém by bylo možné tyto testy realizovat. Dal ím úkolem bude analýza a srovnání výsledk pokus na reálném živo ichovi a mobilním robotovi.

Obsahem této práce není provád ní pokus s potkany. Pro porovnávání jejich výsledk p i u ení úloh jsme využili záznam o již provedených pokusech. Cílem není ani simulace proces , o nichž se p edpokládá, že probíhají v mozku potkana [11] . Spojitost této práce s u ením potkan je ve zp sobu zadávání úlohy pro roboty, v typu úlohy a informaci o ní b hem u ení. Problematika u ení laboratorních potkan je podrobn zpracována v [9] a [10] .

Úloha, na které budeme testování provád t, a její modifikace pro robota a potkana je blíže popsaná v kapitole 2. Kapitola 3 stru n popisuje základní metody evolu ní robotiky, které jsou v této práci použity. Konkrétn obsahuje úvod do genetických algoritm , um lých neuronových sítí a dále zmi uje základní aspekty mobilní robotiky. Ve 4. kapitole jsou popsány jednotlivé kroky, jakými se robot danou úlohu nau í. Implementace t chto krok je blíže rozepsána v kapitole 5. Tato

kapitola pojednává i o architektu e a hardwarové a softwarové implementaci mobilního robota a testovacího prost edí. V kapitole 6 je porovnání zp sobu e ení úlohy potkanem a robotem a dal í testy adaptabilní strategie robota. Kapitola 7 shrnuje dosažené výsledky této práce.

2 Specifikace zadání

2.1 P vodní zadání

Cílem práce je hledání podobností a odli ností mezi zp sobem u ení živo ich a robot . Na základ reálných pokus s u ením živo ich (laboratorních potkan) na jednoduchých úlohách navrhnout analogické úlohy pro mobilního robota, navrhnout metody u ení takových úloh a implementovat je. Implementované metody vyzkou et na simulátoru a potom i na reálném robotovi. Navržený systém by m l umožnit co nejv t í ást (asov velmi náro ného) u ení robota realizovat v simulátoru a výsledné programy potom p enést do reálného robota a ídící program „doladit" na konkrétním hardware. Na základ experiment porovnat u ení robot a živo ich .

2.2 Bliž í ur ení zkoumané úlohy

V této kapitole jsou popsány základní požadavky na úlohu, na které chceme ná výzkum provád t. Dále obsahuje rozbor variant, které lze použít p i zprost edkování informace o úloze u ícímu se robotovi. V poslední ásti je popsána konkrétní úloha, kterou jsme nakonec pro porovnávání robota s potkanem zvolili.

2.2.1 Obecné požadavky

Na ím cílem bylo porovnávání u ení laboratorního potkana s robotem. Motivací pro výb r potkan jako živo ich , s nimiž budeme robota porovnávat, byla možnost spolupráce s Fyziologickým ústavem Akademie v d, kde mají pokusy s u ením potkan už jistou tradici, a také to, že úlohy, které se potkani u í, jsou pom rn p ímo a e popsatelné i pro sv t robot . Nejprve bylo t eba vybrat testovací úlohu. Na její výb r jsme m li následující požadavky:

- Úloha musí být dob e popsatelná pro potkana i robota pokud možno tak, aby žádného jedince p ímo nezvýhod ovala.
- Testovací prost edí pro robota musí být realizovatelné v na ich podmínkách, tzn. relativn jednoduché.
- Chceme se vyhnout provád ní nových pokus s potkany, proto musí jít o jednu z úloh, jejíž výsledky na potkanech jsou již k dispozici.
- Úloha musí být snadno modifikovatelná, aby bylo možné ov it, že je robot schopný nau it se i její varianty a jeho ízení není jednoú elov zam ené jen na jeden typ úkolu.

Aby bylo porovnání potkana a robota možné a jeho výsledky zajímavé, m li jsme dále následující požadavky na robota:

- Robot musí mít pokud možno o prost edí, v n mž u ení probíhá, stejné informace jako potkan. Cílem práce ov em nebylo navrhnout robota se senzory, které by v rn napodobovaly smysly potkana p i e ení úlohy (zrak, hmat, sluch...), ale spí e navrhnout ízení robota tak, aby, kdyby tyto

senzory m l, dokázal úlohu e it. Omezili jsme se proto na to, že testovací prost edí zprost edkovává robotovi takové informace, které by o n m m l mít i potkan. Tyto informace dále up esníme v dal ím textu. Samo prost edí je získává zpracováním obrazu z kamery, která test sleduje.

- Robot je v pr b hu pokusu, stejn jako potkan, pln autonomní, tj. robot není žádným zp sobem ízen z ven í a obsluha by ani nem la ovliv ovat výsledek procesu robotova u ení. Tedy, ve keré innosti, které v pr b hu testu robot provede, by m ly být d sledkem jeho strategie a úkolem obsluhy by m lo být pouze iniciovat jednotlivé kroky, které je nutné v pr b hu procesu u ení projít.
- Potkan je p i svém u ení omezen i asov . Jeden pokus, kdy se potkan pohybuje v prost edí a u í se úlohu, trvá 20 minut, p i emž asi po 20 takových pokusech je zdravý jedinec schopný na i úlohu efektivn e it. Stanovili jsme si toto asové omezení i pro robota. Jeho pobyt v testovacím prost edí nesmí být del í, než má k dispozici potkan. Toto omezení bylo ur eno i kv li tomu, že robot nebyl p vodn navržen na mnohahodinové genetické testy v reálu. Genetická evoluce v reálném prost edí je tedy tímto vylou ena.

2.2.2 Jak zadat úlohu robotovi?

Máme tedy autonomního robota, který má podobné vjemy o okolním prost edí jako potkan. Otázkou z stává, jak zprost edkovat robotovi informaci o úloze, kterou má v tomto prost edí e it. Potkan získává informaci o úloze typicky systémem odm n, resp. trest , které v pr b hu testu dostává v p ípad , že plní danou úlohu správn , resp. patn . K zadanému úkolu lze p istupovat v zásad v t chto t ech úrovních nezávislosti inteligence robota na znalosti úkolu:

Robot zná úlohu, ale nezná e ení

Jedná se o úrove s nejv t í informací. Robotova inteligence zná prost edí, ve kterém se bude úlohu u it, a zná i reakci prost edí na svoje chování — v každé situaci dokáže p edem posoudit, bude-li následovat odm na, nebo trest. Robot ale nemá explicitn zadané, jak má úlohu e it. Celou úlohu lze tedy nasimulovat v softwarovém simulátoru, ve kterém m že prob hnout i rozhodující ást u ení robota. Simulovaný robot se nau í vhodnou strategii pro e ení úkolu. Problém je zde s p enesením na robota reálného, který se m že chovat za stejných podmínek odli n než robot simulovaný[*].

[*] Tento problém je typický pro celou oblast evolu ní robotiky a nevyhnou se mu ani dal í zde popisované úrovn inteligence. Proto zde nastíníme základní p ístupy k jeho e ení:

- Je t eba mít co nejkvalitn j í simulátor, který co možná nejv rn ji nahrazuje chování reálného robota.
- Strategie robota pro e ení úkolu musí být co nejrobustn j í. Nesmí spoléhat na exaktnost um lého prost edí v simulátoru.
- Robota je v ur ité fázi t eba p estat u it v simulátoru a pokra ovat ve vývoji v reálném prost edí.

My jsme použili kombinaci prvních dvou postup . Na ím cílem bylo od za átku omezit reálné jízdy robota na minimum, kv li jejich asové a hlavn hardwarové náro nosti.

Reálných jízd se dá v tomto p ípad využít k nasbírání informací o pohybu reálného robota a na základ t chto informací pak zkalibrovat simulátor, aby co nejvíce odpovídal skute nosti.

Robot nezná úlohu ani e ení, ale zná zú astn né elementy úlohy

V této variant má robot o n co mén informací, protože p edem nezná reakci prost edí na své chování. P ed tím, než do prost edí skute n vstoupí, tedy neví, jak se bude chovat, za co bude následovat odm na a za co trest. Krom kalibrace simulátoru musí tedy robot p i reálných jízdách odpozorovat i to, jak se chová prost edí a za jakých okolností je vyvolána odm na i trest. Robot ale p edem zná konfiguraci prost edí — ví tedy, jak vypadá mapa prost edí a zná a dokáže odli it v echny elementy, které se v n m mohou pohybovat.

Tato varianta je p íhodn j í pro porovnávání chování robota s potkanem, protože potkan p edem také neví nic o tom, co se od n j vlastn o ekává. Dá se ale p edpokládat, že si je schopen pom rn brzy uv domit konfiguraci jednoduché mapy (v na em p ípad kruhové arénky) a existenci dal ích zajímavých objekt v této map (v na em p ípad je t jeden tvor).

Robot navržený pro tuto variantu by m l být teoreticky schopen nau it se v echny úlohy, které se v daném prost edí dají definovat.

Robot nezná úlohu, její e ení, ani prost edí

V této variant nezná robot reakci prost edí, stejn jako ve variant p edchozí, neví ale ani nic o tom, jak prost edí vypadá. V e musí sám zjistit a odli it pomocí vlastních senzor . Na této úrovni je robot, co se týká autonomie, naprosto ekvivalentní potkanovi — je umíst n do neznámého prost edí a má plnit neznámý úkol. Oproti potkanovi má tu nevýhodu, že se musí nau it rozpoznávat i zú astn né elementy v prost edí. Dá se p edpokládat, že toto umí potkan sám od sebe.

Na ím cílem bude návrh robota pro druhou úrove , tedy kdy robot nezná úlohu ani e ení, ale zná zú astn né elementy. Tato varianta nechává dostate ný prostor pro otestování kvality metod u ení robota a zárove se dob e hodí pro na e abstrahování od senzorických problém robotiky, kdy chceme robotovi zprost edkovávat informace o konfiguraci prost edí z venku, aniž by musel cokoliv zpracovávat svými senzory. e ení t etí úrovn by vyžadovalo robotický hardware, který nemáme k dispozici, a navíc by se celé zkoumání zkomplikovalo nad rozsah této diplomové práce.

2.2.3 Zadání úlohy

...pro potkana

V kruhové arénce bez dal ích p ekážek se pohybuje potkan a robot. Potkan p edstavuje ko ist, robot predátora. Potkan-ko ist nesmí dovolit, aby vzdálenost mezi ním a predátorem klesla pod 25 cm, jinak dostane slabý elektrický ok. Ko ist je hladová a v pravidelných intervalech padají do náhodných míst arénky zrnka potravy (t stoviny), je tedy motivována se pohybovat. Predátor se po arénce pohybuje náhodn po p ímých trajektoriích. Po nárazu do st ny predátor couvne, náhodn se oto í a jede op t po p ímce. Pokus trvá 20 minut.

Aparaturu testu tvo í arénka kruhového tvaru o pr m ru 85 cm, obehnaná tvrdým papírem, s podlážkou z drát né sí oviny (elektricky uzemn nou). Oba tvorové v aréncе jsou ozna eni LED diodou li ící se velikostí, kterou mají na zádech. Ko ist je navíc vybavena okovým drátkem, který je p ipevn ný k nerezové kotvi ce, jež je z ásti implantována pod k ži zví ete. V prost edí se navíc nachází krmítko s disperzorem, které každé cca 3 sekundy pustí dávku 2–3 zrní ek t stovin. Celou situaci sleduje kamera, zaznamenávající pohyb LED, spojená s po íta em. Ten každých 20ms ukládá sou adnice obou potkan a vyhodnocuje jejich vzdálenost, aby vydal/nevydal povel k elektrickému oku.

...pro robota

V p ípad , který budeme dále zkoumat v této práci, p edstavuje ko ist robot. Tomuto robotovi budeme v dal ím textu íkat RoboKrysa, protože budeme pracovat je t s jedním robotem — predátorem. Úloha pro RoboKrysu je stejná jako pro potkana. Musí se držet od predátora ve vzdálenosti 25 cm, jinak následuje trest. Trestem není tentokrát slabý elektrický ok, ale je to jen informace p edaná ízení robota, že k tomuto oku do lo. ízení RoboKrysy je nastaveno tak, že tuto informaci vnímá negativn . Pravideln po 3 sekundách padají do arénky virtuální zrnka potravy. Prost edí si pamatuje kde zrnka jsou a v p ípad , že robot na n jaké narazí, je zrní ko automaticky sn deno a tuto událost vnímá robot pozitivn . Test trvá 6 minut (tato doba je posta ující).

Aparaturu testu tvo í v tomto p ípad — krom obou robot — kruhová, 7 cm vysoká, hliníková ohrada o pr m ru 85 cm. Oba roboti jsou ona eni r znobarevnými LED diodami. Ty slouží k ur ení pozice a odli ení obou robot . Robot-predátor se pohybuje autonomn zp sobem popsaným vý e. RoboKrysa je vybavena Bluetooth modulem pro bezdrátový p enos. Celou scénu sleduje kamera p ipojená k po íta i, který ídí celý pokus. Na tomto po íta i b ží hlavní ídící program RoboKrysy, který jí posílá p íkazy na základ konfigurace prost edí. Zde se zpracovávají i informace o pr b hu testu, na základ nichž se generují virtuální oky pro ko ist. Šoky jsou oznamovány p ímo ízení RoboKrysy.

2.3 Zp sob porovnávání

Jak již bylo zmín no, máme k dispozici údaje o pr b hu u ení potkana vý e popsané úlohy. Konkrétn se jedná o textové logy zaznamenávající v pravidelných asových intervalech pozici predátora a ko isti v aréncе. Takový log máme pro každý reálný pokus, který jeden potkan v pr b hu svého procesu u ení vykonal. Tedy pro jednoho jedince se jedná o 20 dvacetiminutových log . Na ím cílem bude v první ad p ipravit RoboKrysu a testovací prost edí tak, aby bylo možné po ídit srovnatelné záznamy o její jízd a pohybu predátora. Dal í velkou ástí této práce bude návrh a implementace robotova ídícího systému, který umožní se nau it daný úkol. Pr b h u ení budeme zaznamenávat do log , na nichž pak bude probíhat na e dal í zkoumání:

- Pokusíme se najít souvislosti mezi pohybem potkana a robota, hlavní charakteristiky tohoto pohybu v závislosti na ídícím systému, v em se robotova strategie li í a v em s potkanem shoduje.

- Bude nás zajímat doba u ení úlohy, zejména s ohledem na po et a dobu reálných pokus robota, protože jedním z na ich cíl bylo tyto parametry minimalizovat.
- P i porovnání se pokusíme dát do souvislosti i proces u ení obou jedinc . Tedy, jakými fázemi pro la strategie robota a potkana, než dosp l proces u ení k více i mén zdárnému konci.
- Uvedeme absolutní porovnání nau ených jedinc podle kvality e ení zadaného úkolu.
- Kone n ov íme schopnost robota nau it se, se stejným ídícím a u ícím systémem, modifikace úlohy ve stejném prost edí a otestujeme vlastnosti chování nau eného jedince po p enesení do modifikovaného prost edí.

3 Úvod do evolu ní robotiky

Oblast evolu ní robotiky je velice iroká a i její stru né zmapování by zna n p evý ilo rozsah této diplomové práce. V této kapitole uvedeme proto jen stru ný popis základních pojm , metod a problém , týkajících se evolu ní robotiky a robotiky obecn , se kterými bude tento text dále pracovat. Komplexn j í pohled na celkovou problematiku lze nalézt nap íklad v [1] .

3.1 Genetické algoritmy

První z metod evolu ní robotiky, kterou zde popí eme, jsou genetické algoritmy [2] . Je to metoda kopírující p írodní evoluci a její hlavní princip — p irozený výb r. Genetický algoritmus pracuje na populaci um lých *chromozom* (genetických vzork). V populaci se ast ji *reprodukují* (a tedy p edávají sv j genetický materiál) kvalitn j í jedinci (*fenotypy*). Nov vzniklí jedinci p ebírají genetickou informaci svých rodi (tomuto jevu budeme dále íkat *k ížení*) a jsou na n aplikovány náhodné zm ny, které mohou jejich genetickou výbavu modifikovat (tzv. *mutace*). Tento proces reprodukce, k ížení a mutace se opakuje v cyklech — generacích.

V chromozomu je zakódována kompletní charakteristika jedince. Je-li nap íklad um lým jedincem neuronová sí , jsou v chromozomu uloženy její váhy, charakterizuje-li jedince jen jedno íslo, je v chromozomu zakódována jeho hodnota. Kvalitu jedince udává tzv. *fitness funkce (ú elová funkce)*. ím vy í je hodnota fitness funkce, tím kvalitn j í je jedinec. Postup typického genetického algoritmu je následující:

1. Genetické u ení typicky za íná populací náhodn vytvo ených chromozom .
2. Každý chromozom je p eveden na jedince a z n ho pak spo ítána hodnota fitness.
3. Nová generace vznikne ze starých chromozom t emi genetickým operátory — selektivní reprodukcí (p írodním výb rem), k ížením a mutací.
4. Pokra uje se bodem 2, dokud není nalezen hledaný jedinec, nebo se hodnota fitness nep estane zvy ovat.

Selektivní reprodukce spo ívá v množení jedinc vybraných podle jejich hodnoty fitness. Tedy, jedinci s vy í fitness budou vytvá et své potomky ast ji než jedinci s niž í fitness. astou implementací selektivní reprodukce je metoda rulety. Každý dílek rulety odpovídá jednomu jedinci a velikost dílku je úm rná velikosti fitness p íslu ného jedince. Výb r pak probíhá tak, že se to í ruletou a který dílek se vybere, ten jedinec se rozmnoží.

Tato metoda funguje v t inou dob e, problémy v ak mohou nastat v následujících dvou p ípadech. Malý po et jedinc má o hodn vy í fitness než ostatní. Pak se pro reprodukci vybírají v t inou oni a zužuje se prohledávaný prostor. Druhý p ípad nastane, mají-li v ichni jedinci tém stejnou fitness. Pak se vybírají v ichni p ibližn stejn asto a genetický algoritmus se zm ní na náhodné prohledávání prostoru e ení. e ením m že být nap íklad kálování fitness — zv t ení, i zmen ení rozdíl mezi jednotlivými hodnotami. Problém e í i výb r založený na po adí — tedy hodnota fitness slouží k se azení jedinc . Výb r se pak provádí podle po adí, ne podle absolutní velikosti fitness.

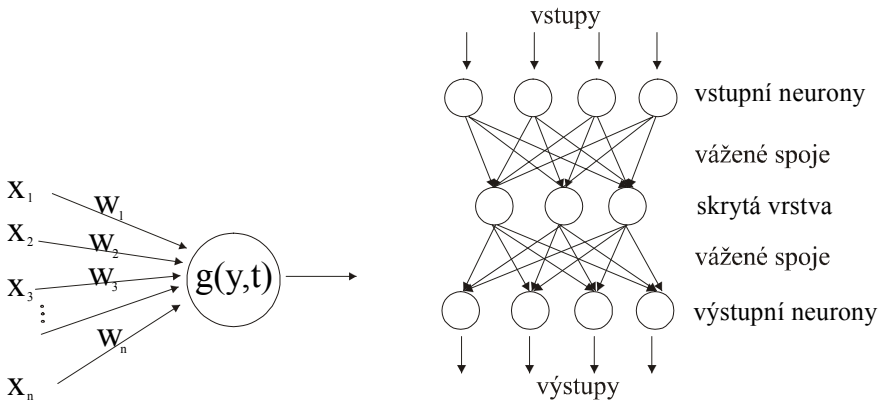
K ížení je operátor, p i n mž dochází k vým n genetického materiálu mezi jedinci populace. Ti se náhodn spárují a vym ní si navzájem ást svého chromozomu. P i *mutaci* dochází k náhodné zm n na náhodném míst chromozomu. V na em p íkladu, kdy chromozom reprezentuje váhy neuronové sít , m že být mutací nap íklad p i tení náhodné hodnoty k jedné z vah sít .

Velkou výhodou genetických algoritm pro evolu ní robotiku je obecnost jejich využití. Lze-li kvalitu e ení úlohy jakkoliv kvantitativn ohodnotit, je možné se pokusit najít její e ení genetickým algoritmem. Velká ást e ení úkol evolu ní robotiky má charakter posloupnosti akcí, které ve finále vedou i nevedou k cíli, a mnohdy je velice složité hodnotit již mezikroky, nevíme-li, povedou-li k cíli. Ideální metodou pro e ení takových úkol jsou práv genetické algoritmy. Vyvíjí totiž e ení jako celek, který se snaží zlep ovat. P íkladem m že být genetický vývoj strategie robota. Strategii je možné ohodnotit až po jejím otestování v úloze, kterou má e it.

Genetická evoluce s sebou nese i mnoho úskalí, jako je bootstrap problem a dal í, které jsou ale nad rámec této kapitoly. Zabývá se jimi ale nap íklad [2] .

3.2 Neuronové sít

Dal í metodou um lého u ení jsou neuronové sít . Stejn jako pro genetické algoritmy byla i pro neuronové sít vzorem p íroda — tentokrát propojení neuron v mozku. Neuronová sí se tedy skládá, stejn jako biologický mozek, ze základních jednotek — neuron (perceptron). Um lý neuron má libovolný po et vážených vstup a jeden výstup.



Obr. 1 Formální neuron a neuronová sí

Neuronová sí vznikne spojením n kolika neuron . Spoje mohou obecn vést mezi libovolnými dv ma neurony. V této práci ov em pracujeme jen s tzv. vrstevnatou sítí, což je speciální druh, kde spoje vedou jen z *i*-té vrstvy do *i+1*-ní. Navíc na e sí bude mít vždy propojené v echny neurony mezi dv ma po sob následujícími vrstvami. Výpo et neuronu probíhá tak, že zjistí hodnoty $x_1, x_2, \ldots, x_n$, na svých *n* vstupech (odpovídají výstup m neuron z p edchozí vrstvy nebo vstup m sít), vynásobí je p íslu nými váhami $w_1, w_2, \ldots, w_n$ a se te, tj. spo ítá se hodnota

$$y = \sum_{i=1}^{n} w_i x_i ,$$

ozna ovaná jako *vážený vstup neuronu*. Tato hodnota se dále transformuje pomocí p enosové funkce. Typickou p enosovou funkcí je tzv. sigmoida:

$$g(y,t) = \frac{1}{1+e^{-\lambda(y-t)}} ,$$

kde parametr ur uje *strmost p enosové funkce*. Parametr *t* se nazývá *práh neuronu*. Práh bývá obvykle zahrnován do hodnoty váženého vstupu. V každé vrstv existuje je t jeden my lený neuron s výstupem *-1* a spojením se v emi neurony v následující vrstv . Hodnota tohoto váženého vstupu pak reprezentuje práh tohoto neuronu. Je z ejmé, že jak p ístup s prahem jako dal ím parametrem neuronu, tak jako dal ím váženým vstupem jsou ekvivalentní. V na í práci používáme práh jako dal í vážený vstup, protože zjednodu uje reprezentaci neuronové sít . Výsledná hodnota $g(y, t)$ ur uje *aktivaci neuronu*. Krom sigmoidy lze použít i jiné p enosové funkce, nap íklad lineární funkci i signum [4] .

Postup p i výpo tu sít je následující:

1. Aktivují se neurony ve vstupní vrstv , tj. je jim nastavena hodnota *y*.
2. Každý neuron v následující vrstv na základ hodnot v p edchozí vrstv spo ítá podle vý e uvedených vztah svoji aktivaci.
3. Tento postup se zopakuje postupn ve v ech vrstvách sít , dokud nejsou spo ítány aktivace neuron vrstvy výstupní.
4. Aktivace neuron výstupní vrstvy jsou výstupy celé sít .

Ve kerá informace je v síti uložena ve vahách (p íp. prazích). Klí ovou otázkou fungování sít je tedy vhodné nastavení vah. Pro v t inu úloh samoz ejm není vhodná hodnota vah z ejmá, proto existují metody, jak nastavit váhy sít automaticky.

Typickou metodou pro automatické nastavení vah vrstevnaté sít je algoritmus zp tného í ení chyby (backpropagation). Jedná se o u ení s u itelem[†]. K dispozici je tedy trénovací množina vstup a k nim požadovaných výstup , které by m ly úlohu, kterou se má sí nau it, charakterizovat. Kvalita adaptace sít na úlohu se hodnotí pomocí *energetické* (*chybové) funkce*

[†] U ení s u itelem - V režimu u ení (nastavování vah) jsou síti p edkládány dvojice [vstup, požadovaný výstup], které se má nau it. P i u ení bez u itele není k dispozici požadovaný výstup.

$$E = \frac{1}{2} \sum_{\substack{p \\ p \text{ je vstup} \\ \text{p edložený síti}}} \sum_{\substack{j \\ j \text{ je neuron na} \\ \text{výstupní vrstv}}} (y_{jp} - d_{jp})^2$$

kde y_{jp} je aktivace j-tého neuronu výstupní vrstvy (tj. skute ný výstup) p i p edložení p-tého vzoru z trénovací množiny a d_{jp} je požadovaný výstup stejného neuronu p i tomtéž vzoru. ím lépe je sí na úlohu nau ená, tím mén se li í skute ný a požadovaný výstup a tím je tedy hodnota E men í.

My lenkou algoritmu zp tného í ení je minimalizace této funkce, tj. hledají se takové váhy w_{ij}, které minimalizují hodnoty E pro v echny prvky p z trénovací množiny. Vzhledem k tomu, že výstup sít a tudíž i hodnota E závisí na použité p enosové funkci, budeme dále p edpokládat, že použitou p enosovou funkcí je sigmoida, protože tu používáme i v na í práci. P esné odvození pravidel lze nalézt v [4] , zde uvedeme jen nástin fungování metody.

Váhy sít se m ní po každém p edložení trénovacího vzoru. Zm na vah sít w_{ij} mezi dv ma vrstvami se provádí gradientní metodou a je založená na výpo tu $\frac{\partial E}{\partial w_{ij}}$ a probíhá ve dvou krocích:

1. spo ítá se chyba na l-té vrstv ,
2. na základ této chyby se adaptují váhy mezi $(l - 1)$-ní a l-tou vrstvou.

Podstatné je, že se postupuje sm rem od výstupní vrstvy sm rem ke vstupní vrstv , tj. p i prvním kroku algoritmu je l rovno po tu vrstev sít . Nech je n-tá vrstva sít vrstvou výstupní. Pak se chyba na j-tém neuronu této vrstvy spo ítá podle vztahu

$$\delta_j^n = y_j (1 - y_j) (d_j - y_j),$$

kde je parametr ur ující strmost p enosové funkce, y_j je aktivace j-tého a d_j je požadovaný výstup j-tého neuronu této vrstvy.

Pokud vrstva není výstupní, spo ítá se chyba na j-tém neuronu l-té vrstvy podle vztahu

$$\delta_j^l = \lambda y_j (1 - y_j) \sum_{k=1}^{n} \delta_k^{l+1} w_{jk} .$$

Adaptace vah mezi vrstvou $(l - 1)$ a vrstvou l (za p edpokladu, že na vrstv l je pro v echny neurony spo ítána chyba δ_j^l) pak probíhá podle vztahu

$$w_{ij}(t + 1) = w_{ij}(t) + \delta_j^l y_i + [w_{ij}(t) - w_{ij}(t - 1)],$$

kde $w_{ij}(t)$ ozna uje váhu mezi i-tým neuronem vrstvy $l - 1$ a j-tým neuronem vrstvy l v ase t, δ_j^l je chyba na j-tém neuronu l-té vrstvy, y_i je aktivace i-tého neuronu $(l - 1)$-ní vrstvy a a jsou parametry systému, jejichž význam je popsán dále. Chyba z výstupní vrstvy se takto zp tn í í až ke vstupní vrstv , p i emž p epo ítáním vah mezi vstupní a po ní následující vrstvou algoritmus skon í. Adaptace se obvykle provádí opakovan , p i emž p i první iteraci je $t = 1$, $w_{ij}(t)$ jsou malé náhodné hodnoty a $w_{ij}(t) = w_{ij}(t - 1)$.

len $\delta_j^l y_i$ odpovídá požadované zm n vah ur ené parciální derivací funkce E podle w_{ij} vynásobené konstantou . len $[w_{ij}(t) - w_{ij}(t - 1)]$ odpovídá zm n vah od p edchozího kroku adaptace vynásobené konstantou .

Jak již bylo e eno, sí se snaží touto numerickou metodou minimalizovat chybu vyjád enou pomocí energetické funkce. Parametr , obvykle nazývaný *parametr u ení*, zaji uje p esko ení m lkých lokálních minim. P íli velké hodnoty tohoto parametru vedou k rychlému vývoji vah sít , který je žádoucí spí e na za átku výpo tu. Proto by m la hodnota v pr b hu u ení klesat. Parametr , obvykle nazývaný *moment u ení*, má význam p i udržování sm ru sít v p ípad pr chodu úzkými lokálními minimy a snižuje nebezpe í ustálení sít v takových minimech. Obecn nelze stanovit optimální hodnotu parametr a ani jejich vzájemný pom r. Jejich vhodné nastavení závisí na konkrétní úloze a je asto nutné ho ov ovat experimentáln .

Dal í metodou k optimálnímu nastavení vah neuronové sít jsou genetické algoritmy. Této metody se využívá v úlohách, kdy je obtížné vytvo it trénovací množinu, nebo se kvalita sít projeví až po jejím del ím používání. V genetickém algoritmu jsou v chromozomu zakódovány váhy sít a p irozen se touto metodou vyvíjejí váhy, které e í problém efektivn ji. Tohoto postupu budeme využívat i v na í práci.

Metoda neuronových sítí je v evolu ní robotice hojn využívána. Neuronová sí je vhodný prost edek jak k ízení robota tak nap íklad k p edpovídání stavu senzor i rozpoznávání situací. V na í práci budeme využívat geneticky u enou neuronovou sí k ízení strategie robota a dv sít u ené algoritmem zp tného í ení k zachycení a simulaci reálných jev v na em simulátoru.

3.3 Mobilní robotika

Robotika jako celek je masivn se rozvíjející odv tví, které se s postupujícím vývojem techniky zabývá ím dál ir ím okruhem témat. Jednodu í i složit j í roboty lze dnes potkat tak ka v ude — pr mysloví roboti v montážních halách, manipulátory v automatizovaných skladech, p esná za ízení najdou své místo i v medicín , vybavení pro vesmírný výzkum, zbrojní pr mysl, roboti jsou nyní velkým hitem i v pr myslu filmovém... Velká ást t chto velice praktických aplikací se v ak týká robot statických — umíst ných na jedno místo, kde vykonávají sv j úkol.

Mobilní roboti, kte í mohou sami m nit svou pozici, stále na sv j boom ekají. I zde se objevují výjimky, které d lají mobilní robotice dobré jméno v praxi — roboti na Marsu, záchraná tí roboti, automatické vysava e a seka ky atd., ale na masivní roz í ení mobilních robot se stále eká. D vodem je jist i složitost e ení problém , které vzniknou p i vypu t ní robota do volného prost edí, protože stavový prostor, který toto prost edí nabízí a který musí mobilní robot e it, je obrovský [12] . V této podkapitole nastíníme n kolik základních problém mobilní robotiky a popí eme základní pojmy, které budeme používat i v dal ím textu. B hem své existence si mobilní robot musí položit minimáln t i základní otázky:

Kde to jsem?

Na tuto otázku se snaží odpov d t jedna z nejd ležit j ích ástí mobilní robotiky — lokalizace. Lokalizace je disciplína, která se snaží z informací

z robotových senzor ur it jeho polohu ve známém i neznámém prost edí. Bohužel neexistuje senzor, který by obecn dokázal tento problém vy e it v libovolném prost edí. Proto se musí p i lokalizaci brát v úvahu informace z n kolika zdroj a p edpokládat i možné nep esnosti m ení. Vyvinuté metody a algoritmy musí být robustní a rychlé, aby byly schopny pracovat v reálném ase a reálném prost edí, v n mž se robot pohybuje. To vyžaduje mnohdy nemalý výpo etní výkon, proto se dají dne ní mobilní roboti s nadsázkou ozna it jako „po íta e na kole kách".

Informace ze senzor , které má robot pro lokalizaci k dispozici, se d lí vzhledem k robotovi do dvou základních skupin:

- relativní - Ur ují zm nu pozice robota. Do této skupiny pat í nap íklad *enkodery*, což jsou senzory, které snímají otá ení kol robota — sm r otá ení a úhel, o kolik se kolo oto ilo. Je-li tedy robot v neznámém prost edí, lze z vhodn umíst ných enkoder zjistit, jede-li robot rovn , i na jaké p ibližné pozici, v i pozici startovní, se práv nachází. Výhoda relativních senzor je v jejich snadné implementovatelnosti — v t inou nevyžadují nic od vn j ího prost edí. Hlavní nevýhodu p edstavuje možnost, že robot zm ní pozici zp sobem, který tyto senzory nejsou schopny zaznamenat a robot se o této zm n nikdy nedozví. P íkladem m že být nap íklad posuv robota do boku, který vznikne nap íklad p i kolizi. Robot pohybuje, a tedy m ní polohu, kola a enkodery na nich ale z stávají v klidu. Opa ným p íkladem m že být protá ení kole ek nap íklad po nárazu do zdi. Robot sice nem ní svou pozici, enkodery ale pohyb kole ek zaznamenávají.
- absolutní - Ur ují pozici robota v i bodu umíst nému ve vn j ím prost edí. P íkladem budiž nap íklad kamera, která umí rozpoznat nap íklad naviga ní zna ky na zdech, nebo jen jednoduchý aktivní nárazník, který dává informaci o tom, že práv do n eho narazil. Tyto senzory dávají velice cenné informace o absolutní pozici robota v prost edí. Je v ak nutné brát je s rezervou, protože vn j í prost edí se m že m nit, aniž by to robot zaznamenal — pro zmatení kamery sta í už jen malá zm na osv tlení, aktivní nárazník není schopen rozeznat náraz do zdi od kolize s jiným robotem (problematikou robotických senzor se í eji zabývá nap íklad [13]).

Skute nou sílu tedy p edstavuje až kombinace t chto dvou druh senzor , kterou zaji uje nap íklad motoda Monte Carlo Lokalizace (MCL), která je popsána nap íklad v [14] . Její princip je jednoduchý:

1. Udržuje si množinu vážených vzork . Každý vzorek nese krom své váhy je t pozici robota.
2. P i pohybu si upraví vzorky svou pozici podle relativní informace o pohybu a p edpokládané chyby m ení. Nap . enkodery oznámí, že se robot pohnul vp ed o 2 cm. V echny vzorky tedy posunou svou pozici o p ibližn (zapo ítaná chyba m ení) 2 cm vp ed.

3. Provede se korekce podle absolutní informace — vzorky, které odpovídají absolutnímu kritériu (ve své pozici mají teoreticky možnost zaznamenat stejnou absolutní informaci, jako skute né senzory), svou váhu posílí, ostatní vzorky sníží. Hlásí-li nap íklad aktivní nárazník robota, že robot do n eho narazil, vzorky, které odpovídají pozicím blízko st n, se posílí.
4. Provede se p evzorkování — vzorky s p íli malou vahou zaniknou a místo nich se vytvo í na náhodných pozicích nedaleko p edpokládané pozice vzorky nové. Cyklus se opakuje.

Jako výsledná pozice se bere nap íklad pr m rná hodnota vzork s vysokou váhou. To, že je pozice robota reprezentována jako množina vzork a ne jako jedno íslo, s sebou m že nést problémy s mírnou nep esností pozice, nebo její skokovou zm nou. Obecn je v ak metoda MCL velice robustní.

Co tu mám d lat?

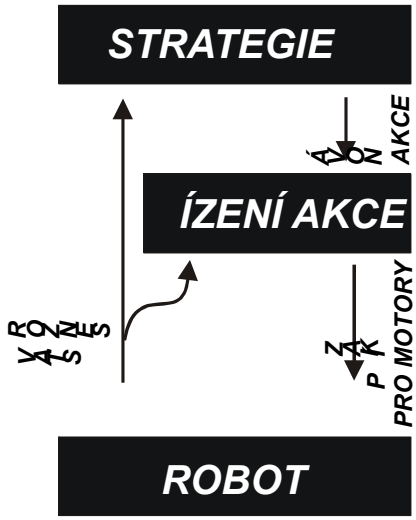
Robot je za ízení, po kterém se typicky chce pln ní n jakého úkolu v prost edí, ve kterém se pohybuje. V p edchozí ásti jsme ur ili robotovu polohu v tomto prost edí. Získá-li i informace ze v ech ostatních senzor , má p ehled o sob i prost edí, který pot ebuje k rozhodnutí o svém dal ím sm ování. ást robotova ízení, která rozhoduje o dal ím postupu se nazývá strategie. Podle stupn její volnosti existují t i varianty strategie:

1. pevná - Dráha po které robot pojede a innosti, které p i tom bude vykonávat, jsou pevn dané a nep ipou tí zm nu. Strategie nemá prost edky na e ení nenadálých stav . Takový robot najde uplatn ní jen ve velmi omezeném po tu úloh, které mohou zajistit um lé prost edí, které se nebude m nit.
2. interaktivní - Robot má zadán úkol a strategii, jak ho dosáhnout. Dokáže ale reagovat na zm ny prost edí a p izp sobovat jim své chování. Robot s tímto druhem strategie je schopný fungovat i v reálném prost edí, kde se pohybují nap íklad lidé, nebo jiní roboti.
3. inteligentní - Strategie robota není dána. Robot zná pouze zadání úkolu, který má splnit. To, jak ho splní ale záleží na n m. Tímto druhem strategie se budeme zabývat v této práci. Její možnosti jsou sice zatím zna n omezené, ale teoretická síla takového p ístupu slibuje rozvoji takových strategií velkou budoucnost.

Strategie tedy plánuje akce, které robot v budoucnu vykoná, aby splnil zadaný úkol. Z stává poslední otázka — jak tyto akce vykonat.

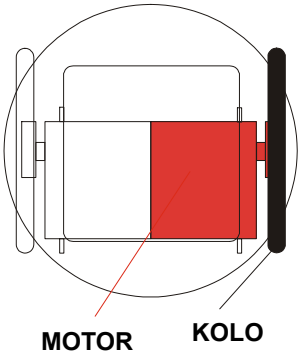
Jak to mám ud lat?

Základním druhem akce, kterou m že strategie pro robota naplánovat, je jízda. Realizaci této akce m že zajistit bu sama strategie, nebo m že pouze vydat p íkaz typu „je na [x,y]“ nebo „je 30 cm rovn “ a p edat ízení jinému modulu. Tento modul by se m l následn postarat o ovládání motor robota takovým zp sobem, aby byl p íkaz strategie spln n.



Obr. 2 ídící schéma robota

Pro rozd lení do dvou modul hovo í fakt, že zatímco plánovat celkové chování robota m že vyžadovat velký výpo etní výkon a del í as pro zpracování v ech údaj ze senzor , pro ízení pohybu robota je nutná rychlá interakce mezi stavy enkoder , indikujícími okamžitou rychlost, a p íkazy pro motory, které rychlost ovliv ují. Pro zaji t ní plynulosti a p esnosti pohybu se využívají techniky z oboru regulace. Nej ast j í metodou, kterou využívá i ná robot, je tzv. PID kontrolér, který generuje p íkazy pro motory tak, aby byla pokud možno co nejp esn ji dosažena požadovaná rychlost. Zp sob, jakým toho dosahuje je popsán v [15] .



Obr. 3 P dorys RoboKrysy

Ovládání robota souvisí i s jeho konstrukcí. V ichni roboti, které budeme používat v této práci, jsou diferenciáln ízená vozidla. To znamená, že robot m ní sm r tím, že se jeho hnaná kola na hlavní ose otá í r znou rychlostí. Roboti mají 2 motory a 2 hlavní hnaná kola. Dále mají 1-2 kola op rná. Takto zkonstruovaný robot má mnoho výhod — jednoduchou matematiku pro výpo et zm ny pozice, jednoduchou hardwarovou implementaci, snadné ízení atd. Tato konstrukce v ak není vhodná do náro n j ího terénu.

Motory samotné lze ovládat r znými zp soby podle druhu motoru. Roboti, o nichž se budeme zmi ovat, využívají stejnosm rné motory, které ídí metodou modulace í ky pulzu (Pulse Width Modulation - PWM). Tato metoda využívá pro regulaci výkonu motor r zn dlouhých pulz vstupního nap tí. Del í pulzy

znamenají v t í napájení motoru proudem, tedy vy í výkon. Krat í pulzy naopak výkon snižují. P íkazy pro motory, o kterých se budeme zmi ovat v této práci, jsou práv kombinace délky pulzu a sm ru otá ení motoru.

V této kapitole jsme se zmínili jen o zlomku zajímavých ástí mobilní robotiky. Zam ili jsme se jen na oblasti, které se p ímo dotýkají ízení a konstrukce na eho robota. Lep í p ehled o mobilní robotice lze získat v [12] .

4 Struktura u ení um lého systému

V této kapitole popí eme architekturu systému, který umožní robotovi adaptaci na danou úlohu, a podrobn ji rozebereme jeho jednotlivé ásti. Záv re ná ást kapitoly shrnuje celkový postup p i adaptaci systému ízení robota na neznámou úlohu.

4.1 Simulátor

Jak je již uvedeno vý e, jedním z na ich hlavních požadavk na proces u ení byla minimalizace po tu reálných jízd robota. Toho jsme cht li dosáhnout maximálním využitím simulátoru p i vývoji strategie. P i u ení je obecn t eba mnohokrát ov it vlastnosti nau ené strategie v prost edí, pro které je ur ena, a na základ výsledk test dál pokra ovat v procesu u ení. Testy v reálném prost edí, zejména vyžaduje-li metoda u ení t chto test mnoho, jsou ale asov i hardwarov velmi náro né.

V úvahu p ipadá testování v prost edí simulovaném. V softwarovém simulátoru vymodelujeme robota i jeho testovací prost edí a napojíme systém ízení robota tak, aby tentokrát ídil robota simulovaného. Použití virtuálního sv ta s sebou ov em nese problém nekorespondence simulovaných vlastností robota a prost edí se sv tem reálným. Zam ili jsme se proto na vytvo ení co nejrealisti t j ího simulátoru, aby zmín ný rozdíl mezi simulací a skute ností byl co nejmen í. Protože se ale této vlastnosti simulátoru nelze vyhnout zcela, snažili jsme se i o vytvo ení co možná nejrobustn j í strategie tak, aby byl p i p echodu do reálného prost edí co nejmén ovlivn n úsp ch robota p i pln ní úkolu. V této podkapitole popí eme ná p ístup k vytvo ení takového simulátoru.

4.1.1 Fyzikální simulátor

Ná první pokus o realistický robotický simulátor el cestou simulace fyzikálních jev , které se p i pohybu robota v prost edí objevují. P edesílám, že tuto metodu finální e ení nepoužívá. V nujeme jí ale tuto podkapitolu, protože se jedná o zajímavou metodu a jako mezikrok v cest k výslednému e ení byla velice d ležitá.

Zjednodu en e eno — robot je fyzikální t leso složené ze základních t les a definovaných pevných a volných spoj mezi nimi. Prost edí, ve kterém se takový robot pohybuje, je složeno ze stejných základních t les. V prost edí p sobí podobné síly jako v reálném sv t . Mezi robotem a prost edím m že docházet ke kolizím, p i kterých se v echny zú astn né objekty chovají podle svých fyzikálních vlastností. Vytvo ení takového simulátoru je netriviální úkol, který vyžaduje hluboké znalosti fyziky a nemalé množství asu. Na ím hlavním cílem ov em nebylo zabývat se fyzikálními simulacemi, použili jsme tedy hotový fyzikální simulátor, který byl sou ástí softwarového projektu „Eurobot 2004“ [16] , na n mž se autor této práce podílel.

Tento simulátor pracuje v principu tak, že vytvo í simulovaný sv t, do kterého se p ipojí jeden i více nezávislých robot . P ipojený robot má t lo se senzory a aktivními prvky, které se objeví v simulovaném sv t , a své ízení, které robota ovládá. Ve sv t s roboty pak probíhá simulace po krocích. V každém kroku dostane

ízení robota informaci o stavu senzor a strategie robota vygeneruje na základ této informace p íkaz pro aktivní prvky (nap . pro motory). To zp sobí zm ny sil v simulaci fyzikálních jev v prost edí, což se projeví komplexn v dal ím simula ním kroku. Jak pozd ji rozebereme v samostatné kapitole, ízení na eho robota je založeno na pravidelné vým n takových stavových a p íkazových zpráv. Simulátor tedy nahrazuje reálného robota pouze na nejniž í úrovni této vým ny, ale samotné vy í ízení (nap . strategie robota) z stává stejné.

Úvod do Open Dynamic Engine (ODE)

Pro plnou náhradu reálného robota je bezpodmíne n nutné realisticky simulovat fyzikální jevy ve virtuálním prost edí. Simulátor Eurobot 2004 používá pro simulaci fyzikálních objekt produkt t etí strany — knihovnu Open Dynamic Engine. Zde uvedeme stru ný p ehled základ této knihovny, protože fyzikální simulátor je s ní úzce svázaný a v jeho popisu se budeme na tyto základy odkazovat. ODE je voln i itelná knihovna pro simulaci kloubových nepružných t les. Je rychlá, robustní a má vestav nou detekci kolizí. ODE zná n kolik základních t les a tzv. kompozitní objekty, které jsou z t chto t les sestavené. Základními t lesy jsou:

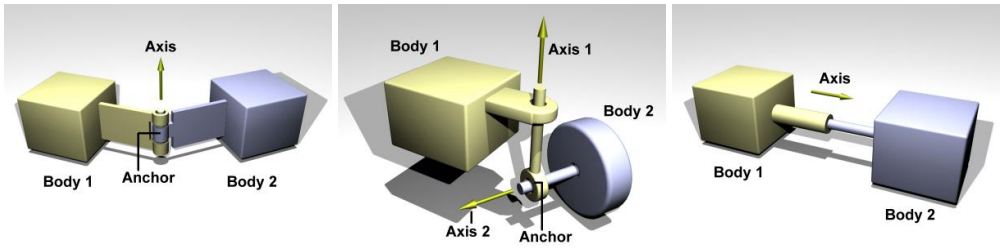
- Krychle
- Koule
- Válec s oblými podstavami
- Paprsek
- Rovina

T lesa jsou ur ena vlastnostmi, které mohou být dynamické, nebo statické. Mezi statické vlastnosti pat í:

- rozložení hmoty v t lese
- pozice t ži t t lesa vzhledem k referen nímu bodu t lesa
- hmotnost t lesa

Dynamickými vlastnostmi jsou:

- pozice t lesa, která je ur ena polohovým vektorem referen ního bodu
- vektor rychlosti referen ního bodu
- orientace objektu, která je dána maticí nebo quaternionem
- úhlový vektor rychlosti

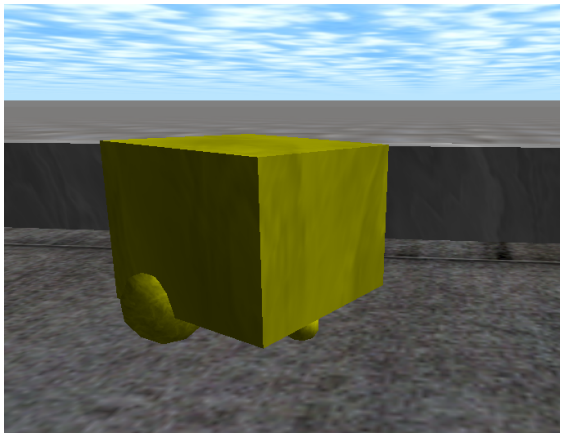


Obr. 4 Schéma spoj pant, záv s a posuvný spoj

Krom t les se v simulovaném sv t vyskytují je t tzv. spoje, pomocí nichž se vyjad ují vztahy mezi objekty. Konkrétn zavád jí omezující podmínky pro pohyb mezi dv ma objekty. Použitelné typy spoj jsou pant, záv s, posuvný spoj a pevný spoj. Pro intuitivní p edstavu práce s objekty a spoji v ODE posta í uvedená schémata. Na objekty a spoje v simulovaném sv t p sobí, stejn jako ve sv t reálném, r zné síly. Síly vznikají bu interakcí objekt v prost edí (nap . p i kolizi), nebo je lze vložit z venku (nap . chceme objekt uvést do pohybu).

Vytvo ení sv ta a robota v simulátoru

Ná fyzikální simulátor usnad uje vytvá ení sv t a robot tím, že používá vlastní jazyk na bázi XML pro jejich popis. Tento jazyk je p ímo ur en pro robotické ú ely, proto implementuje, krom základních entit ODE, i základní senzory a aktivní prvky robota. Pro na i aplikaci bylo tedy t eba vytvo it popis pevného prost edí, ve kterém se bude robot pohybovat, a popis robota.



Obr. 5 RobuTec3 ve fyzikálním simulátoru (o tomto robotovi se zmíníme v kapitole 5 o implementaci systému)

K tomu, aby lo simulátorem nahradit reálné pokusy, je nutné, aby byl popis robota co možná nejp esn j í. Po p esném vymodelování tvaru robota je nutné nastavit mnoho parametr v ech zú astn ných spoj a objekt . Jedná se o:

- p ilnavost kole ek k podlaze - toto je v simulátoru záležitost dvou parametr , jejichž fyzikální ekvivalent není z dokumentace ODE ani z pozorování zcela z ejmý
- hmotnost robota a umíst ní t ži t
- hmotnost dal ích ástí spojených s robotem - nap íklad kole ka
- maximální síla motor - íká, jak velkou sílu (ne rychlost) maximáln vyvine motor k tomu, aby dosáhl požadované rychlosti
- maximální rychlost robota

Nastavení t chto parametr bohužel není zdaleka p ímo aré a je pot eba experimentáln zjistit, jaké nastavení odpovídá reálnému robotovi nejlépe. To je ov em pom rn netriviální úloha. Vezmeme-li navíc v úvahu, že vlastnosti robota se mohou m nit — nap íklad jeho malou zm nou v konstrukci, nebo jen vybitím baterií

— lze o ekávat, že se tato operace nebude provád t pouze jednou. Rozhodli jsme se proto e it ji automaticky.

Idea na eho systémového e ení tohoto problému spo ívá v genetickém vývoji parametr tak, aby chování simulovaného robota s t mito parametry co nejvíce odpovídalo chování robota reálného. Celý postup p i genetickém u ení parametr je následující:

1. Nasbíráme data o reálném chování robota. Provedeme s reálným robotem kalibra ní jízdu, která se skládá z n kolika typických manévr robota. P i této jízd se zaznamenává pozice v ase do souboru.
2. Klasickým genetickým algoritmem vyvíjíme sadu parametr , které zajistí co nejv t í shodu dráhy simulovaného robota s kalibra ní dráhou. Robot je samoz ejm ízen stejnou strategií jako p i kalibraci. Shodu drah m íme v po tu pozic, které se p ibližn shodují jak v ase, tak v umíst ní. Míra této shody slouží zárove i jako fitness funkce p i genetické evoluci parametr .

Výsledky u ení fyzikálního simulátoru

P i implementaci a testování tohoto genetického p ístupu k nastavení parametr robota jsme do li k následujícím pozorováním:

- Simulovaný robot byl schopen nau it se shodn projet kalibra ní dráhu jen v p ípad , že byla dosti jednoduchá (nap . jízda vp ed 60 cm, zato ení o 90°, jízda vp ed 50 cm).
- Simulovaný robot se nenau il složit j í dráhu, která vyžadovala použití virtuálního nárazníku‡ (nap . jízda vp ed 60 cm, zato ení o 90°, jízda vp ed 50 cm, p i emž po 30 cm následovala ze — po kolizi se zdí ji m l robot pomocí virtuálního nárazníku detekovat a zareagovat na kolizi couvnutím). V tomto p ípad je nutné nastavovat i parametry p ilnavosti kol k podlaze, které se siln ovliv ují s rozložením vah a hmotností robota. Na t chto vlastnostech zase závisí nastavení maximálních sil na motorech, setrva nost po vypnutí motor , odst edivá síla v zatá kách atd. Ukázalo se, že pro vývoj takto komplexního chování by mohlo být výhodn j í rozd lit úlohu na n kolik podúloh, které by jednotliv sloužily k vyvinutí jednotlivých parametr . Zvolili jsme rozd lení na tyto dv ásti:
  - Simulovaný robot se u í projet takto definovanou dráhu — jízda vp ed 60 cm, zato ení o 90°, jízda vp ed 50 cm. Nastavuje parametry maximální rychlost, maximální síla motoru a hmotnost.
  - Simulovaný robot se u í jet proti zdi a cílem je, aby neprokluzovala kole ka po nárazu. Nastavují se parametry p ilnavosti kole ek k podlaze.

‡ Virtuální nárazník je neexistující senzor, který detekuje kolizi. Využívá informace z pohybových senzor (enkoder) na kolech robota a informace o požadovaném výkonu motor . Je-li del í dobu požadován nenulový výkon motor a p esto enkodery informují o tom, že se kola neto í, virtuální nárazník ohlásí, že je robot v kolizi. K tomu, aby se dal virtuální nárazník použít, je nutné, aby kola robota neprokluzovala. To u reálného robota platilo. T ení kol v simulátoru ovliv ují parametry „p ilnavost kole ek k podlaze“.

Umíst ní t ži t robota jsme nastavili napevno podle parametr reálného robota. Cht li jsme ov it p edpoklad, že bude-li se simulovaný robot chovat realisticky na jedné úloze, bude fungovat p ibližn dob e i v ostatních situacích. To se bohužel nepotvrdilo — robotovo chování na p íkaz zato ení s odli ným úhlem než kalibra ním bylo velice vzdálené od chování reálného. Pomoci by mohlo u ení parametr na n kolika r zných úhlech, ale úloha by se tím zna n zkomplikovala a zpomalila.

- N které kombinace parametr v ODE zp sobí pohyb, který nemá paralelu v reálu (knihovna ODE je stále ve vývoji) — robot se nap íklad propadá do podlahy, nebo se p i prudkém bržd ní vznese. Tedy, robot se n kdy v simulátoru nau í projet kalibra ní dráhu dle na eho jednoduchého kritéria správn , p i simulaci se ale dosáhne této trajektorie odli ným zp sobem než ve skute nosti. Krom vizuální kontroly je v podstat nezjistitelné, že se p i jízd v simulátoru jedná o tento p ípad. Navržení fitness funkce, která by takové vzorky odhalila a trestala by bylo složité a neobecné.
- Simulátor je navržen pro testování více robot sou asn a je velice složitý. P i jeho vývoji se nep edpokládalo jeho využití p i genetickém u ení, které vyžaduje vysokou rychlost simulace, protože testuje mnoho vzork v mnoha generacích. Simulátor je tedy relativn pomalý a i nau ení jednoduché trajektorie zabere nemalé množství asu.

Na základ t chto pozorování, jejichž výsledky nevyznívají pro použití fyzikálního simulátoru p íli pozitivn , a faktu, že v programu, postaveném na produktu t etí strany (ODE), nelze snadno dohledat p í iny zmín ných problém , jsme se rozhodli hledat alternativní cestu k rychlé, p esné a spolehlivé simulaci robota. P ínosem v této slepé v tvi bylo d kladné otestování genetického frameworku, o kterém se dále zmíníme v kapitole 5.3 o implementaci genetického u ení a který jsme použili i ve výsledném systému. Modelování robota a jeho vlastností do fyzikálního simulátoru odhalilo i slabiny na eho prvního reálného robota, což nakonec vedlo k vývoji robota nového.

4.1.2 Neuronový simulátor

V p edchozí podkapitole nám p ibyly dal í požadavky na simulátor. Celkem tedy po simulátoru požadujeme:

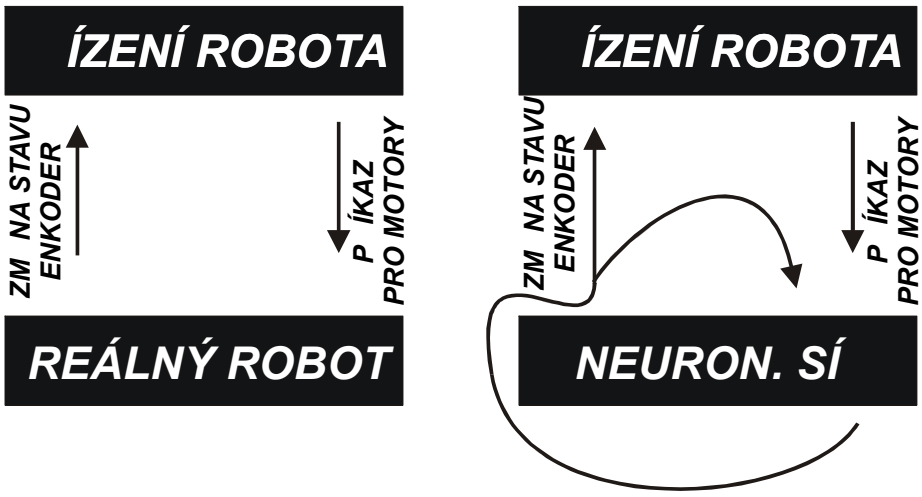
- Musí maximáln odpovídat realit .
- Musí být velice rychlý.
- Musí umož ovat snadné vytvo ení modelu reálného robota.
- Nechceme používat cizí produkty, které nelze od základu upravovat.

Jelikož jsme p edpokládali využití metody neuronových sítí v n kterých ástech adaptabilního ídícího systému, vytvo ili jsme obecný neuronový framework — sadu t íd umož ující snadnou implementaci vrstevnaté neuronové sít u ené algoritmem zp tného í ení. Po analýze požadavk na simulátor a možnosti snadné implementace obecné neuronové sít jsme se rozhodli ov it možnost použití této metody p i simulaci robota. Idea je jednoduchá:

1. Reálný robot p i kalibra ní jízd nasbírá data ve form :

```
< as 2><stav enkoder v ase 1><p íkaz na motory v ase 1><stav enkoder v
 ase 2>
< as 3><stav enkoder v ase 2><p íkaz na motory v ase 2><stav enkoder v
 ase 3>
< as 4><stav enkoder v ase 3><p íkaz na motory v ase 3><stav enkoder v
 ase 4>
...
```

   Na každém ádku logu máme tedy informaci o stavu senzor na kolech p ed provedením p íkazu, p íkaz pro motory a stav senzor po provedení p íkazu.
2. Pro simulátor požadujeme neuronovou sí , která má na vstupu stav enkoder a p íkaz na motory v aktuálním kroku a na výstup dá stav enkoder v kroku následujícím. Data z logu lze tedy použít p ímo jako trénovací množinu pro na i sí .
3. Pomocí této trénovací množiny nastavíme váhy sít metodou zp tného í ení.
4. Již bylo zmín no, že ízení robota pravideln posílá niž ím vrstvám p íkazy a zp t od nich dostává stav senzor . Na i nau enou neuronovou sí m žeme tedy p ímo p ipojit do tohoto systému na nejniž í úrove , kde nám bude nahrazovat skute ný základní hardware robota. Bude nám tedy nadále p edpovídat, jakým zp sobem se robot pohne v závislosti na p edchozí poloze a p íkazu na motory.
5. P edpokládáme, že — díky vlastnosti zobec ování — p i zvolení vhodné kalibra ní dráhy bude neuronová sí schopna p edpovídat s rozumnou p esností i pohyb, který nebyl p ímou sou ástí trénovací množiny.



Obr. 6 Náhrada reálného robota neuronovým simulátorem

Tato idea nabízela, v p ípad funk nosti, spln ní v ech ty na ich požadavk . Použití již nau ené neuronové sít je velice rychlá metoda, jak aproximovat i velice složité funkce. Nau ená neuronová sí by m la mít tu vlastnost, že na vstupech

trénovací množiny bude dávat tém stejné výstupy jako p i kalibra ní jízd , a na jiných vstupech by m ly být výstupní hodnoty rozumnými interpolacemi mezi výstupy trénovacími. To by m lo sta it k pom rn reálné simulaci robota. Zm na parametr reálného robota by pak vyžadovala jen projetí kalibra ní dráhy a p eu ení sít podle nové trénovací množiny.

Naopak nevýhodou této metody je obtížná zjistitelnost, co se vlastn sí nau ila, protože její otestování ve v ech situacích a porovnání s chováním reálného robota je nerealizovatelné. P i implementaci se objevily i dal í problémy vlastní neuronovým sítím, o nichž se pozd ji také zmíníme. V echny v ak bylo možné e it drobnou úpravou vstup i výstup sít .

Testované varianty

Než jsme dosp li k finálnímu stavu neuronového simulátoru prezentovaného v této práci, pro li jsme následujícími fázemi vývoje:

1. **Dv sít**

   Abychom zjednodu ili práci neuronové síti, rozhodli jsme se pro dva moduly simulátoru, kde jeden bude simulovat jízdu vp ed a druhý rotaci. P vodní verze ízení robota byla navržena tak, že pro jízdu vp ed používala pro oba motory stejnou hodnotu PWM signálu, která byla pevn dána. Pro rotaci na míst sloužily jiné pevné hodnoty. Ob sít m ly stejnou architekturu:

   - dop edná vrstevnatá sí bez skrytých vrstev
   - sigmoidální p enosová funkce
   - Vstupem sít byly 2 hodnoty enkoder , výstupem 2 hodnoty enkoder v následujícím kroku (po provedení p íkazu pro motory). Pro kódování každé hodnoty sloužilo 100 neuron (Tento po et byl posta ující pro p esné zachycení rozsah používaných hodnot). Aktivace jednotlivých vstupních neuron odpovídala konkrétním diskrétním hodnotám vstup sít . V síti byl tedy vždy pro každou vstupní hodnotu aktivní práv jeden vstupní neuron. Pro hodnoty, které spadaly do intervalu mezi 2 neurony, jsme proporcionáln upravili aktivaci neuronu, který odpovídal vy í diskrétní hodnot vstupu. Sí tedy obsahovala jakési odd lené moduly pro jednotlivé hodnoty vstup . Výstupní hodnota sít odpovídala neuronu s nejvy í aktivací na výstupní vrstv .
   - 200 vstupních a 200 výstupních neuron
   - algoritmus zp tného í ení pro u ení sít
   - 50 prvk trénovací množiny pro každou sí

   Vstupy sít byly p ír stky enkoder v kroku *n*. Výstupy byly p ír stky enkoder v kroku *n + 1*. Sí nepot ebovala mít na vstupu p íkaz na motory, protože ty byly pro daný zp sob pohybu vždy stejné a podle nich se p edem rozhodlo, jaká sí se použije a p ípadn se upravily vstupy a výstupy (inverze v p ípad rotace).

   Výhodou tohoto p ístupu byla jednoduchost otestování správné funkcionality. Velkou nevýhodou se ukázala být nemožnost simulace

p echod mezi zp soby pohybu — nap íklad když robot jede vp ed a pak p ijde p íkaz na rotaci, má ve skute nosti jízda vp ed ur itou setrva nost. Robot se tedy neza ne to it ihned na míst . V této variant simulátoru to ov em nebylo možné docílit.

Po zkonstruování reálného robota a testech se také ukázalo, že nep esnosti v uložení kol a malé rozdíly jinak stejných typ motor zp sobují, že oba motory se to í po zadání stejné hodnoty PWM r znou rychlostí a navíc tato rychlost není ve v ech testech stejná — krom jiného závisí i na stavu baterie robota.

2. Spole ná sí s p íkazy pro motory

Kv li e ení vý e zmín ného problému s r znými vlastnostmi pohonných soustav bylo t eba vložit mezi p íkaz strategie pro motory a ovládání motor softwarový PID regulátor [15] , který m ní PWM hodnoty pro jednotlivé motory plynule a nezávisle tak, aby výsledná rychlost robota byla pokud možno stejná jako požadovaná. To ov em zkomplikovalo simulaci v tom, že se nyní mohly hodnoty PWM plynule m nit tak, jak je nastavoval simulátor a již nebylo možné používat zvlá tní sí pro každou kombinaci hodnot.

Vytvo ili jsme tedy pouze jednu univerzální sí , které na vstupu p ibyly p íkazy pro oba motory (PWM signál), s následující architekturou:

- dop edná vrstevnatá sí bez skrytých vrstev
- sigmoidální p enosová funkce
- stejné kódování vstup a výstup jako v p ede lém p ípad
- Vstupem byly 2 hodnoty enkoder a dv hodnoty PWM (p íkazy pro motory), výstupem 2 hodnoty enkoder v následujícím kroku.
- 400 vstupních a 200 výstupních neuron
- algoritmus zp tného í ení pro u ení sít
- 200 prvk trénovací množiny

Tato sí sice vy e ila problém s d ív j í nutností samostatného modulu pro každou kombinaci p íkaz na motory, ne e ila v ak je t nereálnost simulace vzniklou ignorováním setrva nosti robota v p echodech mezi akcemi. Takto navržená sí nem la nap íklad možnost odli it stav, kdy robot jede trvale vp ed a rázem se rozhodne rotovat (setrvaností by se m l tedy stále pohybovat vp ed), od stavu, kdy rotuje neustále(žádná setrva nost sm rem vp ed již pohyb neovliv uje). P itom je z ejmé, že v každém z obou stav by m la být výsledná rychlost robota jiná.

P i implementaci této sít se také projevil problém s na í metodou kódování vstup . Z její podstaty je t eba velké množství neuron . Sí se proto u í pomalu a jen na tení vah již nau ené sít do pam ti trvá nezanedbateln dlouho, což jde proti jednomu z na ich požadavk na simulátor — m l by být rychlý.

3. Spole ná sí s p íkazy pro motory a zrychlením

Proto jsme v dal ím pokusu opustili p vodní metodu kódování a za ali jsme testovat architekturu sít , kde každý vstupní neuron reprezentuje jeden vstup. To vyžaduje mapování skute ných hodnot vstup (nap íklad hodnoty enkoder mohou nabývat celých ísel od -70 do 70) do intervalu (0;1), tedy p ípustných hodnot pro vstupní neuron. Podobn je t eba zacházet samoz ejm i s výstupy.

Tato metoda se ukázala jako výhodn j í. Díky výraznému úbytku neuron jsme mohli za ít testovat i vícevrstvé architektury sít . Po pokusech jsme jako nejvýhodn j í podle kvality simulace a rychlosti u ení vyhodnotili sí s jednou skrytou vrstvou obsahující 15 neuron .

Dal í zm nou v tomto modelu bylo zavedení dal ích dvou vstup , reprezentujících zrychlení obou kol robota. Od toho jsme o ekávali, že bude možné simulovat setrva nost reálného robota, což se potvrdilo. Celková architektura této sít je:

- dop edná vrstevnatá sí s jednou skrytou vrstvou
- sigmoidální p enosová funkce
- Kódování, resp. dekódování vstup , resp. výstup , probíhalo tak, že jeden neuron odpovídal jedné vstupní hodnot a jeho aktivace byla úm rná této hodnot .
- Vstupem byly 2 hodnoty enkoder , dv hodnoty PWM a 2 hodnoty zrychlení jednotlivých kole ek. Výstupem sít byly 2 hodnoty enkoder v následujícím kroku.
- 6 vstupních, 15 skrytých a 2 výstupní neurony
- algoritmus zp tného í ení pro u ení sít
- 200 prvk trénovací množiny

Robot simulovaný touto sítí se ukázal jako dostate n v rné zastoupení reálného robota, proto jsme tuto sí nakonec použili i ve výsledném e ení.

Úprava vstup a výstup sít

Jak bylo uvedeno vý e, lze sí nau enou na p edpovídání pohybu robota použít vcelku p ímo a e v ídícím systému robota pro jeho simulaci. K plné použitelnosti takového simulátoru bylo nutné provést je t následující kroky:

- Simulátor nemá žádné prost edky pro e ení kolizí s mantinelem arénky ani s predátorem. Kolize tedy e íme napevno tak, že se robot zastaví, je-li k p ekážce p íli blízko (tj. výstupem simulátoru jsou nulové rychlosti kole ek). Takto reprezentované kolize lze i v simulátoru detekovat nap íklad virtuálním nárazníkem.
- Neuronová sí simulátoru m la problém p edpovídat správn hodnoty enkoder , byly-li vstupní hodnoty enkoder blízké nule. To lze vysv tlit tak, že mezi pomalou jízdou vp ed a vzad je z hlediska p ír stk na enkoderech jen malý rozdíl. Z pohledu výsledného chování je to ale pro pozorovatele rozdíl výrazný. Proto jsme do simulátoru zavedli jakýsi prvotní impuls, který p i malých hodnotách enkoder (nap íklad p i

rozjížd ní robota) nastavil jako vstup sít malou nenulovou hodnotu enkoder ve správném sm ru, i když byla p vodn nulová.

- Dal ím problémem plynoucím z nedokonalého chování neuronové sít v okolí nulových hodnot vstupních enkoder byla skute nost, že sí patn p edpovídala finální fázi bržd ní. Simulovaný robot typicky nedokázal vyvinout p esn nulovou rychlost p i nulových požadovaných rychlostech na obou kole kách, ale rychlost kolísala v okolí nuly. Toto e íme nastavením rychlosti robota na nulu, je-li požadovaná rychlost nulová a výstupy enkoder jsou velmi malé.
- V nau ené síti se vyskytují stavy, ve kterých se vstupní hodnoty enkoder rovnají výstupním. To ov em znamená, že se robot p i stejných požadovaných rychlostech chová vždy stejn , tedy dostane-li se do takového stavu, už jej nikdy neopustí, dokud se nezm ní požadovaná rychlost. Tyto stavy vedou na p íli um lá chování (robot se pohybuje bu po p esné kružnici nebo p ímce), proto se je snažíme eliminovat p idáním malého umu na vstupní hodnoty enkoder pro sí . Tento um v podstat odpovídá i nep esnostem v m ení stavu a vykonávání p íkaz reálného robota.

Shrnutí

Shrneme-li vlastnosti vý e popsaného neuronového simulátoru, dostáváme následující výhody a nevýhody:

Výhody:

- rychlost
- snadná p enositelnost na jiného robota
- odpovídá rozumn realit

Nevýhody:

- obtížná ov itelnost správné funkcionality
- sám o sob ne e í kolize
- nelze v n m simulovat více robot

Tento simulátor je velice vhodný pro simulaci jednoho diferenciáln ízeného robota v jednoduchém prost edí, kde není t eba zvlá tním zp sobem e it kolize objekt . Simulátor vnímá robota jako kruh o polom ru, který odpovídá vzdálenosti p i které má nastat kolize. Druhým parametrem, který je t eba nastavit je po et tik enkoderu za jednotku ujeté vzdálenosti. T etím a posledním parametrem je vzdálenost pohán ných kol od sebe. Robot v simulátoru je pak charakterizován t mito t emi parametry a souborem vah vý e popsané neuronové sít . Pro budoucí roz í ení simulátoru tak, aby nebylo pot eba žádné nastavování parametr robota a sta ilo jen projetí kalibra ní dráhy se nabízí následující metoda.

Jak bude popsáno pozd ji v této práci, na e testovací prost edí se skládá mimo jiné z pevn umíst né kamery, pomocí níž dokážeme analyzovat pohyb robota v testovací arénce. B hem projetí kalibra ní dráhy lze tedy zaznamenávat absolutní robotovu pozici kamerou a pak jednodu e z uložených informací o zm nách hodnot enkoder dopo ítat jak vzdálenost kol, tak rozli ení enkoder . Dojde-li

b hem kalibra ní jízdy ke kolizi s mantinelem arénky, lze odhadnout i polom r robota.

4.2 Simulace predátora

Sou ástí prost edí, v n mž se má RoboKrysa pohybovat, je predátor. V p ípad reálných test s potkany jím m že být jak dal í potkan, tak i robot. My jsme provád li testy pouze s robotem. RoboKrysa o n m ví z po átku jen to, že je to objekt, který se m že pohybovat. Jeho dal í vlastnosti má možnost zjistit b hem svých reálných jízd.

Na ím požadavkem na u ení robota byl minimální po et reálných jízd. Tém celé u ení je tedy nutné provést v simulátoru. Aby byla RoboKrysa schopna se úsp n úkol nau it, je ale samoz ejm pot eba, aby se v simulátoru objevil i predátor. Navíc by jeho chování m lo být co nejpodobn j í predátorovi skute nému. K tomu, jak zajistit simulaci predátora s vlastnostmi podobnými reálu, jsme testovali následující metody:

1. **Predátor jako neuronovou sítí ízený robot**

   První, pozd ji opu t ná, metoda pohlížela na predátora jako na virtuálního robota, který se musí nau it n jakému konkrétnímu chování. Kvalita tohoto chování se dá m it jako míra podobnosti chování reálnému predátorovi. Tj. v podstat není nutné, aby se simulovaný predátor pohyboval po identické dráze s predátorem reálným (nikde není ani e eno, že se skute ný predátor v každém testu pohybuje po identické dráze), ale m l by se v podobných situacích zachovat podobn .

   Dokážeme-li tuto míru podobnosti chování m it, není je t zcela jasné, jak se má simulovaný predátor chovat. Jsme ale schopni ohodnotit kvalitu jeho libovolného chování. To je velice podobný problém jako u ení RoboKrysy — robota, který má úsp n e it úlohu, kterou p edem nezná. Když se ale o e ení pokou í, podle etnosti elektrických ok pozná, e í-li ji správn . Na základ tohoto srovnání jsme se tedy rozhodli implementovat ízení simulovaného predátora stejn jako ízení RoboKrysy (to bude popsáno pozd ji podrobn , proto ho nyní popí eme jen stru n).

   Rozhodování robota o dal ím postupu (strategii) zaji uje neuronová sí , která na základ vstup ze senzor vygeneruje následující akci pro robota. Nejprve je ov em nutné nastavit váhy této neuronové sít . Ty nastavíme genetickou evolucí vah — testujeme v simulátoru jednotlivé jedince s nastavenými r znými váhami sít a m íme míru jejich úsp nosti (nap . podobnost chování s reálným predátorem). Tuto míru používáme jako fitness funkci. V p ípad RoboKrysy je fitness funkcí etnost ok . Z vý e popsaného plyne, že ízení predátora i RoboKrysy m že být v podstat identické. Jen je t eba roboty geneticky vyvíjet s odli nou fitness funkcí. Zatímco u RoboKrysy je fitness funkce relativn jednoduchá a p ímo ará (po et ok), m it podobnost dvou chování je zna ný problém.

   Po zkoumání jiných postup jsme se rozhodli omezit pojem podobnosti chování na fakt, že simulovaný i reálný robot stráví v podobných oblastech arénky podobný as. To byl údaj který bylo možné získat jednodu e jak z jízdy predátora skute ného, tak z testovaní simulovaného predátora

p i genetickém u ení. P i tomto testování se projevila vlastnost genetického u ení, o které se ve velké v t in p ípad hovo í jako o výhod , v p ípad na em byla ov em spí e na kodu — abnormální schopnost vymý let jedince, kte í plní kritérium dané fitness funkcí velice dob e, ale úpln jinak, než tv rce algoritmu na za átku ekal. Jinými slovy — ukázalo se, že lze toto kritérium podobnosti splnit chováním, které v bec nesouvisí se skute ným zp sobem pohybu predátora.

2. Predátor jako p ehráva logu

Výzkum p edchozí metody velkým dílem p isp l ke kone nému navržení ídící neuronové sít a jejího genetického vývoje pro RoboKrysu. Pro predátora jsme v ak nakonec použili jiný zp sob, jak jeho chování zprost edkovat simulátoru prost edí.

Reálných jízd RoboKrysy využíváme mimo jiné k získání záznam o p esném pohybu predátora (RoboKrysa vnímá pozici predátora b hem jízdy). Náhodné ásti t chto záznam pak p ehráváme p i u ení RoboKryse. Náhodností zaji ujeme, aby se nenau ila na konkrétní chování predátora, které se m že v jiném testovacím p ípad zm nit. Kolize s RoboKrysou e íme zastavením p ehrávání záznamu (tedy zastavením predátora), dokud se RoboKrysa sama z kolize nedostane. Tato metoda je velice jednoduchá, p itom dostate n v rn simuluje predátora.

4.3 Šokova

Dal í sou ástí testovacího prost edí pro RoboKrysu je tzv. okova . Jedná se o za ízení, které na základ polohy predátora a robota rozhodne o tom, zda má robot dostat elektrický ok. Je to pro na eho robota dal í neznámá prost edí, kterou se musí nau it, aby zvládl úlohu.

V zadání základní úlohy, kterou se má RoboKrysa nau it, rozhoduje okova o oku na základ vzdálenosti RoboKrysy od predátora. Je-li men í než 25 cm, dostane robot ok. Obecn ale m že být podmínka pro ud lení oku libovolná, proto ji RoboKrysa nemá p edem zadanou, ale musí být schopna chování okova e zjistit. Rozhodování okova e jsme omezili na dv oblasti — o ud lení oku m že rozhodovat bu podle vztahu pozice RoboKrysy a predátora, nebo podle absolutní pozice RoboKrysy.

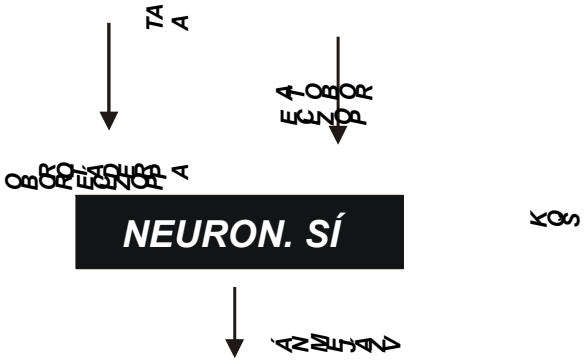
Šokova definuje úlohu, kterou se má RoboKrysa nau it. Informace od n j je jediná zp tná vazba, kterou robot o správném i patném pln ní úlohy má. Elektrický ok je pro potkana a (z definice) i pro robota nep íjemný a snaží se mu vyhýbat. Je z ejmé, že potkan i robot musí nejprve n kolikrát ok dostat, aby m li možnost vytvo it si souvislost mezi chováním okova e a chováním svým. Problém s vytvo ením této souvislosti pro RoboKrysu plynoucí z toho, že informace o patném pln ní úlohy (tedy ok) následuje p íli pozd a není tedy z ejmé, jaká akce do patné situace vedla, je popsán v kapitole 3 o použití neuronových sítí a genetických algoritm v evolu ní robotice[1] .

P i reálných testovacích jízdách robota je okova prost edí napevno naprogramován na úlohu, kterou se má robot nau it. RoboKrysa ale samoz ejm

p edem neví (stejn jako skute ný potkan), za co bude ok následovat. To musí nejprve zjistit. Proto si robot v prvních testovacích jízdách pouze ukládá informace o situacích, ve kterých do lo k oku, ale nesnaží se t mto situacím nijak p edcházet, aby co nejlépe pokryl stavový prostor úlohy.

U ení strategie RoboKrysy probíhá v simulátoru. K úplné simulaci úlohy je ale pot eba i kvalitní simulátor okova e, který ov em RoboKrysa nemá pro obecn neznámou úlohu p edem k dispozici. Jediné její znalosti o reálné úloze jsou záznamy z uskute n né trénovací jízdy (pam u potkana). Tyto záznamy použije RoboKrysa k vytvo ení modelu fungování okova e (uv dom ní si podstaty úlohy u potkana). Tento model pak používá místo skute ného okova e p i u ení v simulátoru (potkan nalezne vhodné e ení, když ví, v em úloha spo ívá).

RoboKrysa má proto zabudovaný mechanismus, jakým si dokáže vytvo it model fungování okova e ve své pam ti. Ten tvo í neuronová sí , která na základ vzájemné pozice RoboKrysy a predátora nebo absolutní pozice RoboKrysy rozhoduje o tom, zda má v této situaci vygenerovat ok i nikoliv.



Obr. 7 Funk ní schéma neuronové sít okova e

Trénovací množinu pro tuto sí tvo í data z logu nasbíraného b hem testovací jízdy. V n m je tedy uložena pro každý asový okamžik testu pozice a orientace obou pohybujících se jedinc a stav okova e. Data do trénovací množiny vybíráme z náhodných úsek logu, aby množina obsahovala pravd podobn stavy z rozli ných konfigurací jedinc v aréne. Architektura neuronové sít okova e je následující:

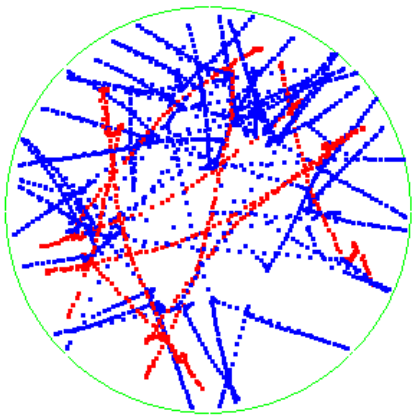
- dop edná vrstevnatá sí bez skrytých vrstev
- sigmoidální p enosová funkce
- 6 vstupních a 1 výstupní neuron
- algoritmus zp tného í ení pro u ení sít
- 500 prvk trénovací množiny

Vstupy této sít tvo í:

- x-ová sou adnice robota (absolutní pozice)
- y-ová sou adnice robota
- vzdálenost robota od st edu arénky (relativní pozice v aréne)
- orientace robota v i spojnici jeho pozice se st edem arénky

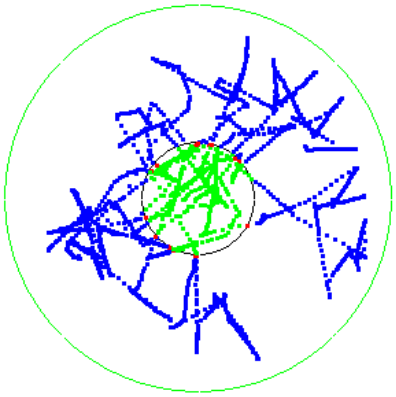
- vzdálenost robota od predátora (relativní pozice v i predátorovi)
- úhel, pod jakým robot vidí predátora

Na následujícím obrázku jsou znázorn ny trajektorie pohybu RoboKrysy a predátora b hem t íminutové reálné jízdy v kruhové aréncе o pr m ru 85 cm. Tuto jízdu jsme použili pro nasbírání dat pro trénovací množinu okova e. Robot b hem ní dostával oky, p iblížil-li se predátorovi blíž než na 25 cm.



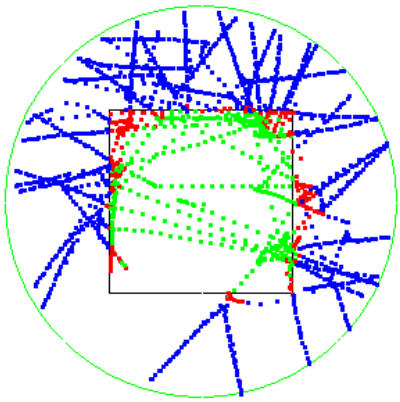
Obr. 8 Trajektorie RoboKrysy a predátora b hem reálné jízdy. Mod e je znázorn na dráha RoboKrysy, erven dráha predátora.

Následující obrázek znázor uje jinou reálnou jízdu, kde je zobrazena trajektorie RoboKrysy relativn k predátorovi, aby byla z ejmá souvislost chování okova e se vzdáleností RoboKrysy od predátora (st ed p edstavuje predátora). Úkol byl stejný jako v p edchozím p ípad . Chybovost nau eného okova e na tuto úlohu se pohybuje v rozmezí 1-2%.



Obr. 9 innost okova e v závislosti na vzdálenosti RoboKrysy od predátora. Predátor je umíst n ve st edu schématu (tj. v po átku sou adné soustavy). Mod e vyzna ená je dráha RoboKrysy bez oku, o které rozhodl již nau ený okova a rozhodnutí bylo správné. V zelené ásti trajektorie okova správn rozhodl o vygenerování oku. V ervených bodech se nau ený okova zmýlil. erná kružnice uprost ed ozna uje hranici 25 cm od predátora.

Následující obrázek znázor uje trajektorii jiné jízdy RoboKrysy v kartézských sou adnicích. Úkolem bylo tentokrát vyhýbat se tvercové oblasti o hran 40 cm uprost ed arénky. Chybovost okova e byla na této úloze pr m rn 9%. I to ov em posta uje k tomu, aby se pomocí tohoto okova e RoboKrysa nau ila inteligentn tuto úlohu e it.



Obr. 10 innost okova e v kartézských sou adnicích. Robot se m l vyhýbat oblasti vyzna ené erným tvercem. Ostatní barvy mají stejný význam jako na p edchozím schématu.

A koliv jsou možnosti okova e zna n omezené (hranice t ídy úloh, které je teoreticky schopný se nau it, je vid t už z jeho vstup), tato konstrukce nám zcela posta uje k pokus m se základními úlohami, které se testují na potkanech. P i pot eb u it robota i úlohy, které stávající okova není schopen pojmout, ho není problém p i jeho jednoduchém napojení do systému roz í it o dal í vstupy, i dal í vrstvy neuronové sít . Šokova je jedním ze dvou základních prvk adapta ního systému robota. O druhé sou ásti pojednává následující podkapitola.

4.4 Genetická strategie

Máme-li k dispozici funk ní okova nau ený na úlohu a kvalitní simulátor testovacího prost edí úlohy a predátora, dokážeme provád t testy virtuáln s podobnou informací o pr b hu (oky a pozice predátora) jako p i testech reálných. Zbývá navrhnout zp sob, jak v simulovaném prost edí nau it robota e it zadanou úlohu.

K tomu jsme se rozhodli využít metodu genetických algoritm . Jak bylo popsáno v kapitole 3.1 o použití genetických algoritm v evolu ní robotice, je tato metoda velice vhodná, nemáme-li k dispozici pro každý krok požadovaný výstup, ale existuje pouze informace o tom, jak kvalitní bylo celkové chování. A to je p esn ná p ípad — lze jen velmi t žko íct, který krok robota vedl do patné situace a kde konkrétn by se m lo chování zm nit. Hodnotit a vyvíjet lze jen celek. Základní my lenka genetické evoluce strategie robota byla tedy následující:

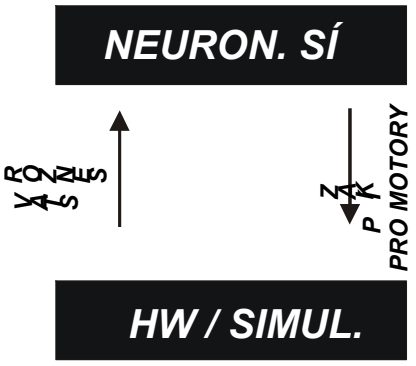
- Jedna instance robota je jeden jedinec v populaci genetického algoritmu.
- Pro výpo et fitness každého jedince provedeme simulovanou jízdu a zjistíme po et ud lených ok a „sn deného“ zrní.

K implementaci této my lenky bylo t eba navrhnout ízení robota tak, aby jeho chování bylo ur ováno n kolika parametry. Vý e popsaný genetický algoritmus by pak generoval tyto parametry, ímž by efektivn generoval r zné strategie pro RoboKrysu.

Jako ideální metoda, ve které lze parametricky ovliv ovat výstupy generované z konkrétních vstup , se zdály být op t neuronové sít . Rozhodli jsme se tedy jít tímto sm rem. Jak již bylo napsáno, tentokrát jsme nemohli použít klasické metody zp tného í ení pro nau ení sít , protože nebyla možnost získat jakoukoliv trénovací množinu. Pro adaptaci vah sít jsme využívali vý e zmín ného genetického algoritmu [5] . Pro nalezení zp sobu, jak sí napojit do systému robota a jak ji navrhnout, jsme li dv ma cestami, které nyní popí eme.

4.4.1 Low-level p ístup

První zp sob, který jsme implementovali a otestovali, byla neuronová sí , jejímž vstupem byly údaje ze senzor a výstupem hodnoty dále používané jako vstupy motor . Ne lo ov em o klasický model tzv. Breitenbergova autí ka[§] [1] , protože v na em p ípad jsou údaje ze senzor výrazn komplexn j í a nelze navrhnout p ímé napojení jednotlivých senzor váženým spojem na konkrétní motor.



Obr. 11 Schéma neuronové sít jako low-level ízení robota

Architektura na í neuronové sít byla následující:

- dop edná vrstevnatá sí bez skrytých vrstev
- sigmoidální p enosová funkce
- 6 vstupních a 2 výstupní neurony
- sí se u í genetickým vývojem vah

Vstupy této sít tvo ily:

- vzdálenost robota od st edu arénky
- orientace robota v i spojnici jeho pozice se st edem arénky

[§] Breitenbergovo autí ko je jednoduchý model robota, ízeného neuronovou sítí. Výstupy ze senzor tvo í p ímo vstupy sít a jsou váženými spoji napojené na motory. Jedná-li se o vhodn umíst né vzdálenostní senzory, lze takto implementovat nap íklad jednoduchou strategii pro vyhýbání se p ekážkám.

- vzdálenost robota od predátora
- úhel, pod jakým robot vidí predátora
- informace o tom, skon ila-li poslední akce kolizí
- informace o tom, skon ila-li poslední akce okem

Výstupy byly dv hodnoty ur ující požadovanou sílu na motorech.

Tímto „low-level“ e ením jsme cht li otestovat jeho možnosti pro dal í použití, protože v p ípad funk nosti by nabízelo velice jednoduchý a rychlý zp sob ízení robota neuronovou sítí. e ení jsme testovali na úloze collision-avoidance (vyhýbání se p ekážkám)[**]. Geneticky vyvinutý robot dosahoval v simulátoru dobrých výsledk . Problém byl ov em s p echodem do reálu. Vyvinuté e ení bylo p íli závislé na p esném chování robota v simulátoru a rozdíly v chování reálného robota mu vadily natolik, že bylo prakticky nepoužitelné.

Pro zmen ení závislosti nau eného chování na p esných vlastnostech robota jsme zavedli do na eho systému dv opat ení:

1. **Šum na enkoderech**

   Neuronový simulátor p edpovídá nové hodnoty enkoder z hodnot p edchozích a z p íkaz pro motory. Abychom snížili závislost chování robota na p esných vlastnostech tohoto p edpovídání, p idáváme k ob ma p edpov eným hodnotám s ur itou pravd podobností náhodnou hodnotu. Od toho jsme si slibovali v t í robustnost nau eného e ení — aby nefungovalo jen pro robota s konkrétními vlastnostmi, ale tyto vlastnosti mohli ležet v ur itém rozsahu.

2. **Testování ve více situacích**

   Abychom p ede li v genetické evoluci vysokému ohodnocení vzork , které díky astné shod okolností v jednom z testovacích p ípad usp ly, ale e ení není schopné se uplatnit v jiných p ípadech, zavedli jsme do genetického u ení to, že každý vzorek budeme testovat ve t ech r zných p ípadech a fitness spo ítáme ze v ech t chto test . Takže genetický algoritmus bude preferovat spí e vzorky, které jsou schopny usp t bez ohledu na konkrétní konfiguraci testu.

Bohužel, ani tato opat ení nesta ila k dostate né p enositelnosti nau eného chování ze simulátoru do skute ného robota. Jako možné e ení tohoto stavu se nabízelo provád t poslední ást genetického vývoje na reálném robotovi. Toto je ale v kolizi s po áte ními požadavky. Rozhodli jsme se jít tedy jinou cestou.

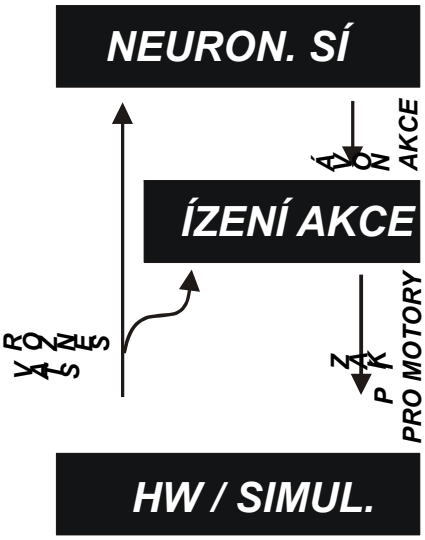
4.4.2 Hi-level p ístup

V p edchozím p ípad m la ídící neuronová sí vlastn dva úkoly — ízení pohybu robota a ízení strategie. ízením pohybu zde myslíme nap íklad to, jak zajistit, aby jel robot rovn , když strategie rozhodne, že je to v danou chvíli žádoucí

[**] Robot má za úkol projet co možná nejdel í dráhu s co nejmén kolizemi s mantinelem arénky, nebo s predátorem.

chování. Strategii chápeme jako vy í vrstvu, která rozhoduje o koordinaci základních akcí tak, aby vedly k úsp nému zvládnutí úlohy.

Problém p edchozího p ístupu byl v neodd litelnosti ízení a strategie. Z pozorování bylo z ejmé, že kdyby bylo v simulátoru správn zvládnuto ízení (reálný robot by jezdil stejn rovn a stejn zatá el jako v simulátoru), byli bychom pravd podobn schopni vyvinout strategii tak, aby byla p enositelná. Proto jsme p idali dal í vrstvu do modelu ízení. Tato vrstva má za úkol sjednotit chování simulovaného a reálného robota.



Obr. 12 Schéma neuronové sít jako hi-level ízení robota

Nejprve jsme definovali ty i „základní stavební kameny“ — chování, která jsou úkolem vrstvy ízení. Vrstva strategie tyto stavební kameny transparentn používá. Strategie tedy nemusí rozli ovat mezi simulovaným a skute ným robotem. P ibližnou invariantnost základních chování p i p echodu mezi reálem a simulátorem jsme zajistili regula ními metodami mobilní robotiky. Implementace t chto chování je blíže popsána v kapitole 5 o implementaci software robota. Jednotlivá chování a tedy i celá strategie nejsou závislá na absolutní pozici robota a predátora. Strategie rozhoduje o jednotlivých akcích podle relativní polohy jedinc v i sob .

Typy akcí:

- jízda vp ed
- zatá ení na míst vzhledem k predátorovi
- zatá ení na míst vzhledem k aréne
- ekání

Cílem neuronové sít je ur it typ a parametr akce, jež se má vykonat v p ípad , že p edchozí skon ila, nebo byla p eru ena (okem nebo kolizí). Architektura této neuronové sít je podobná se sítí pro low-level variantu ízení, li í se jen v po tu a interpretaci výstup .

Sí má tedy 5 výstup s následujícím významem:

- následující akce
- vzdálenost, kterou má ujet akce „jízda vp ed“
- úhel, na který se má oto it akce „zatá ení na míst vzhledem k predátorovi“ (úhel od spojnice pozice robota a predátora)
- úhel, na který se má oto it akce „zatá ení na míst vzhledem k aréncе“ (úhel od spojnice pozice robota a st edu arénky)
- doba, po kterou má ekat akce „ ekání“

Tato metoda, v kombinaci s vý e popsaným vysokým umem na enkoderech a testováním každého vzorku ve více situacích, vedla na robustní a p enositelné e ení adaptace robotova chování. Proto jsme ji použili i ve výsledném e ení. Pro implementaci genetického u ení jsme použili ná systém GeneticLab, který bude popsán pozd ji i s postupem u ení, které zaji uje. Pro u ení jsme použili populaci 100 robot . Tento po et se ukázal jako posta ující a doba nutná pro nau ení jedince se omezila na p ibližn 3 hodiny. Na vzorcích vah neuronové sít jsme neprovád li p i vytvá ení nové populace k ížení, protože kombinace vah r zných neuronových sítí nedává dobrý smysl. Mutaci jsme implementovali jako náhodné vynásobení nebo vyd lení 10% vah vzorku vybraného k mutaci náhodným íslem.

Ohodnocení jedince probíhá tak, že GeneticLab spustí externí proces ízení robota, pro který ale p edem p ipraví konfigura ní soubor s váhami strategie. Robot se spustí v simulátoru t ikrát (pokaždé s jinou trajektorií predátora) a do souboru si zaznamenává po et ok a po et „sn deného zrní“. Z tohoto souboru po dokon ení testování jednoho jedince GeneticLab spo ítá výslednou fitness.

4.5 Celkový postup u ení

P edpokládejme, že je systém pro u ení pln funk ní a máme definovánu úlohu v definovaném prost edí. Dále máme robota ízeného adaptabilním systémem pro dané prost edí, který ale je t o úloze nic neví. V této podkapitole shrneme cestu, kterou je t eba b hem u ení projít, aby se robot na zadanou úlohu adaptoval.

1. **Pr jezd kalibra ní dráhy**

   Nejprve je t eba zkalibrovat neuronový simulátor podle skute ného robota, aby se simulace co nejvíce p iblížila realit . K nasbírání dat pro trénovací množinu neuronového simulátoru jsme používali kalibra ní dráhu mimo testovací arénku, sestávající z 20 základních akcí, rozvržených tak, aby mezi nimi byl stejný po et rotací na ob strany a p echody mezi r znými stavy. Trénovací množina tak pokryla velký po et stav , které mohou b hem reálné jízdy nastat. Kalibra ní dráhu lze definovat i v testovací aréncе a využít p i tom p ipojeného kamerového trasovacího systému k nastavení konstant definujících robota.

   K pr jezdu kalibra ní dráhy používáme ízení robota s odli nou nejvy í vrstvou (ízením strategie) od té, která se používá u robota ízeného

neuronovou sítí. Tato varianta ízení robota s p eddefinovanou strategií je popsána v kapitole 5 o implementaci robota.

2. Nau ení neuronového simulátoru

Po napln ní trénovací množiny z ní m žeme nau it neuronový simulátor konkrétního robota. K tomu slouží samostatný program SimulatorLearning odvozený ze systému NeuralNetwork, který vygeneruje soubor s hodnotami vah neuronové sít simulátoru. K u ení používá pevn nastavený po et 1000 cykl po 100 iteracích. Nyní máme k dispozici funk ní simulátor jízdy na eho robota.

3. Nasbírání dat pro okova

M žeme tedy p istoupit k prvnímu reálnému pokusu ve skute ném testovacím prost edí v plné konfiguraci — tedy v aránce, s predátorem, s okova em pevn nastaveným na zadanou úlohu a s kompletn zapojeným snímacím systémem. Op t používáme ízení robota s pevnou strategií. Tentokrát je strategie nastavena na náhodné ježd ní v aránce po dobu 6 minut. Robot jezdí vždy vp ed a p i kolizi (s mantinelem i predátorem) couvne, náhodn se oto í a pokra uje. P i této jízd se zaznamenávají konfigurace prost edí, p i kterých do lo k elektrickému oku. Robot se nijak nesnaží kolizím ani ok m p edcházet.

4. Nau ení okova e

Z nasbíraných dat se vytvo í trénovací množina pro neuronovou sí simulátoru okova e. Nyní m žeme programem ShockerLearning, rovn ž vytvo eným ze systému NeuralNetwork, nau it okova . K u ení používáme tentokrát pevn nastavený po et 5000 cykl po 100 iteracích. Výsledkem je op t samostatný soubor vah, který budeme nadále používat v na em simulátoru. V tomto okamžiku je tedy ná simula ní systém kompletní a je p ipraven k provád ní kompletních simulovaných pokus .

5. Genetická evoluce strategie

Nyní poprvé použijeme ízení robota, jehož nejvy í vrstvu zaji uje neuronová sí . Platformou pro robota v této fázi bude neuronový simulátor. Program NeuroRobutecLearning, odvozený ze systému GeneticLab, ídí genetický vývoj parametr strategie tak, že spou tí robota s r znými nastaveními parametr a vyhodnocuje kvalitu e ení úlohy b hem testu. Podle toho se rozhoduje o dal ím sm ování evoluce vah. Genetický vývoj je automaticky zastaven po 100 generacích, kde každá má 100 vzork . Instanci strategie, která e í zadanou úlohu, získáme tak, že z poslední generace vezmeme nejúsp n j ího jedince (váhy) a toho použijeme ve výsledném robotovi.

6. Otestování funk nosti nau eného e ení

K provedení reálného testu nau eného e ení použijeme reálné testovací prost edí v plné konfiguraci a ízení robota, stejné jako v p edchozím p ípad , s tím rozdílem, že tentokrát bude platformou skute ný robot.

5 Implementace

V této kapitole budeme prezentovat konkrétní metody, které jsme použili b hem realizace test . V první ásti popí eme vývoj a kone nou verzi RoboKrysy a jejího ízení. Druhá ást se krátce zmíní o testovacím prost edí a p ídavných za ízeních, které jsme b hem test používali. Ve zbytku této kapitoly popí eme dva systémy, ve kterých jsme realizovali adaptabilní ásti softwaru ízení robota.

5.1 RoboKrysa

RobuTecIV, p ezdívaný RoboKrysa, je robot, kterého používáme jako ko ist v na em systému. Vyvíjeli jsme jej cílen pro tento ú el s následujícími požadavky:

- jednoduchost - Protože hlavním cílem této práce bylo zkoumání metod evolu ní robotiky a ne vyvíjení robot , omezili jsme se na nejjednodu í e ení, které bylo použitelné pro na e ú ely. Požadavek jednoduchosti navíc úzce souvisí se v emi dal ími...
- spolehlivost - Jednoduchou hardwarovou architekturou a použitím ov ených postup jsme cht li zajistit, aby byl robot co možná nejmén náro ný na údržbu.
- replikovatelnost - Protože se chceme oblastí evolu ní robotiky zabývat i v budoucnu, a to i nap íklad otázkou evoluce skupin robot , p edpokládáme, že budeme pot ebovat více podobných za ízení. Proto jsme p i návrhu RoboKrysy brali v úvahu jen sou ásti, které jsou voln dostupné a nebude problém s jejich použitím na dal ích robotech.
- nízké náklady - Nezanedbatelnou otázkou byla i nízká cena robota, která nám dovoluje provád t podobné experimenty z vlastních zdroj .

RoboKrysa ale nebyla prvním robotem, který jsme pro na e experimenty použili. Proto následující ást textu v nujeme krátkému popisu vývoje, který jsme pro li od p vodního robota k sou asné RoboKryse.

5.1.1 Vývoj

První robotickou platformou, se kterou jsme uvažovali pro na e experimenty byl RobuTecIII. Jedná se o diferenciáln ízeného robota se základní deskou s jedno ipovým mikrokontrolérem Atmel AT90S8535. Jeho p dorys tvo í tverec o hran 16,5 cm. Do systému byl napojen stejn jako RoboKrysa, m l i velice podobné ízení.

Na tomto robotovi jsme provád li testy s fyzikálním simulátorem a pozd ji i první testy p i vývoji neuronového simulátoru. P i jeho používání ale bylo ím dál tím víc jasné, že as strávený softwarovým kompenzováním problém zp sobených jeho patnou konstrukcí je vy í než p ípadný as strávený vývojem nového robota. RobuTecIII totiž nebyl robot stav ný pro pohyb na malé vzdálenosti v uzav ené aréne. Jeho motory nebyly dost silné na kvalitní u ízení robota, proto se v reálných testech choval velice nedeterministicky, což velice zt žovalo možnost kvalitní simulace.

Jako e ení jsme tedy p i li s architekturou RoboKrysy. Je to robot o mnoho men í, a leh í, který neklade takové nároky na motory, jeho p dorys má tvar kruhu o pr m ru 10 cm, který robotovi velice usnad uje manévrování na malém prostoru, jakým testovací arénka je. Podstatn se také zvý ila výdrž robota na jedno nabití baterií.

5.1.2 Low-level architektura

Základní model ízení

RoboKrysa je ízena dv ma procesory. První, jedno ipový procesor AVR ATMEGA8 od firmy Atmel [19] je na základní desce robota a p ímo na jeho vstupy a výstupy jsou p ipojeny jednotlivé periferie robota. Tento procesor posílá jednotlivým za ízením signály, kterými je ídí a zárove v pravidelných intervalech te data ze senzor . T mito daty vypl uje strukturu, kterou pak na požádání posílá komunika ním modulem do druhého po íta e robota, standardního PC, na kterém b ží ídící aplikace.

Tato aplikace v každém kroku, kterých je 10 za vte inu, dostane strukturu odrážející stav senzor robota. Podle této informace, a obecn i p edchozího pr b hu jízdy, ur í modul strategie dal í postup pro robota. Vygenerovaný postup strategie jednozna n ur uje p íkazy pro hardware v následujícím kroku. Ty se po lou do AVR, kde se i provedou. Následuje dal í krok celého cyklu.

To, že ídící aplikace b ží na PC, které není p ímou sou ástí robota a tedy RoboKrysa není použitelná bez externího po íta e, není v rozporu s konceptem autonomie ízení RoboKrysy. Je to pouze implementa ní detail, který v ak podstatným zp sobem usnad uje práci s robotem. Pro implementaci RoboKrysy jako autonomního robota bez jakékoliv externí sou ásti je t eba jen dokonalej í hardware. Logika i podstatná ást kódu ízení by v ak z stala stejná. Výhody externího ízení jsou:

- Hardware robota m že být podstatn jednodu í a tedy i levn j í.
- Jednoduchý hardware umož uje i výrazné zmen ení rozm r robota, takže lze reálné testy provád t v jednodu ích podmínkách.
- ídící aplikace pracuje na standardním PC, což usnad uje její vývoj a napojení na testovací prost edí a monitorovací za ízení. V echny tyto sou ásti se spou tí na stejném po íta i.
- P i zm nách strategie i hi-level ízení robota není t eba složit programovat nestandardní procesory pro ízení hardwaru.

Ovládání HW v jedno ipu

Jedinou ídící jednotkou p ímo na robotovi je mikrokontrolér AVR ATMEGA8. Jedná se o osmibitový jedno ipový procesor s architekturou RISC, 8 KB p episovatelné pam ti na kód, 1 KB RAM a 23 vstupn /výstupními digitálními linkami, pomocí nichž se ovládají ve kerá hardwarová za ízení robota. Ta využívají i dal ích vlastností procesoru, jako nap íklad analogov digitální p evodník na esti linkách, hardwarový generátor PWM signálu na t ech linkách, asova e, atd.

Kód pro AVR jsme psali v jazyce C, p ekládali speciálním p eklada em avr-gcc a pomocí programu avr-dude p es programovací kabel nahrávali z PC do jedno ipu. AVR je vsazeno do námi navržené a vyrobené základní desky.

Základní funkcí programu v AVR je udržování struktury, která odráží stav hardwaru robota. Tuto strukturu procesor aktualizuje 500krát za vte inu v p eru ení na asova i. Aktuální strukturu AVR posílá synchronn pomocí Bluetooth-modulu do ídícího PC, když obdrží podobnou strukturu obsahující p íkazy pro hardware. Položky struktury odrážející stav hardwaru jsou:

- `unsigned char R_timer` – po et tik 500Hz asova e od za átku b hu programu s p etékáním („hodiny“ robota)
- `unsigned char R_encoder0` – po et tik enkoderu na levém kole ku; P i otá ení kole ka v jednom sm ru se k hodnot p i ítá, ve druhém sm ru od ítá (hodnota s p etékáním).
- `unsigned char R_encoder1` – to samé pro pravé kole ko
- `unsigned char R_digitalInputs` – byte obsahující obraz digitálních vstup na vstupn -výstupním portu základní desky; Digitálním vstupem lze detekovat nap íklad stisk aktivního nárazníku.
- `unsigned char R_analog[8]` – pole osmi analogových vstup ; N které piny I/O portu základní desky lze softwarov nastavit jako analogové vstupy. Ty p evádí analogový elektrický signál na digitální hodnotu od 0 do 255. Tuto vlastnost lze použít nap íklad p i zpracování signálu ze vzdálenostních idel.

Struktura obsahující p íkazy pro hardware se posílá z PC do AVR desetkrát za vte inu. AVR po každém p ijetí této struktury odpoví aktuálním stavem hardwaru. Položky struktury s p íkazy jsou:

- `unsigned char W_executeAt` - stav hodin AVR, ve kterém požadujeme, aby AVR odpov d lo stavem HW; Tedy as robota ídí hodiny v AVR a ne v PC, což umož uje udržet stejnou frekvenci komunikace i p i r zn dlouhých výpo tech strategie a stejné asování ídícího algoritmu nad reálným robotem i nad simulátorem prost edí.
- `char W_pwm0` - hodnota od -127 do 127 udávající velikost pln ní signálu PWM pro ízení levého motoru; P i kladné hodnot se motor otá í vp ed, p i záporné vzad odpovídající silou.
- `char W_pwm1` - to samé pro pravý motor
- `unsigned char W_watchDog` - kontrolní hodnota, od které se p i každém tiku 500Hz asova e ode te 1, a jakmile je hodnota na nule, robot se zastaví; Je to pojistka, aby robot, který p estal komunikovat s ídícím programem v PC, neza al nekontrolovan jezdit, ale zastavil se.
- `unsigned char W_servo[4]` - pole 4 hodnot od 0 do 255, které udávají nato ení jednotlivých serv na robotovi; V echna serva se ídí speciálním standardním signálem, který dokážeme generovat v AVR pomocí dal ích asova až pro 6 serv sou asn .

- `unsigned char W_digitalOutputs` - kontejner, ve kterém jednotlivé bity ur ují nastavení pin I/O portu základní desky robota, které jsou nakonfigurovány jako digitální výstupy.

Software pro AVR je navržen tak, aby ho pokud možno nebylo t eba modifikovat p i zm n hardwaru robota. Struktura pro p íkazy a stavy je natolik obecná, že ji lze využít pro ovládání tém libovolného jednoduchého robota se dv ma motory pro pohon a se standardními senzory. Samotné hi-level ízení robota (interpretace položek HW-struktury a generování p íkaz strategie) je postavené nad t mito obecnými strukturami.

ídící SW v PC

Vrstvu ídícího software robota, která pracuje na externím PC, jsme od za átku vyvíjeli a testovali pro opera ní systém MS WindowsXP. Jen velice málo ástí je ov em závislých na tomto opera ním systému, proto by jeho p enesení na jiný opera ní systém po malých úpravách nebyl problém. Kód je psán v jazyce C++. K p ekladu jsme využívali interní p eklada prost edí MS Visual C++ 6.0.

Schéma hlavního cyklu ízení robota v PC:

```
HardwareFilter *hw;
IRobot robot;
ISensorStatus status;
ICommand cmd;
Strategy *strategy;

hw = new HardwareFilter( new RealHardware() );
strategy = new Strategy;
status = hw->update(cmd);

while(!EndOfMatch)
{
  robot.position = updatePosition(status);
   robot.sensors = status;
   cmd = strategy->update(robot);
  status = hw->update(cmd);
}
```

Hlavním cyklem projde ídící program robota desetkrát za vte inu.

Popis základních t íd a struktur:

- **`HardwareFilter`**

  T ída `HardwareFilter` zprost edkovává p eklad na „jazyk strategie“ a naopak. Tedy interpretuje obecné položky stavové a p íkazové struktury pro ízení hardware jako konkrétní senzory konkrétního robota a naopak. Je-li nap íklad nastaven 4. bit v položce `R_digitalInputs`, `HardwareFilter` ví, že bit odpovídá stisku tla ítka na robotovi, které signalizuje náraz do zdi. Nastaví tedy prom nnou „NarazDoZdiDetekovan“ na hodnotu true. Na druhou stranu strategie vyprodukuje p íkaz pro motory — dop ednou a úhlovou rychlost. `HardwareFilter` ji p evede na hodnoty PWM pro jednotlivé konkrétní motory. Tato t ída zaji uje i synchronizaci komunika ní struktury mezi PC a AVR, resp. mezi ídícím

programem na PC a t ídou, která se zadá konstruktoru `HardwareFilteru`. Tuto t ídu máme v n kolika modifikacích:

- `RealHardware`
  Synchronizuje strukturu po Bluetooth lince s AVR (opravdovým robotem).
- `ServerIface`
  Synchronizuje strukturu se serverem fyzikálního simulátoru, ve kterém se pohybují virtuální roboti ve virtuálním sv t .
- `NeuralSimulator`
  Synchronizuje strukturu s neuronovým simulátorem.
- `Logger`
  T ída `RealHardware` p i své synchronizaci zaznamenává ve kerou prob hlou komunikaci do souboru. T ída `Logger` slouží pro p esné krokování procesu, který ídící program vykonal b hem reálné jízdy. Zadá se jí jméno logu a `Logger` pak vypl uje stavovou ást komunika ní struktury podle záznam v tomto logu a kontroluje, zda p íkazy generované strategií odpovídají p íkaz m, které generovala strategie p i reálné jízd . To umož uje krokovat práv prob hlou reálnou jízdu a odhalit, na základ eho se strategie rozhodovala.

Pro `HardwareFilter` je práce s t mito t ídami úpln transparentní.

- **`ISensorStatus`**

  `ISensorStatus` je t ída, kterou vypl uje `HardwareFilter` a která zprost edkovává strategii obraz stavu konkrétního robota.

  Položky pro RoboKrysu:

  - `double time` - as v sekundách od spu t ní robota
  - `ISpeed speed` - aktuální dop edná a úhlová rychlost robota
  - `bool collisionDetected` - p íznak, že robot detekoval „sepnutí“ virtuálního nárazníku
  - `bool shockDetected` - p íznak, že robot detekoval elektrický ok. Robot ve skute nosti nemá žádný senzor pro vnímání elektrických ok , ale bit digitálních vstup odpovídající tomuto p íznaku je nastavován p ímo testovacím prost edím v p ípad , že má dojít k oku.
  - `IPose predator` - pozice a orientace predátora. Tuto informaci robotovi zprost edkovává rovn ž testovací prost edí. Strategie se ale nerozhoduje podle této hodnoty. Z ní se pouze spo ítá relativní poloha predátora v i RoboKryse, která se pak p edkládá strategii.
  - `IPose rat` - pozice a orientace robota (tedy potkana). I tuto informaci robotovi zprost edkovává testovací prost edí, v p ípad , že tuto informaci má (že robota vidí kamera). Strategie nepracuje ani s touto hodnotou. Z ní se po ítá relativní pozice RoboKrysy v aréne.

- **ICommand**

  T ída `ICommand` slouží pro p edávání p íkaz ze strategie do `HardwareFilteru`, který ji pak p ekládá na stavovou strukturu posílanou do robota. Má pouze položku `ISpeed speed`, která reprezentuje požadovanou rychlost robota.

Framework strategie

Frameworkem strategie nazýváme prost edí, které jsme vytvo ili za ú elem snadného zapisování (a pozd ji automatického generování) strategií RoboKrysy. Základem tohoto frameworku je t ída `Strategy`. Tato t ída pracuje se stavem robota (`IRobot`), ve kterém je jeho pozice (kde si robot myslí, že je) a stav senzor (`ISensorStatus`). Jejím úkolem je generovat p íkazy (`ICommand`) pro `HardwareFilter`. Strategie dokáže provád t následující akce:

- `ActionGroup` - speciální akce, která obsahuje posloupnost jiných akcí
- `Driver` - robot jede bu vp ed nebo vzad o zadanou vzdálenost
- `Rotator` - robot se oto í o zadaný úhel nebo tak, aby jeho nato ení bylo v zadaném úhlu
- `Go2WayPoint` - robot jede na absolutní pozici zadanou v parametru této akce
- `Wait` - robot eká po zadaný po et vte in

V echny tyto akce jsou samostatné t ídy, které jsou zd d ny z rodi ovské t ídy `AbstractAction`, která jim dává jednotný interface, se kterým pak strategie pracuje. Každá akce (až na `ActionGroup`, která se už p i inicializaci rozbalí na vložené akce) tedy implementuje následující metody:

- `ICommand update(const IRobot &in_robot)` - základní funkce každé akce, která se volá v každém kroku hlavního cyklu ízení; Generuje p íkaz, který se má provést v následujícím kroku akce.
- `int isEnd(const IRobot &in_robot)` - metoda, která se zavolá v každém kroku cyklu ízení a oznámí, jestli aktuální akce skon ila; Skon ila-li, vrátí návratovou hodnotu d vodu skon ení akce (nap íklad standardní ukon ení, ukon ení kv li kolizi, ukon ení kv li elektrickému oku atd...).
- `void init(const IRobot &in_robot)` - inicializace akce; Akce se inicializuje je t t emi parametry typu `double`, které sta í na to, aby každé akci ekly v e, co musí v d t pro sv j b h.

T ída `Strategy` implementuje zásobník obsahující akce, které se mají vykonat. Vždy vyzvedne akci z vrcholu zásobníku, pak akci provádí a když akce skon í, vyzvedne ze zásobníku dal í akci.

Zápis strategie pro kalibraci m že vypadat nap íklad následovn :

```
Action CALIBRATION[] = {
  Action(STATE,Strategy::ST_CALIBRATION), //strategie je ve stavu kalibrace
  Action(GO_FWD,0.5),                     //je  vp ed 0,5 m
  Action(ROTATE,-Math::PI/2,1),           //oto  se na míst  o PI/2
  Action(GO_FWD,0.5),
  Action(ROTATE,Math::PI/2,1),
  Action(GO_FWD,0.5),
  Action(GO_BACK,0.5),                    //couvni o 0,5 m
  Action(WAIT,1)};                        // ekej 1 s
```

Pak sta í vložit v konstruktoru t ídy `Strategy` na zásobník skupinu akcí `CALIBRATION` a provedou se v ní obsažené p íkazy.

Základní funkcí t ídy `Strategy` je funkce `update`, která se volá z hlavní smy ky programu:

```
ICommand Strategy::update(IRobot &in_robot)
{
  ICommand cmd;
  cmd.init();
  int retVal;
  /* test, je-li aktuální akce platná a neskon ila-li už (nap íklad po
kolizi, nebo dosažení cíle).*/
   if(!m_currentAction || (retVal = m_currentAction->isEnd(in_robot)))
  {
    /*zareaguje na návratovou hodnotu. Nap íklad byla-li detekována kolize,
dá na zásobník akci „couvni o 30 cm"*/
    dispatchReturnValue(m_currentAction,in_robot,retVal);
    delete m_currentAction;

    /* vezme ze zásobníku dal í akci */
    m_currentAction = popAction();
    /* byla-li akce typu ActionGroup, dej na zásobník v echny podakce, které
       obsahuje */
    while(translateAction(m_currentAction))
      m_currentAction = popAction();
    /* inicializuj akci */
    m_currentAction->init(in_robot);
  }
  /* aktualizuj p íkazy generované akcí ze stavu robota */
  cmd = m_currentAction->update(in_robot);
  return cmd;
}
```

Jeden krok strategie má dv varianty:

- rozhodovací krok - V p ípad , že p edchozí akce skon ila, rozhodne o spu t ní, p ípadn vygenerování, dal í akce.
- aktualiza ní krok - Provede aktualizaci p íkaz pro hardware, aby vedly ke zdárnému dokon ení práv provád né akce. Tento krok se provede vždy — robot musí být vždy ve stavu provád ní n jaké akce.

Ve keré rozhodování o dal í akci strategie se vykonává ve funkci `dispatchReturnValue`, která na základ návratové hodnoty p edchozí akce vygeneruje dal í akce na zásobník. Existuje ada možností, jak tuto funkci implementovat. Dva zp soby, které jsme zvolili my, popisuje následující podkapitola.

5.1.3 Hi-level architektura

Autonomní robot, který má pevnou strategii pro vykonání úkolu, má metodu `dispatchReturnValue` implementovánu jako jednoduchou rozhodovací funkci na základ návratové hodnoty p edchozí akce:

```
switch(retVal)
{
  case COLLISION_DETECTED:
    pushAction(ROTATE,PI/4);
    pushAction(GO_BACK,0.2);
    break;
  case CORRECT_END:
    break;
}
```

Tuto variantu používáme p i pro kalibraci neuronového simulátoru. Na zásobník vložíme strategii pro pevnou dráhu, kterou má robot projet. Po vykonání v ech akcí robot skon í. Rozhodovací funkce m že být samoz ejm i pro robota s pevnou strategií výrazn složit j í. Na zásobník lze vkládat krom vý e popsaných základních akcí i stavy a dal í objekty, na základ nichž se strategie m že rozhodovat o dal ím pr b hu. Framework strategie byl navržen pro obecné e ení robotických strategií. V kone né fázi jsme pro ná výzkum použili pouze základní funkce tohoto frameworku. Ostatním se zde proto nebudeme v novat.

Dal ím p ípadem, vyžadujícím jiné e ení rozhodovací funkce, je autonomní robot s adaptabilní strategií, ízený neuronovou sítí:

```
switch(retVal)
{
  case COLLISION_DETECTED:
      /*kolize se  e í pevným zp sobem -> vložíme akci „couvni o 20 cm“ na
zásobník*/
      pushAction(GO_BACK,0.2);
      break;
  case SHOCK_DETECTED:
  case CORRECT_END:
      /*V p ípad  zdárného konce akce nebo p eru ení  okem p edáme  ízení
neuronové síti, která vygeneruje na základ  stavu senzor  dal í akci*/
      m_neuralNetwork->SetInputs(in_robot);
      m_neuralNetwork->Compute();
      m_neuralNetwork->GetOutputs(cmd,param);
      /* Neuronová sí  generuje p íkaz (jaká bude následující akce) a jeho
parametr*/
      switch(cmd)
      {
         case 1:
              pushAction(GO_FWD,param);
              break;
         case 2:
              pushAction(ROTATE,param);
              break;
         case 3:
              pushAction(GO_BACK,param);
              break;
         default:
              pushAction(WAIT,param);
              break;
     }
  default:
      break;
}
```

Tato varianta v podstat zásobník nepoužívá (v zásobníku je vždy jen jedna akce). Zásobník strategie je na za átku prázdný. V každém rozhodovacím kroku neuronová sí vygeneruje dal í akci, která se okamžit za ne provád t. Výjimkou je e ení kolizí, které vykonává strategie pevným zp sobem.

5.2 Testovací prost edí

V p edchozí ásti této kapitoly jsme popsali základní principy ízení a hardware RoboKrysy. Pro provedení kompletního reálného testu jsou ov em nutné je t dal í prvky testovacího prost edí, o kterých se zmíníme v této ásti.

5.2.1 Kamera

K analýze pr b hu testu a vlastn i k jeho realizaci je v na em p ípad bezpodmíne n nutná kamera, která celou scénu sleduje. Testy by lo teoreticky provád t i bez centrální kamery — jak RoboKrysa, tak predátor jsou roboti, kte í by m li být schopni ze svých enkoder znát svou pozici v aréne. Kdyby tuto pozici pravideln hlásili centrálnímu systému ízení, ten by pak mohl zaznamenávat pr b h pokusu i rozhodovat o ocích atd. Enkodery jsou v ak jen relativní senzory a (jak jsme nastínili v kapitole 3.3 o mobilní robotice) velice rychle u nich dochází ke zkreslení domn lé pozice. Roboti by tedy vyžadovali je t absolutní senzory, které by je ale hardwarov zkomplikovali.

My jsme se rozhodli jít opa nou cestou — roboty mít co nejjednodu í a složitou lokalizaci e it pevným centrálním systémem, který bude informace o pozici zprost edkovávat robotovi. Robot tím samoz ejm ztrácí na své autonomii, ale systém jeho ízení je navržen tak, že byl-li by k dispozici p esný senzor na ur ování polohy, dal by se do systému okamžit p ipojit a robot by tím byl zcela autonomní (pomineme-li jeho ízení z centrálního po íta e, které má také d vod v maximální jednoduchosti hardware robota).

K centrálnímu po íta i ídícímu pr b h testu máme tedy p ipojenou pevn umíst nou kameru, která snímá celou arénku. Pomocí kamery a analýzy dat z ní jsme schopni ur it absolutní pozici obou pohybujících se jedinc v aréne. Tyto pozice zaznamenáváme pro pozd j í analyzování pr b hu testu a také pomocí nich zprost edkováváme RoboKryse chyb jící senzory. Jak jsme psali v úvodu této práce, necht li jsme používat pro vjemy RoboKrysy žádné její senzory, protože vytvo ením a p ipojením senzor , které by mohly alespo p ibližn zastoupit vjemy skute ného potkana, by se celá úloha neúm rn zkomplikovala. Pomocí kamery tedy RoboKryse zaji ujeme následující vjemy:

- absolutní pozice RoboKrysy - RoboKrysa ale tuto a následující informaci nepoužívá p ímo k rozhodování o dal ím pr b hu strategie. Na základ t chto hodnot pouze p ipravuje vstupy neuronové síti strategie, které více odpovídají reálnému vnímání situace potkanem. Tyto vstupy jsou popsány v kapitole 4.4 o genetickém vývoji strategie.
- absolutní pozice predátora
- kolize s predátorem
- kolize s arénkou

CMUcam2

Do na eho prost edí jsme se rozhodli použít kameru CMUcam2 kv li jednoduchosti jejího použití pro úlohy podobné té na í. Je to jednoduchá kamera propojená s jedno ipovým po íta em, který má naprogramovány základní algoritmy pro zpracování obrazu. Jedno ip zpracovává obraz a vyhodnocená data pak posílá dál po sériová lince RS232 do po íta e, který má data z kamery dál používat.

Jednou z metod zpracování obrazu, pro kterou je CMUcam ur ena, je vyhledání a sledování oblasti a ur itou barvou. Jednodu e e eno, p i této motod se kamera nakonfiguruje na konkrétní barvu, kterou má sledovat, a jedno ip pak posílá do po íta e packety se sou adnicemi a rozm ry takto barevné oblasti v obraze. Tuto metodu používáme p i sledování RoboKrysy i predátora v aránce. RoboKrysa je ozna ena výrazným erveným a predátor výrazným zeleným sv tlem. Kamera sice neumí trasovat dv barvy sou asn , zm níme-li ale parametry hledané barvy po každém cyklu, m žeme s polovi ní frekvencí sledovat ob barvy.

CMUcam2 není rozhodn ideální e ení pro v echny robotické aplikace vyžadující zpracování obrazu, ale pro konkrétní úlohy, jejichž hlavním cílem není prakticky se zdokonalit v metodách analýzy obrazu, dokáže velice zjednodu it celý systém. Proto jsme ji zvolili i my. Více o tomto modulu lze najít v [18] .

Kamera je do systému ízení robota p ipojena jako standardní senzor, který vrací absolutní pozici robota na h i ti, nebo informaci, že pozici nelze zjistit. Z kamery bohužel nelze p ímo zjistit orientaci robota na h i ti, proto není informace z ní brána p ímo jako pozice robota. Informace z tohoto senzoru slouží lokalizaci na principu MCL, který je popsán vý e v kapitole 3.3 o mobilní robotice, jako absolutní informace o pozici. Spole n s relativními vstupy z enkoder se pak po ítá pozice i orientace RoboKrysy v aránce.

ArenaCalibrator

Aby byla data z kamery použitelná pro p esné ur ování pozice, je nutné kameru po každém sestavením testovacího prost edí softwarov zkalibrovat, nebo hardwarovými prost edky není možné zajistit identické umíst ní kamery v každém testu. K tomu slouží ná program ArenaCalibrator. Kalibrace kamery probíhá následovn :

1. Umístíme kameru a arénku do p ibližn správné polohy.
2. ArenaCalibrator získá snímek arénky z kamery a zobrazí ho na pracovní plo e.
3. My í ozna íme t i body v obraze, které leží na hranici arénky. Program t mito t emi body proloží kružnici, kterou zobrazí.
4. Neshoduje-li se kružnice s okrajem arénky, upravíme pozici arénky, i kamery (vodorovnost) a postup opakujeme. Je-li pozice identická, program uloží konfiguraci kružnice (posunutí a velikost arénky v i kame e) a ídící program robota pak p evádí sou adnice kamery na sou adnice pozice robota správn .

5.2.2 Komunikace

Dal ím prvkem testovacího systému je komunika ní systém mezi RoboKrysou a ídícím po íta em. V kapitole 5.1 o implementaci RoboKrysy bylo popsáno, že na samotném robotovi je jen jednoduchý jedno ipový po íta , ve kterém neprobíhají žádné složité výpo ty a ve keré operace týkající se ízení strategie robota se vykonávají v externím PC, které zárove ídí celý test, tedy zpracovává data z kamery atd. Pro propojení jedno ipu s PC existuje n kolik r zných metod. Od jednoduché sériové linky UART i I2C po bezdrátová e ení jako WiFi, IrDA nebo Bluetooth. Na tuto komunikaci jsme m li tyto požadavky:

- musí být bezdrátová, aby se p ede lo hardwarovým problém m p i reálných testech a zbyte né kabely nep ekážely kame e
- musí být jednodu e a levn realizovatelná jak na jedno ipu, tak i na PC

První metodou, kterou jsme zvolili, byla klasická sériová komunikace UART/RS232. Je to velice jednoduchá a rychlá metoda, která nevyžaduje ani na jedné ze stran žádné p ídavné za ízení ani složitá softwarová e ení. Nevýhoda této metody je nutnost propojení kabelem. Používali jsme ji ale hlavn v za átku pro odlad ní základních ástí komunikace a složit laditelného software v jedno ipu robota.

Je t relativn jednoduchou bezdrátovou náhradou sériové linky je IrDA, tedy infra ervená komunikace. Její velkou nevýhodou je ale sm rovost — komunikující za ízení na sebe musí vid t, na což jsme se necht li omezovat. Dal í nevýhodou jak UART tak IrDA komunikace je její dvoubodovost, tedy komunikovat mohou po jedné lince vždy pouze 2 za ízení. To by v na em p ípad sta ilo, snažili jsme se ov em vytvo it systém, který by byl použitelný i pro na i dal í práci nap íklad na evoluci skupin robot , kde už by bylo pot eba mít komunikujících za ízení více.

Jako bezdrátové e ení vícebodové komunikace p icházel v úvahu standard WiFi, který jsme ov em zavrhli kv li jeho vysoké spot eb energie na stran jedno ipu. Nakonec jsme zvolili technologii ze sv ta mobilních telefon — Bluetooth, který je p ímo dimenzován na nízkospot ebová jednoduchá za ízení.

Bluetooth je velice propracovaný standard, který popisuje a implementuje v echny vrstvy komunika ního modelu. Na nejvy í vrstv nabízí profily pro snadnou komunikaci konkrétních typ za ízení — telefon , headset , bezdrátových modem atd. Jedním z profil je i sériový profil, který by m l transparentn nahrazovat sériový port. Pomocí dvou takto nastavených a spárovaných za ízení lze pak bezdrátov nahradit sériový kabel.

Ve snaze o vytvo ení univerzálního komunika ního systému pro více robot jsme se ale rozhodli tohoto jednoduchého, ale op t pouze dvoubodového, e ení nevyužít. Bluetooth model nabízí o n kolik vrstev níže hladinu s HCI protokolem, která se zdála být pro na e plány mnohem použiteln j í. HCI nabízí soubor p íkaz a událostí, pomocí nichž lze zrealizovat spojení n kolika za ízení a v tomto systému pak adresn zasílat datové zprávy. Rozhodli jsme se tedy jít tímto sm rem. Výhody HCI protokolu jsou z ejmé — nabízí rychlé mnohobodové spojení, nad kterým máme plnou kontrolu. Nevýhodou je obtížn j í implementace, protože se jedná o pom rn low-level e ení. Po p ekonání v ech problém s implementací se ale tato metoda p i reálných testech osv d uje.

Výsledkem na eho snažení je tedy soubor funkcí realizující HCI interface jak na PC tak na jeno ipu robota. Pomocí Bluetooth dokážeme bezdrátov komunikovat s robotem na vzdálenost až n kolika desítek metr bez p ímé viditelnosti, což velmi podstatn usnad uje práci s robotem a eliminuje hardwarové problémy vzniklé opot ebováním kabel . Systém navíc umož uje pozd j í p ipojení dal ích komunikujících za ízení. Kompletní dokumentace k protokol m technologie Bluetooth se nachází v [17] .

5.2.3 Predátor

Je t jsme se nezmínili o realizaci predátora v na ich testech. Predátor je do testovacího prost edí zapojen v podstat pouze svou p ítomností. Tedy systém monitoruje pozici predátora pomocí kamery (predátor je ozna en barevným sv tlem), ale nijak ho ne ídí. V po áte ní fázi pokus jsme používali místo pohyblivého predátora pouze ozna enou pasivní p ekážku, kterou jsme manipulovali posouváním po aréne. Pozd ji jsme využívali na eho star ího robota RobuTecII [20] , kterého používá k pokus m na potkanech i Fyziologický ústav AV R, což usnad uje porovnatelnost na ich výsledk s jejich na skute ných potkanech. O tomto robotovi se zde nebudeme více rozepisovat, protože nebyl vytvo en p ímo pro tuto diplomovou práci a lze jej nahradit jinými prost edky.

5.3 GeneticLab

Tato a následující podkapitola se zmíní o dvou programových knihovnách, které jsme vytvo ili jako pom cku p i vývoji systém využívajících metod evolu ních algoritm a neuronových sítí.

GeneticLab je soubor obecných t íd, který umož uje snadnou a rychlou implementaci genetického u ení pro libovolný úkol. Tyto t ídy samy zaji ují ízení u ení, pot ebné ovládací prvky pro jeho kontrolu a zaznamenávání jeho pr b hu. GeneticLab implementuje následující algoritmus genetického u ení:

1. Vytvo í po áte ní populaci.
2. Ohodnotí fitness pro každého jedince v populaci.
3. Vytvo í novou generaci tak, že náhodn vybere vždy dva jedince z p vodní populace a jejich zk ížením vznikne jedinec nový. Toto se opakuje, dokud není nová populace plná. P i výb ru k reprodukci mají v t í anci jedinci s v t í hodnotou fitness. GeneticLab p i výb ru používá metodu rulety, na které každému jedinci p ipadá díl úm rný jeho umíst ní mezi ostatními podle velikosti fitness.
4. Na n které jedince aplikuje mutaci. Jedinci se vybírají náhodn . Pravd podobnost výb ru k mutaci je volitelný parametr.
5. Pokra uje bodem 2.

K vytvo ení nové úlohy pro genetické u ení v GeneticLab je t eba vytvo it t ídu implementující jednoho jedince v nové populaci pro tento úkol. Tato t ída musí být potomkem t ídy pro obecného jedince `CAbstractSample` a implementovat následující metody:

- `void Create()` - Nastaví vlastnosti jedince do první populace.
- `CAbstractSample * Crossover(CAbstractSample * sample)` - Vrátí nového jedince, který vznikl zk ížením jedince p vodního a jedince zadaného v parametru.
- `void Mutate()` - Provede mutaci na jedinci.
- `void Evaluate()` - Spo te hodnotu fitness jedince.
- `void LogSample(char * text)` - Vytvo í et zec charakterizující jedince. Tento et zec se používá k zaznamenávání nejlep ích jedinc jednotlivých generací.
- `void PrintValue(char * text)` - Vytvo í et zec charakterizující jedince. Tento et zec se používá k zobrazení charakteristiky jedince b hem jeho evaluace.

Systém GeneticLab jsme použili k implementaci genetického u ení vah neuronové sít , která ídí strategii RoboKrysy. Tato úloha pro genetické u ení se od ostatních li í v jednom bod . Ohodnocení jednoho vzorku totiž vychází pokaždé jinak. D vodem je používání náhodného umu na senzorech robota a vždy jiné dráhy simulovaného predátora. Snažíme se tím p edcházet adaptaci strategie na konkrétní chování jednoho predátora, což by vedlo k neefektivit s jiným chováním predátora. Šum používáme k zrobustn ní nau ené strategie. Kolísání ohodnocení jednoho vzorku se snažíme kompenzovat tím, že ohodnocení jedince po ítáme vždy jako pr m r ohodnocení t í jízd.

P esto se nelze kolísání vah zcela vyhnout. GeneticLab proto ohodnocuje každého jedince populace i v okamžiku, kdy je identický s již ohodnoceným jedincem z p edchozí generace (nedo lo k mutaci a jedinec se dostal do dal í generace). Bez tohoto opat ení docházelo k stavu, kdy jeden jedinec získal vysoké ohodnocení díky shod náhod v konfiguraci testu a tento jedinec m l pak nejv t í anci na reprodukci. Nedo lo-li k mutaci, z stalo novému jedinci p vodní vysoké ohodnocení a í il se do dal ích generací, a koliv by v dal ích ohodnoceních tak kvalitní nebyl. Toto chování lze ov em stejn jako dal í vlastnosti systému zm nit parametrem p i jeho kompilaci.

5.4 NeuralNetwork

Skupina t íd NeuralNetwork je dal í pom ckou p i vývoji adaptabilního systému. Tyto t ídy implementují obecnou vrstevnatou neuronovou sí . Systém umož uje snadné vytvo ení neuronové sít s t mito vlastnostmi:

- Sí má pouze dop edné spoje.
- Jedná se o vrstevnatou sí , tj. když *i* je index vrstvy sít , spoje mohou vést jen z vrstvy *i* do vrstvy *i+1*.
- Sí má vždy propojeny v echny neurony mezi sousedními vrstvami.
- V každé vrstv je jeden neuron, který nemá žádné vstupy a jeho výstup je vždy -1. Spojení tohoto neuronu s neurony v následující vrstv nahrazuje p ítomnost prah v neuronech.
- Pro adaptaci vah sít lze použít algoritmus zp tného í ení chyby, nebo p i propojení se systémem GeneticLab genetické u ení.
- Sí používá sigmoidální p enosovou funkci.

Pro vytvo ení vlastní neuronové sít je pot eba vytvo it potomka obecné t ídy pro neuronovou sí `CAbstractNNetwork`. Chceme-li, aby se nová neuronová sí u ila metodou zp tného í ení, je nutné v ní implementovat tyto metody:

- `void InitPatterns()` - Inicializuje trénovací vzory neuronové sít .
- `void InitConstants()` - Nastavení konstant, které se používají v algoritmu u ení.

Nastavitelné konstanty ovliv ující pr b h u ení jsou následující:

- po et iterací - Kolikrát bude p edložen síti jeden trénovací vzor v jednom cyklu u ení.
- po et cykl - Kolikrát se bude opakovat celé u ení.
- alfa - moment u ení; Je z intervalu 0 až 1. Ur uje jaký vliv na výslednou váhu bude mít úprava váhy z p edchozího kroku, tedy jde o jakousi setrva nost v u ení.
- eta - Ur uje jaký vliv na zm nu váhy bude mít aktuáln spo tená chyba u ení.

Systém NeuralNetwork jsme použili jako základ neuronového simulátoru, simulátoru okova e a neuronové strategie. V prvních dvou p ípadech se neuronová sí adaptovala algoritmem zp tného í ení. V p ípad neuronové strategie váhy sít vyvíjel genetický algoritmus.

6 Testování

V této kapitole popí eme výsledky, kterých dosáhl ná adaptabilní robotický systém v reálných testech. Nejprve se zam íme na základní úlohu, která byla spole ná pro RoboKrysu i pro potkana. Na ní provedeme porovnání výkonnosti, charakteristiky chování a u ebního procesu obou jedinc . V dal í ásti uvedeme výsledky adaptace RoboKrysy na jiné úlohy a provedeme diskusi jejích možností.

6.1 Porovnání potkana a robota na stejné úloze

Jako platformu, na které budeme porovnávat proces u ení robota se skute ným potkanem, jsme si v za átku této práce ur ili úlohu „predátor a ko ist“. Ko ist nesmí dovolit, aby vzdálenost mezi ním a predátorem klesla pod 25 cm, jinak dostane elektrický ok. Jak potkan tak i RoboKrysa p edstavují v této úloze ko ist, které jde o minimalizaci po tu ok , jenž b hem testu dostane.

Výsledk , prezentovaných v této kapitole, bylo dosaženo p i testovacích jízdách se skute ným robotem. P enos nau ené strategie ze simulátoru na skute ného robota nemá výrazný vliv na zm nu kvality e ení. Základní charakteristika chování z stává i po p enosu stejná. Skute ný robot se pohybuje plynuleji než simulovaný. Chování skute ného robota je místy ovlivn no skokovou zm nou pozice (nap íklad p i proto ení kole ek po nárazu do mantinelu), což je d sledek použití metody MCL k lokalizaci robota v aréne. V n kterých p ípadech do lo i ke „ztrát “ robota v aréne — pozice, na které si robot myslel že je, byla výrazn jinde než jeho skute ná pozice. Tento problém je ale e itelný kvalitn j ím hardware robota.

Skute né testy s laboratorními potkany používaly dva druhy predátora. Predátorem byl b hem celého procesu u ení bu dal í potkan, nebo robot. Chování potkana-predátora nebylo nijak ovliv ováno. Robot-predátor byl identický s robotem, kterého jsme používali jako predátora my.

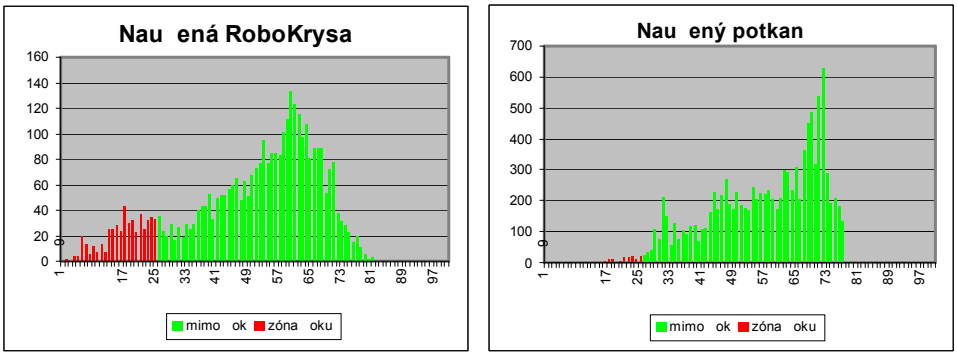
Je pot eba poznamenat, že u ení potkana s r zným druhem predátora má r zný pr b h. Potkan má p irozenou tendenci se pohybujícímu se um lému t lesu (robotovi), na rozdíl od dal ího potkana, vyhýbat. Proto v tomto p ípad dosahuje velice dobrých výsledk už v prvních trénovacích jízdách. RoboKrysa tuto p edem zabudovanou vlastnost nemá. Dal í odliností je chování predátora. Robot-predátor se pohybuje konstantní rychlostí náhodn po celé plo e arénky. Predátor-potkan se v t inu asu nepohybuje tak rychle a nepokrývá takovou plochu (asto se zdržuje u mantinel , kde eká).

Nejsme tedy schopni zajistit testování za úpln identických podmínek. P esto je možné ve v t in aspekt zkoumaného chování obou jedinc najít paralely, které popí eme v dal ím textu.

6.1.1 Porovnání nau ených chování

Charakteristika chování nau eného potkana	Charakteristika chování nau ené RoboKrysy
Zdržuje se výhradn p i okraji arénky.	Robot se zdržuje p i okraji arénky.
V t inu asu stojí na míst (náhodn padající potrava pro n j není dostate nou motivací).	Neustále jezdí na malém prostoru sm rem k mantinelu a od n j. Tím pokrývá v t í plochu a „sní“ více zrní.
Blíží-li se predátor jeho sm rem, po okraji se p esouvá na opa ný konec arénky.	Na blížícího se predátora nereaguje.
V okamžiku oku má i nenau ený potkan tendenci rychle zm nit pozici, což ho z oblasti oku v t inou vyvede.	V okamžiku oku v t inou zm ní svou orientaci sm rem od predátora a jede vp ed. Toto chování ale není zcela pr kazné.

Následující ty i diagramy znázor ují etnost výskytu ko isti v ur ité vzdálenosti od predátora. Vodorovná osa reprezentuje vzdálenost v cm, svislá osa dobu výskytu v dané vzdálenosti v desetinách sekundy b hem testu.



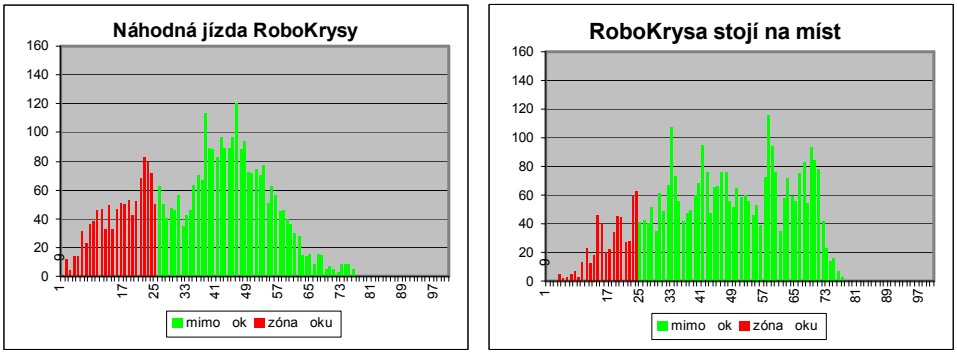
Obr. 13 Histogramy výskyt nau ené RoboKrysy a potkana v r zných vzdálenostech od predátora.Test RoboKrysy trval 6 minut, test potkana 20 minut.

P edchozí histogramy ukazují rozdíl ve kvalit nau eného chování mezi RoboKrysou a potkanem. Potkan je schopen efektivn ji udržovat bezpe nou vzdálenost od predátora. Nau ený potkan se nachází v oblasti oku p ibližn ve 2% celkové doby testu, zatímco RoboKrysa pr m rn ve 13%. Pr m rná vzdálenost od predátora je ale v p ípad RoboKrysy i potkana stejná (49 cm).

U RoboKrysy se nevyvinulo chování, které by jí umožnilo p edcházet ok m v p ípad blížícího se predátora. Robot dokáže z oblasti oku efektivn vyjet a zdržovat se v ásti arénky, kde se predátor nevyskytuje p íli asto. Bez efektivního

p edcházení ok m ale nelze dosáhnout výsledk srovnatelných s potkanem. D vod m tohoto problému se budeme v novat pozd ji v tomto textu.

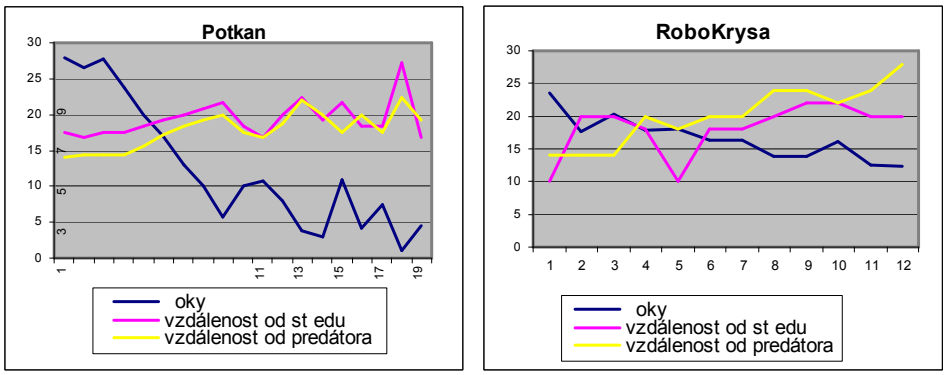
Pro úplnost zde uvádíme následující dva histogramy, které znázor ují rozložení vzdáleností RoboKrysy od predátora v p ípad její náhodné jízdy a v p ípad , že RoboKrysa stojí na nejvhodn j ím míst arénky (40 cm od st edu). Tato pozice se nachází v zón oku pr m rn v 16% celkové doby testu. Pr m rná vzdálenost od predátora je 45 cm. Nau ená RoboKrysa dokáže tedy svým pohybem u okraje arénky nejen zvý it množství sn dené potravy, ale i aktivn snížit riziko oku. RoboKrysa, která se pohybuje náhodn , se nachází v zón oku p ibližn v 33% p ípad .



Obr. 14 Histogramy vzdáleností RoboKrysy od predátora. V prvním p ípad se pohybuje RoboKrysa náhodn . Ve druhém p ípad stojí na míst vzdáleném 40 cm od st edu arénky (u mantinelu).

6.1.2 Porovnání procesu u ení

V této ásti porovnáme cestu, kterou zkoumaní jedinci pro li b hem adaptace na základní úlohu. Jako ilustrace k srovnávací tabulce mohou sloužit následující dva diagramy, které zobrazují souvislost mezi rostoucí kvalitou e ení a zvy ující se pr m rnou vzdáleností od st edu arénky a od predátora.



Obr. 15 Diagramy procesu u ení potkana a RoboKrysy. Svislá osa reprezentuje procento doby strávené v zón oku. Vodorovná osa reprezentuje v p ípad potkana po adí testu a v p ípad RoboKrysy po adí generace genetického algoritmu, jejíž nejlep í jedinec m l zobrazené vlastnosti (n které nereprezentativní generace jsou vynechány). K ivky znázor ující ob vzdálenosti jsou vloženy do graf pro ilustraci jejich souvislosti s procentem ok . Jsou jednotliv kálovány a posunuty, aby jejich trendy byly co nejvýrazn j í. Jejich konkrétní hodnoty proto nedávají v tomto zobrazení smysl.

	Proces u ení potkana	**Proces u ení RoboKrysy**
Po et skute ných trénovacích jízd	20	1 (Nepo ítáme projetí kalibra ní dráhy simulátoru, protože odpovídá tomu, že potkan také p edem ví, jakým zp sobem se pohybuje a dokáže koordinovat svoje chování)
Celková doba trénovacích jízd	20 x 20 minut = 6 hodin 40 minut	6 minut
Chybovost nau eného jedince	2 %	13%
Rychlost u ení	Pot ebuje ke snížení své chybovosti na 13% p ibližn 7 trénovacích jízd, tedy 2 hodiny 20 minut.	Úrovn 13% dosáhne po jedné 6 minutové trénovací jízd . Dál už se ale nezlep uje. K tomuto asu je ale je t nutno uvažovat s dobou genetického u ení v simulátoru, které trvá p ibližn 3 hodiny. Reálné testy s potkany ov em také neprobíhají po sob , ale potkan má minimáln 1 den as o každém testu „p emý let“.
Vývoj chování b hem u ení (viz p edchozí 2 grafy)	Potkan se v prvních trénovacích jízdách pohybuje v podstat náhodn . Má p irozen inteligentní chování v p ípad oku — út k. Postupn dochází k optimalizaci chování v i ok m tím, že se zdržuje více p i okraji, není tak aktivní a p edchází ok m v asným út kem. K vývoji t chto vlastností chování dochází spole n .	Genetický algoritmus odhalil už v prvních generacích závislost po tu ok na vzdálenosti od st edu arénky. Proto nejlep í jedinci t chto generací v t inou ekali u okraje arénky. V pozd j ích generacích dochází postupn ke zvy ování pr m rné vzdálenosti od predátora, která také p ímo souvisí s po tem udílených ok . Toho RoboKrysa dosahuje áste n inteligentním chováním v p ípad , že je v zón oku.

6.2 Výsledky testování na jiných úlohách

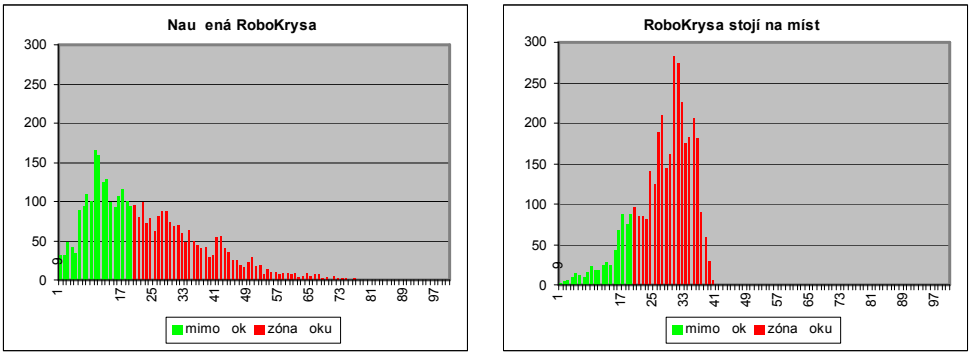
V této podkapitole uvedeme výsledky n kolika modifikovaných pokus , abychom ukázali univerzálnost adaptabilního systému RoboKrysy.

6.2.1 Lovení predátora

Následující úlohu jsme nazvali „lovení predátora“. Testovací prost edí i chování predátora je totožné s na í základní úlohou „predátor a ko ist“, rozdíl je pouze v chování okova e. RoboKrysa tentokrát dostává oky, zvý í-li se její vzdálenost od predátora nad 20 cm.

Pro tuto úlohu nemáme ekvivalent v pokusech s potkany, provedeme tedy pouze porovnání se statických chováním RoboKrysy. Nejvýhodn j í bod pro setrvání RoboKrysy b hem celého pokusu je st ed arénky. Predátor se totiž nejpravd podobn ji vyskytuje práv v jeho okolí.

	Nau ená RoboKrysa	**Statická RoboKrysa**
Chybovost nau eného jedince	48%	82% (chybovost náhodn jezdícího robota je pr m rn 87%)
Pr m rná vzdálenost od predátora	22 cm	27 cm (43 cm u náhodného jedince)
Charakteristika chování	Robot aktivn využívá schopnosti oto it se sm rem k predátorovi. P evážn se to í a pohybuje sm rem k n mu.	RoboKrysa stojí v bod s nejv t í pravd podobností výskytu predátora.



Obr. 16 Histogramy vzdáleností RoboKrysy od predátora. V prvním p ípad se RoboKrysa pohybuje podle nau ené strategie, ve druhém stojí na míst uprost ed arénky.

Na této úloze se uplatnila schopnost RoboKrysy zatá et relativn k predátorovi mnohem ve v t í mí e než v na í základní úloze. Zatímco nau ené e ení základní úlohy má spí e charakter „zaujmi co nejbezpe n j í pozici v aréne a na ní setrvej“, p i e ení této úlohy je robot výrazn aktivn j í.

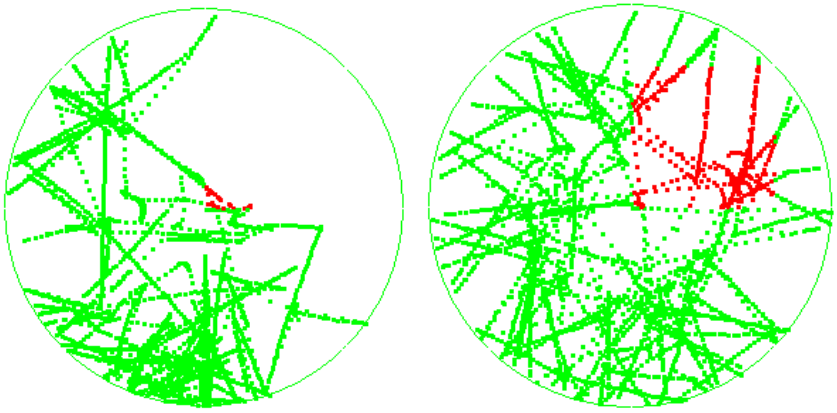
6.2.2 Vyhýbání se prostoru

Na úloze „vyhýbání se prostoru“ ukážeme schopnost adaptace RoboKrysy na úkol, který p ímo nesouvisí s predátorem. Úkolem je nyní nevstupovat do tverce 30 x 30 cm umíst ného napravo a nahoru od st edu arénky. RoboKrysa startuje uprost ed arénky a je orientována napravo. V aréne se pohybuje i predátor, který ale na oky nemá vliv.

Nau ená RoboKrysa je schopna splnit tento úkol s chybou p ibližn 4%, zatímco RoboKrysa jedoucí náhodn dostane ok v 11% p ípad .

Zajímavé na této úloze je to, že strategie robota se nemá možnost rozhodovat podle absolutní pozice v aréne. V echny její vstupy o pozici jsou pouze relativní ke st edu arénky, nebo k predátorovi. Šokova absolutní informaci o pozici naopak má (a koliv není jisté, odpovídá-li to realit potkana), takže je schopen úlohu zachytit.

Vyvinutá strategie tedy m že po ítat jen s tím, že robot startuje vždy na stejném míst se stejnou orientací. Dal í absolutní informaci o pozici nebezpe né oblasti dostane RoboKrysa až v okamžiku oku. Její chování spo ívá v jízd od mantinelu tém ke st edu arénky a zp t. Toto se snaží d lat s malým rozptylem. V okamžiku oku v t inou zm ní sm r na opa ný.



Obr. 17 Dráhy nau ené a náhodn jezdící RoboKrysy na úkol „vyhýbání se prostoru“. Zelená ást dráhy je bez oku, ervená s okem.

6.2.3 Zm na podmínek pro nau eného robota

Nyní se vrátíme k základní úloze „predátor a ko ist“ a ukážeme, jaká je výkonnost nau eného robota p i testu se zm n nými podmínkami od t ch, jaké m l p i svém u ení. Budeme se zabývat dv ma p ípady:

1. RoboKrysa je nau ená na statického predátora[††] a p emístíme ji do arénky s pohybujícím se predátorem. Dále ji budeme nazývat statický robot.
2. RoboKrysa je nau ená na pohybujícího se predátora a p emístíme ji do arénky se statickým predátorem. Dále ji budeme nazývat dynamický robot.

	Statický robot	**Dynamický robot**
Chyba nau eného robota v p vodní úloze	5%	13%
Chyba náhodné jízdy na p vodní úloze	15%	33%
Chyba po p enesení do nové úlohy	21%	**30%**

Obr. 18 Tabulka chybovosti RoboKrysy. Stejný odstín ozna uje testy provedené za stejných podmínek.

Statický robot dosahuje po p enesení do objektivn složit j í úlohy pom rn dobrého výsledku. Zajímavý je jeho zp sob e ení nového úkolu, protože se podstatn li í od zp sobu, jakým tuto úlohu e í dynamický robot. e ení statického robota je založené na rychlém opu t ní zóny oku, když se do ní dostane. Jinak se pohybuje po celé plo e arénky, v etn st edové oblasti.

Aktivní opu t ní oblasti oku je pro statického robota p irozené chování, protože když se do této oblasti dostane ve svém p vodním prost edí, predátor se nehýbe. Proto musí vy e it situaci sám. Chybí mu ale p irozené chování dynamického robota, který se nezdržuje v nebezpe né oblasti st edu arénky, kde je výskyt predátora nej ast j í.

Zajímav j í než dobrý výsledek statického robota je ale velice patný výsledek dynamického robota po p enesení na objektivn jednodu í úlohu se statickým predátorem. Dynamický robot na této úloze dosahuje výrazn hor ího výsledku, než kdyby se pohyboval náhodn .

D vodem jsou jeho návyky z p vodního prost edí. P i e ení úkoly se p esune k okraji arénky, kde se pohybuje od mantinelu a zp t. asto p itom m ní polohu v arénce, ale vždy jen p i okraji. Dostane-li se ve svém p irozeném prost edí do oblasti oku, nemusí to prakticky e it, protože pom rn rychle se pohybující predátor brzy p emístí tuto oblast jinam. To se ov em ned je v arénce se statickým predátorem. Narazí-li proto dynamický robot p i svých cestách p i okraji arénky na predátora, trvá velice dlouho, než se ze zóny oku dostane.

[††] Predátor stojí b hem celého pokusu na jedné, náhodn vybrané, pozici. Na této úloze stojí za pov imnutí to, že okova se na ní adaptoval z pouze jediné trénovací jízdy (tedy predátor byl pouze na jedné náhodné pozici!). Šokova odhadl správn souvislost mezi oky a p ítomností predátora. P i testování takto nau eného okova e s pohyblivým predátorem byla jeho chybovost pouze 13%, oproti asi 1,5% u okova e u eného na pohyblivém predátorovi.

Z výše popsaného vyplývá, že ideální chování by vzniklo kombinací dobrých vlastností obou naučených robotů. Vývoj takto komplexního chování je ovšem kvůli univerzálnosti naučeného adaptabilním systému těžko dosažitelný.

6.2.4 Hranice schopností RoboKrysy

Na předcházejících testech jsme ukázali, že je RoboKrysa schopná adaptace i na jiné úlohy, než je základní „predátor - kořist“. Kvalita její adaptability má ale své meze. První okruh omezení úkolů, které je RoboKrysa schopná se naučit, plyne ze struktury adaptabilního čidlového pole. Nepostihne-li čidlová podstatu úlohy z údajů tréninkové jízdy, adaptabilní strategie ztrácí možnost se úlohu naučit. Šidlová nesta í na následující problémy:

- Je-li podmínka pro udělení odměny příliš speciální a RoboKrysa dostane tedy během své jediné tréninkové jízdy odměn málo, neuronový simulátor čidlového pole má tendenci naučit se nedávat odměny vůbec. To samé platí i opačně. Nachází-li se RoboKrysa v oblasti odměny ve velké vzdálenosti od doby tréninkové jízdy, naučí se čidlová dávat odměny neustále. Chceme-li například naučit RoboKrysu úlohu „lovení predátora“ a změníme maximální vzdálenost od predátora, ve které nedochází k odměně, na 10 cm, dostane RoboKrysa během náhodné jízdy odměn v přibližně 92% případů. Z takové tréninkové jízdy se čidlová naučí udělovat odměn neustále a RoboKrysa tedy nemá šanci se na úlohu adaptovat.
- Šidlová se nedokáže naučit složitější oblasti, které určují oblast odměny. Je-li například takovou oblastí čtverec, čidlová má problém s přesným postihnutím jeho ostrých rohů. Spíše ho aproximuje kružnicí, která přibližně odpovídá jeho velikosti. Tento případ ovšem nebrání RoboKrysе v úspěšné adaptaci na úlohu, protože se rozhodující části nebezpečného čtverce naučí vyhýbat i s jeho nedokonalým modelem v čidlové. Je-li bezpečnou oblastí pro RoboKrysu například mezikruží, vytvoří se jeho model v čidlové natolik nepřesně, že už RoboKrysa není schopna se na úkol viditelně adaptovat.

To, že čidlová zvládne postihnout podstatu úlohy s dostatečnou přesností, je dobrým předpokladem ke kvalitní adaptaci strategie RoboKrysy. Naučené řešení ale často nedokáže efektivně řešit základní problém úlohy. Robot se spíše naučí vykonat několik jednoduchých kroků k tomu, aby snížil nebezpečí čidlu z původního stavu. Složitější chování, které by šlo jeho dále k podstatě udělování odměn, se ale vyvine jen v některých případech.

Adaptace na složitější problémy se často v evoluční robotice realizuje inkrementální evolucí jednotlivých dílčích dovedností, nutných k dosažení celkového chování [7]. To ovšem vyžaduje mít speciální fitness funkce pro tyto dovednosti a vhodný algoritmus řídit tak, aby konečné chování bylo kvalitní. Tento přístup v našem případě nebyl možný, protože jsme si dali za úkol vytvořit adaptabilního robota, který by byl univerzální nad jistou třídou úloh, které definovalo pouze chování čidlového pole. Proto je naše fitness zaměřená pouze na počet obdržených odměn (a počet snědeného zrní, které motivuje robota k aktivitě). To sice snižuje kvalitu naučených řešení, ale zvyšuje univerzálnost systému.

7 Záv r

Cílem práce bylo navrhnout a implementovat kompletní systém pro porovnávání procesu u ení živo ich a robot . Vybrali jsme úlohu definovatelnou pro mobilního robota i laboratorního potkana a navrhli a vytvo ili robota a jeho ídící systém, který by se zadanou úlohu dokázal nau it. K adaptaci strategie robota jsme použili vrstevnaté neuronové sít , u ené jak algoritmem zp tného í ení, tak genetickým algoritmem. Provedli jsme testy nau eného e ení a jejich výsledky dali do souvislosti s výsledky experiment s potkany. Výsledný systém jsme testovali i na modifikacích p vodní úlohy, abychom ukázali jeho adaptabilitu.

P i realizaci systému byly dosaženy n které zajímavé výsledky. Poda ilo se vyvinout robota, který je po adaptaci schopen e it zadanou úlohu a její modifikace prokazateln efektivn ji. P i simulaci jízdy robota jsme úsp n použili jako hlavní prvek neuronovou sí , což se ukázalo jako výhodná metoda pro aproximaci složitého chování hardware robota. Vyvinuli jsme obecné testovací prost edí pro evolu ní experimenty s mobilními roboty, které je použitelné i pro dal í úlohy.

P i návrhu systému bylo nutné e it adu problém . Mezi n pat ilo nalezení zp sobu porovnatelného definování úlohy pro robota a potkana, navržení realizovatelné architektury adaptabilního systému, i e ení problém s p enositelností nau ených strategií ze simulátoru na skute ného robota. P i porovnávání výsledk potkana s robotem se ukázalo, že a koliv nau ené strategie robota nemohou s potkanem v kone né fázi soupe it, um lý systém dosahuje v mnoha parametrech podobných vlastností.

Na záv r lze tedy íci, že byl cíl práce podle specifikace spln n. Vytvo ený systém je schopen se adaptovat na úlohy, které jsou srovnatelné s úlohami pro živo ichy. Systém byl implementován pro reálného robota a na konkrétní úloze byly porovnány výsledky a proces u ení robota a potkana. Použité metody se ukázaly jako vhodné e ení zadaného tématu. Je ov em t eba poznamenat, že tato práce si nekladla za cíl vy erpat téma evoluce robot ani její porovnávání s reálnými e eními ze sv ta živo ich zcela.

8 P ílohy

8.1 Obsah CD

Na disku CD p iloženém k textu práce se nachází následující d ležité soubory a adresá e:

- **readme.txt** – textový soubor se stru ným popisem obsahu CD
- **prezentace.bat** – program pro spu t ní prezentace uložené na CD
- **/demo** – soubor aplikací demonstrujících pr b h u ení robota
- **/help** – vysv tlivky k n kterým demo aplikacím
- **/images** – galerie fotografií robot a testovacího prost edí
- **/sources** – zdrojové kódy použitých aplikací
- **/text** – text diplomové práce ve formátech PDF a DOC

8.2 Instalace prezentace

Protože aplikace pro u ení RoboKrysy vyžadují speciální hardware, nachází se na p iloženém CD pouze prezentace jejich použití, která nasti uje postup u ení. Ke spu t ní této prezentace je t eba provést následující:

- P ekopírujte celý obsah CD na pevný disk (cca 25 MB), kde budou mít programy právo zapisovat. N které programy si ukládají záznamy o svém b hu do soubor .
- Pro správné zobrazení grafických výstup prezentace je nutné mít rozli ení obrazovky minimáln 1024x768 pixel .
- Prezentace ke svému b hu pot ebuje program Microsoft Internet Explorer (testováno s verzí 6.0)
- Prezentace se spustí programem „prezentace.bat“ v ko enovém adresá i (více informací v souboru „readme.txt“).

Použité aplikace jsme testovali v OS Windows XP na PC s procesorem Intel Pentium M 1400MHz s 256 MB RAM.

9 Literatura

[1] Nolfi S. & Floreano D. (2000): Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines, MIT Press/Bradford Books, Cambridge, MA.

[2] Michalewicz Z. (1994): Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag.

[3] Spronck P., Sprinkhuizen-Kuyper I.G., Postma E.O. (2001):Evolutionary Learning of a Neural Robot Controller, Universiteit Maastricht, IKAT.

[4] Šíma J., Neruda R. (1996): Teoretické otázky neuronových sítí, Matfyzpress, Praha.

[5] Nolfi S., Parisi D. (2002): Evolution of Artificial Neural Networks, In M. A.Arbib (Ed.), Handbook of brain theory and neural networks, Second Edition. Cambridge, MA: MIT Press, pp418-421.

[6] Wyeth G. (1997): Neural Mechanisms for Training Autonomous Robots, Proceedings of Mechatronics and Machine Vision in Practice Conference, IEEE Computer Society Press.

[7] Nolfi S. (1996):Using emergent modularity to develop control systems for mobile robots, Institute of Psychology, National Research Council, Rome.

[8] Olsen S., Fowles M. (2003): Giving Nolfi a Fair Fight - A Comparison of Emergent Modularity and Elman Recurrent Networks, http://www.cs.swarthmore.edu/~meeden/cs81/projects/olsen-fowles.ps

[9] Gallistel C.R. (1990): The Organisation of Learning, MIT Press, Cambridge, MA.

[10] O'Keefe J., Nadel L. (1978): Hippocampus as a cognitive map, Clarendon Press, Oxford.

[11] Bure M. (2003): Metody u ení pro prostorovou navigaci, Diplomová práce, VUT Praha.

[12] Jones L.J., Flynn A.M., Seiger B.A. (1998): Mobile Robots: Inspiration to Implementation, AK Peters, Ltd., Wellesley, MA.

[13] Everett H. R. (1995): Sensors for Mobile Robots: Theory and Application, A K Peters, Ltd., Wellesley, MA.

[14] Fox D., Burgardy W., Dellaert F., Thrun S. (1999): Monte Carlo Localization: Efficient Position Estimation for Mobile Robots, In Proceedings of the Sixteenth National Conference on Artificial Intelligence, Orlando, FL.

[15] Willis M.J. (1999): Proportional-Integral-Derivative Control, University of Newcastle.

[16] erný P., Hradecký T., Jiroutek P., Lysák J. (2004): Eurobot 2004, Softwarový projekt, MFF UK Praha.

[17] Bluetooth SIG (2001): Specification of the Bluetooth System 1.2, https://www.bluetooth.org/foundry/adopters/document/Bluetooth_Core_Specification_v1.2

[18] Carnegie Mellon University (2003): CMUcam2 Vision Sensor - User Guide, http://www-2.cs.cmu.edu/~cmucam2/CMUcam2_manual.pdf

[19] Atmel Corporation (2004): ATmega8 datasheet, http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf

[20] Pavel Jiroutek (2005): RoboKrysa, http://robotika.cz/articles/roborat/cs