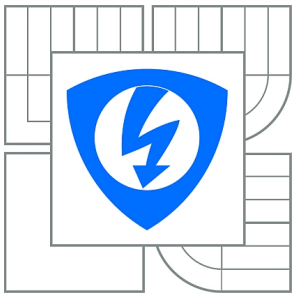
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

ROJOVÁ INTELIGENCE V MRDS

SWARM INTELLIGENCE IN MRDS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

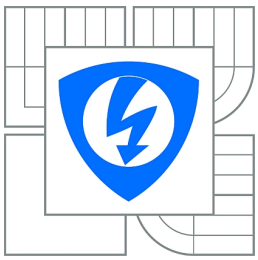
AUTOR PRÁCE — Bc. LUKÁŠ KUČERA

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR HONZÍK, Ph.D.

SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Lukáš Kučera	***ID:***	106581
Ročník:	2	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Rojová inteligence v MRDS

POKYNY PRO VYPRACOVÁNÍ:

Cílem diplomové práce je testování volně dostupného simulačního prostředí Microsoft Robotics Developer Studio (MRDS) na vybraných úlohách z oblasti rojové inteligence. Úkoly jsou následující:
- seznamte se s programováním v MRDS,
- seznamte se se základními typy úloh rojové inteligence a vypracujte stručnou rešerši,
- vyberte si dva opublikované experimenty, popište jejich realizaci, parametry experimentu a výsledky,
- realizujte experimenty v prostředí MRDS,
- srovnejte výsledky prezentované v publikacích s výsledky z vlastní implementace v MRDS,
- zhodnoťte využitelnost MRDS pro modelování a prezentování reálných robotů.

DOPORUČENÁ LITERATURA:

Abraham, A., Guo, H., and Liu, H. (2006). Swarm Intelligence : Foundations , Perspectives and Applications. Dimension Contemporary German Arts And Letters 26, 3-25. Available at: http://www.springerlink.com/index/73t8k16878415541.pdf.

Termín zadání:	6.2.2012	***Termín odevzdání:***	21.5.2012

Vedoucí práce: Ing. Petr Honzík, Ph.D.

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Rešeršní část diplomové práce je zaměřena na vybraná témata z oblasti rojové inteligence. Dále jsou na základě dvou publikací popsány experimenty studující chování skupiny robotů při shromažďování puků a při vyhledávání cíle. Vlastní práce je pak věnována zopakování těchto experimentů v Microsoft Robotics Developer Studiu (RDS), volně dostupném simulačním prostředí pro robotiku. Realizace obou experimentů v RDS je podrobně zdokumentována a dosažené výsledky jsou vyhodnoceny a srovnány s výsledky popsanými v publikacích. Na základě vlastních zkušeností jsou na závěr shrnuty základní vlastnosti, výhody a nevýhody vývoje v RDS.

Klíčová slova

Microsoft Robotics Developer Studio, rojová inteligence, simulace, robotika, C#

Abstract

The background research in this Master's thesis is focused on swarm intelligence. Further, there are two experiments described. They are based on released publications and they study behaviour of a group of robots during a puck gathering and during a target search. The actual thesis follows a repetition of these experiments in Microsoft Robotics Developer Studio (RDS), a free robotics simulation environment. The realization of both experiments in RDS is documented in detail and the achieved results are evaluated and compared with the results described in the publications. In conclusion, the thesis summarizes basic features, advantages and disadvantages of developing in RDS, based on a personal experience.

Keywords

Microsoft Robotics Developer Studio, swarm intelligence, simulation, robotics, C#

Bibliografická citace:

KUČERA, L. *Rojová inteligence v MRDS*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012. 56 s. Vedoucím diplomové práce byl Ing. Petr Honzík, Ph.D.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Rojová inteligence v MRDS jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **21. května 2012** ……

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Petru Honzíkovi Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce. Dále děkuji rodičům, babičce a přítelkyni za podporu a gramatickou kontrolu.

V Brně dne: **21. května 2012** ……

podpis autora

Obsah

1 ÚVOD

Diplomová práce je zaměřena na rojovou inteligenci a její aplikaci v Microsoft Robotics Developer Studiu (RDS). Jedná se o volně dostupné simulační prostředí pro robotiku, kde je možné levně a poměrně rychle testovat algoritmy umělé inteligence na dostatečně přesných modelech skutečných robotů, čímž lze odhalit chyby před samotnou fyzickou realizací.

V první části práce jsou popsány vybrané algoritmy rojové inteligence a jejich využití pro pohyb skupiny robotů. Část práce je také věnována stručnému úvodu do prostředí RDS. Na dvou publikovaných experimentech s rojovou inteligencí je ověřena vhodnost RDS pro další využívání. Experimenty jsou nejprve důkladně rozebrány a následně implementovány. Porovnáním výsledků z RDS s původními a ze zkušeností při práci s tímto prostředím je posouzena jeho využitelnost ve výuce.

2 UMĚLÁ INTELIGENCE SKUPINY ROBOTŮ

Na skupinu robotů lze pohlížet několika způsoby. Většina vychází z chování skupiny živých organismů. Jako zdroj inspirace lze využít skupinové chování živočichů od bezobratlých, přes ptáky, až po savce. Mezi bezobratlé patří například mravenci, kteří přes svoji nízkou mozkovou kapacitu dokážou stavět mohutná mraveniště a shromažďovat potravu. Ptáci mají již větší mozky, takže principy, které používají k utváření formací a seskupování se, nemusejí být až tak jednoduché, ale i přesto jsou rozhodně inspirativní. Mezi savci dominuje funkčností mozku člověk, který má nejvýraznější hierarchický systém, kdy jeden člověk velí druhému.

Z těchto všech zdrojů inspirace lze vytvořit tři základní typy chování robotů. Dílčími prvky při dělení jsou rovnocennost členů skupiny a náplň jejich práce. Prvním typem chování je rojová inteligence, kdy jsou si všichni zúčastnění rovnocenní a dělají stejnou práci. Příklad rovnocenných robotů s jinými pracovními náplněmi je robotický fotbal, kde má útočník jiný úkol než obranář, o brankáři ani nemluvě. Hierarchický systém je nejvýraznější v armádě, a proto je vhodnou ukázkou umělá inteligence skupiny, která se pohybuje ve formaci. Tento pohyb je společnou činností, ale jeden člen skupiny je vůdčí, nadřízený ostatním. Zbývá pouze kombinace nerovnocenných robotů s různými úkoly, což postrádá logiku. Pokud totiž budeme uvažovat roboty, kdy každý bude dělat něco jiného, nepotřebují samy o sobě žádné velení, a když už by jeden robot rozkazoval druhému, pak by bylo příhodné, aby mu i pomohl a ne jenom udával rozkazy [1, 2, 3, 10].

2.1 Rojová inteligence

Rojová inteligence, anglicky Swarm Intelligence (swarm = roj, hejno), čili SI, se týká seskupování a přeskupování. Jedná se o obecný pojem a uplatnění nemá pouze v robotice, ale například v informační technice při přeskupování souborů v síti na optimální pozici [1]. Tato práce se však zaměřuje na využití SI pro robotiku.

Rojová inteligence vychází z chování mravenců a ptáků. Ptáci se v hejnu uspořádávají do co nejvýhodnější pozice, stejně tak se mohou seskupit i roboty. U mravenců bylo vypozorováno shromažďování mrtvol na hromádky, což lze opět využít v robotice k shromažďování předmětů. Rojovou inteligenci lze využít například i pro prohledávání neznámých oblastí a vyhledávání objektů. Všechny tyto jevy budou nyní popsány blíže.

2.1.1 Seskupování členů skupiny

Na seskupování členů skupiny lze použít algoritmus PSO (Particle Swarm Optimization). Ten umožňuje dva druhy seskupování: lokální (lbest) a globální (gbest). Při globálním se pohyb členů skupiny řídí podle následujících rovnic:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_1\left(x_{ij}^{\#}(t) - x_{ij}(t)\right) + c_2 r_2\left(x_j^{*}(t) - x_{ij}(t)\right), \quad (1)$$

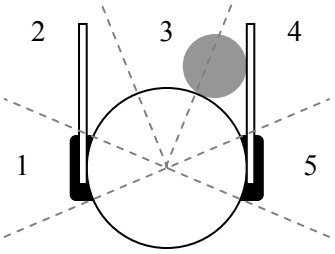
$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \quad (2)$$

kde t je pořadí předchozího časového vzorku, x_{ij} je poloha a v_{ij} je rychlost i-tého člena skupiny v j-té dimenzi (j lze volit například x a y pro 2D seskupování). Poloha $x_{ij}^{\#}$ je nejlepší dosažená poloha i-tého člena ve skupině a poloha x_j^{*} je hodnota nejvhodnější polohy od všech členů ve skupině. Koeficient w realizuje setrvačnost a je na něm závislý celkový prostor, který bude prozkoumán. Čím vyšší w je, tím dál členy dorazí. Koeficienty r_1 a r_2 jsou náhodná čísla od 0 do 1, která zajišťují rozmanitost členů skupiny. Konstanta c_1 má kladnou hodnotu a je nazývána koeficient složky seberozpoznání. Konstanta c_2 je také kladná a nazývá se koeficient společenské složky. Tyto dvě konstanty určují, jak moc se bude každý člen soustřeďovat na svou nejlepší dosaženou polohu a na nejlepší dosaženou polohu nejúspěšnějšího člena. Obvykle se volí $c_1 = c_2$, a to 2 nebo 1,49. Nové experimenty ukazují, že ještě lepších výsledků lze dosáhnout při volbě c_1 o něco vyššího než c_2, avšak jejich součet nesmí přesáhnout hodnotu 4.

Při lokálním druhu seskupování se nezjišťuje nejlepší poloha nejúspěšnějšího člena skupiny, ale pouze nejlepší poloha členů z blízkého okolí. Tato metoda je sice pomalejší, ale zamezuje uvíznutí skupiny v lokálním optimu, což se u globální metody může stát [1].

2.1.2 Shromažďování předmětů

Podle vzoru shromažďování mrtvol mravenci lze navrhnout úkol pro roboty. Vhodným příkladem je shromažďování puků několika roboty. Každý z nich má vpředu chytač, do kterého lehce zachytí puk ležící před ním a couváním z určitého místa na něm puk zanechá. Roboty (viz obrázek 1) jsou řízeny diferenciálním podvozkem se dvěma koly.



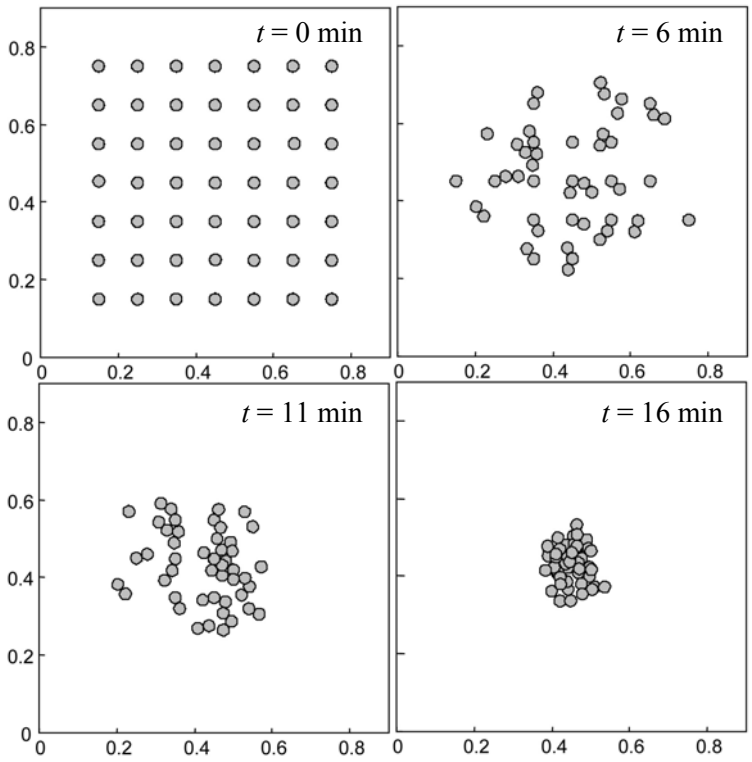
Obrázek 1 – Robot pro shromažďování puků a jeho 5 zón

Každý z robotů má 3 druhy senzorů. Jeden snímá přítomnost puku v chytači, další zjišťuje přítomnost puků v zónách 1-5, a poslední v těchto zónách vyhledává překážky, a to ať už statické, nebo pohyblivé (ostatní roboty).

Pro základní chování robotů stačí následujících pět pravidel:

1. pravidlo: *if* (není puk v chytači) *and* (je puk před robotem) *then* (vezmi puk),
2. pravidlo: *if* (je puk v chytači) *and* (je puk před robotem) *then* (zacouvej),
3. pravidlo: *if* (není puk v chytači) *and* (není puk před robotem) *then* (hledej min),
4. pravidlo: *if* (je puk v chytači) *and* (není puk před robotem) *then* (hledej max),
5. pravidlo: *if* (před robotem je překážka) *then* (vyhni se překážce).

První pravidlo zajišťuje sebrání puku a druhé zanechání puku u hromádky puků. Třetí pravidlo slouží k vyhledání místa s nejmenším (nenulovým) počtem puků na pracovní ploše, aby se k němu robot přiblížil a odebral puk. Ten potom podle pravidla čtyři přesune na místo s největší koncentrací puků, což je pravděpodobně již nakupená hromádka. Když robot narazí na překážku, zastaví a otáčí se náhodně okolo své osy, dokud před ním překážka nebude. Při tomto manévru by měl zůstat puk v chytači díky malým zobáčkům na jeho konci. Vyhýbání překážce má přednost před všemi ostatními pravidly.



Obrázek 2 – Průběh shromažďování puků dvěma roboty (převzato z [2])

Při použití bezpečné navigace robotu mezi překážkami k cíli se nahradí 3.-5. pravidlo pouze jedním pravidlem, a to:

if (není puk v chytači) *or* (před robotem je překážka) *then* (hledej cestu).

Hledání cesty se řídí podle bezpečné sítě, kde se z dat ze snímačů překážek a puků v pěti zónách vyhodnocují výsledná chování robotu. To může být buď otočení určitým směrem, couvání, nebo jízda vpřed.

Ukázka průběhu simulace shromažďování puků s použitím bezpečné navigace je na obrázku 2. Simulace byla prováděna v prostředí MATLAB se dvěma roboty a 49 puky. Nejprve se vytvářejí menší skupinky puků a nakonec vznikne jedna velká společná [2].

2.1.3 Průzkum a hledání objektů

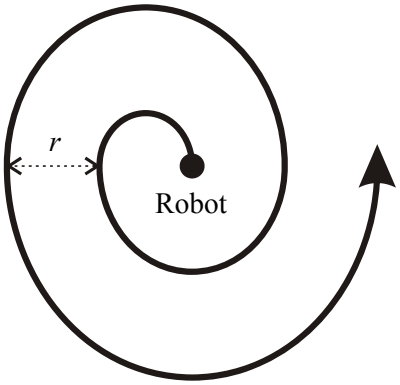
Při vyhledávání objektu více roboty je třeba zajistit mezi nimi komunikaci. Ta je omezena na určitou vzdálenost mezi nimi a je závislá na tvaru terénu. Pro detailnější rozbor bude uvažována praktická úloha, kdy robot hledá uvězněné dělníky při zasypání dolu.

K jejich lokalizaci lze využít některý z následujících signálů. Dělníci se mohou ozývat akusticky, čili svým hlasem. To lze ale pouze za předpokladu, že jsou při vědomí. Další možností je vystopovat dělníky podle rádiového identifikačního signálu, který se v moderní době v dolech používá právě k jejich lokalizaci a umisťuje se do helmy s lampou. Poslední signalizací přítomnosti zavalenin je možný únik podzemních plynů, které s narůstající vzdáleností od místa úniku ztrácejí na intenzitě vlivem difúze a vznikajících vírů.

Při komunikaci mezi roboty je uvažován model, kde se síla signálu doraženého od jednoho robotu k druhému určí jako

$$I(d) = \begin{cases} 0, & d > r \\ \dfrac{P}{d^2} + \eta, & d \le \mathrm{r} \end{cases}, \tag{3}$$

kde d je vzdálenost mezi roboty, r je dosah signálu, P je energie vysílače a η je Gaussovský šum. Stejná rovnice platí pro intenzitu signálu od cíle.

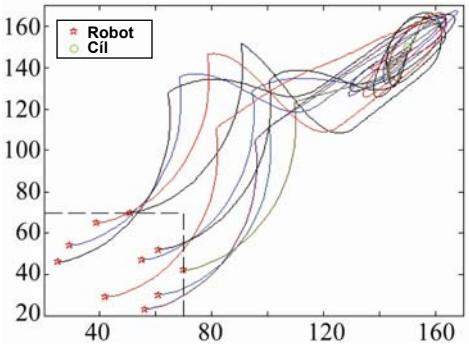


Obrázek 3 – Spirálovitá trajektorie záchranných robotů

Pro jeho vyhledávání se využívá model podobný PSO. V něm začínali členové skupiny v náhodných místech po celém prostoru. Při úkolu hledání uvízlých dělníků však roboty začínají z jednoho počátečního místa, respektive oblasti (na obrázku 4 vlevo dole vyznačena přerušovanou čarou). Jako nejvhodnější trajektorie robotů se

ukázala z hlediska prozkoumání veškeré plochy spirála, která má rozteč totožnou s dosahem signálu *r*. Ta je zobrazena na obrázku 3.

Jakmile některý z robotů narazí ve svém rozsahu signálu na cíl, pošle tuto informaci ostatním robotům v dosahu. Pozice robotu se označí jako nejlepší ve skupině a ostatní zamíří k němu. Jakmile zachytí signál i ostatní roboty, dostane se z intenzit přijímaných signálů od cíle jeho čím dál přesnější pozice. Na obrázku 4 je ukázka ze simulace, kde bylo náhodně rozmístěno 10 robotů v počáteční oblasti, a následně vysláno za účelem najít jeden cíl v prostoru ohraničeném zdmi. Roboty se nejprve vydaly po spirálovité trajektorii, dokud jeden z nich nezachytil signál od cíle. V tento moment byly všechny roboty navzájem v dosahu, a tak se informace dostala mezi všechny pátrací roboty, a ty souhrnně vyrazily k cíli podle algoritmu PSO [3].



Obrázek 4 – Trajektorie simulace rojového vyhledávání cíle (převzato z [3])

2.2 Robotický fotbal

Stejně jako lidé, mohou hrát fotbal i roboty. Existuje několik druhů robotického fotbalu. Mohou hrát roboty s končetinami či kolové, malé či velké, s interní či externí výpočetní jednotkou, s jednou společnou kamerou či s vlastní, apod. Tato práce se však zaměřuje na robotickou ligu malých robotků s diferenciálním podvozkem, společným řízením mimo tělo robotu a se společnou kamerou nad plochou hřiště.

Obrázek 5 – Fotbalový robot (vlevo) a fotbalové hřiště (vpravo) (převzato z [4])

Tento systém je založen převážně na dobrém vyhodnocování obrazu. Ukázka záběru z kamery nad hřištěm je na obrázku 5 vpravo. Každý robot na hřišti má jednu rozlišovací barvu, která určuje tým, a druhou barvu či tvar (to záleží na týmu, který způsob rozlišení si vybere), která identifikuje konkrétního hráče. Jak robot pro malou, střední a velkou fotbalovou ligu vypadá, je vidět na obrázku 5 vlevo. Roboty komunikují s počítačem pomocí rádiových frekvencí [4].

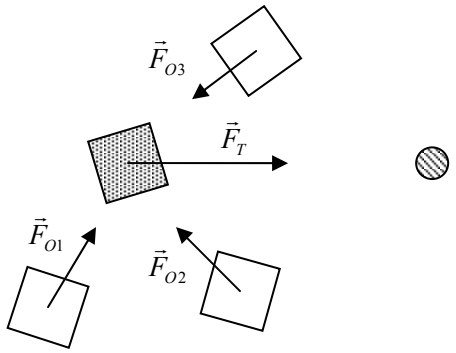
2.2.1 Pohyb robotických fotbalistů

Jeden způsob řešení pohybu robotů je pomocí vektorového pole. Jelikož je celý systém založen na počítačovém vidění, lze si prostor hřiště navzorkovat a do každého vzorku umístit vektor ideálního směru jízdy robotů. Každý z vektorů míří od překážek k míčku. Tím je dosaženo toho, že se roboty snaží dostat k míčku bez kolize s jiným. Každý robot totiž jede směrem, který je určený právě pro vzorek na jeho souřadnicích. Toto pole je třeba počítat pro každý robot zvlášť a v každém časovém intervalu aktualizovat. Proto je pro toto řešení potřeba velký výpočetní výkon [5].

Méně výpočetně náročný je způsob sčítání vektorů. Jak je ukázáno na obrázku 6, každá překážka (hráč opačného týmu) působí na robot odpudivou silou a míček působí silou přitažlivou $\vec{F}_T$. Po sečtení těchto sil dle vzorce

$$\vec{F} = \vec{F}_T + \frac{1}{n}\sum_{i=1}^{n}\vec{F}_{Oi}\,, \tag{4}$$

kde n je počet soupeřů a $\vec{F}_{Oi}$ je odpudivá síla i-tého soupeře závislá na vzdálenosti od robotu, vychází výsledná síla $\vec{F}$, která působí na robot a určuje ideální směr jízdy. Přitažlivá síla k míčku je konstantní, nezávislá na vzdálenosti, aby při příjezdu k němu došlo k jeho odpálení.



Obrázek 6 – Působení přitažlivého a odpudivých vektorů na robot

Pro jednodušší vzorové výpočty bude uvažován pouze jeden soupeř. Je vhodné přepočítat odpudivou sílu působící od soupeře k robotu na sílu kolmou k působení síly přitažlivé k míči. Tím se odstraní nežádoucí složka působící ve směru jízdy a způsobující buď zrychlení či zpomalení. Nákres těchto sil je na obrázku 7. Síly se řídí rovnicí

$$\vec{F} = \vec{F}_T + \vec{F}_{T90}, \tag{5}$$

kde $\vec{F}_{T90}$ je již zmíněná kolmá síla, jejíž velikost se spočítá jako

$$\left|\vec{F}_{T90}\right| = k \cdot \mathrm{G}(x, \mu, \sigma), \tag{6}$$

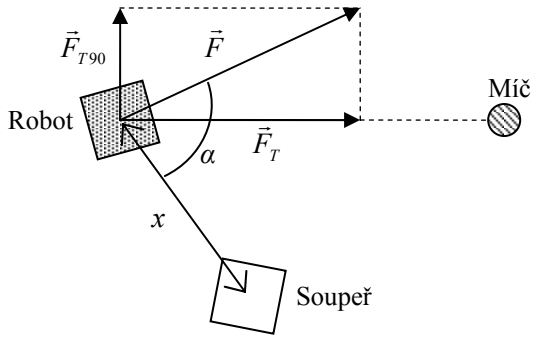
kde k je konstanta realizující měřítko a G(x, μ, σ) je nenormalizovaná funkce Gaussovského rozložení. Ta je definována rovnicí

$$\exp\left(-\frac{(x-\mu(\alpha))^2}{2\sigma^2}\right), \tag{7}$$

kde x je vzdálenost mezi robotem a soupeřem, σ^2 je rozptyl a $\mu(\alpha)$ je funkce úhlu α definovaná jako

$$\mu(\alpha) = \frac{2r_1}{1+\exp\frac{\alpha}{\tau}}, \tag{8}$$

kde τ je konstantní úhel určující strmost funkce a r_1 je maximální hodnota μ při nulovém úhlu. S rostoucím úhlem funkce $\mu(\alpha)$ exponenciálně klesá, takže je dosaženo větších sil, když robot míří k soupeřovi, než když se mu vyhýbá [6].



Obrázek 7 – Vektory od míče a soupeře působící na robot

Dalším způsobem řešení pohybu robotů, který je jakousi kombinací předchozích dvou, je způsob terénního mapování hřiště. Tam, kde jsou hráči soupeře, je uvažován zvýšený terén, naopak kde jsou spoluhráči, je snížený terén. Roboty se potom snaží jezdit po rovině a nejezdit příliš do kopce. Jedná se prakticky o stejný princip, jako je vektorové pole, jenom je lépe realizovatelné přihrávání mezi hráči, protože když stojí robot blízko soupeře, vlivy snížení a zvýšení se vykompenzují. Nejlepší cestu terénní mapou lze nalézt pomocí Dijkstrova algoritmu [7].

2.2.2 Typy robotických fotbalistů

Stejně jako ve fotbale s lidmi, i v robotickém jsou tři druhy hráčů. Útočníci, obránci a brankář. V malé fotbalové lize hraje od každého typu hráče jeden [4], ve střední fotbalové lize je na hřišti za každý tým pět hráčů, ve velké fotbalové lize je hráčů jedenáct. Pro případ jedenácti hráčů je větší hřiště (440×280 cm) a dvě snímací kamery,

aby zabraly celou hrací plochu. Na hřišti je 5 obránců, 5 útočníků a jeden brankář [8]. Tato práce se ale soustředí na hru pěti hráčů. Jedná se tedy o 2 útočníky, 2 obránce a jednoho brankáře na hřišti 220×180 cm.

Pro obránce a útočníky jsou definovány dva režimy. Následování trasy a střílení. Při střelbě dosahuje robot vyšší rychlosti než při přesunu [5] a snaží se vystřelit míč do nejvolnějšího prostoru na hřišti směrem k soupeřově brance, nebo ho dokonce přihrát spoluhráči. Kam míč pošlou, záleží na zvoleném stylu hry [7].

Brankář má úlohu jinou. Pohybuje se pouze po přímce před brankou a cílem jeho pozornosti je míč a jeho trajektorie. Pokud je míč daleko od branky, brankář stojí uprostřed. Jakmile se míč začne blížit, z jeho trajektorie se spočítá přímka a brankář se na ni postaví, bráníce tak míči postupu do branky [5]. Některé týmy dokonce implementují brankáři algoritmy pro zastavení míče a jeho vykopnutí vysokou rychlostí [7].

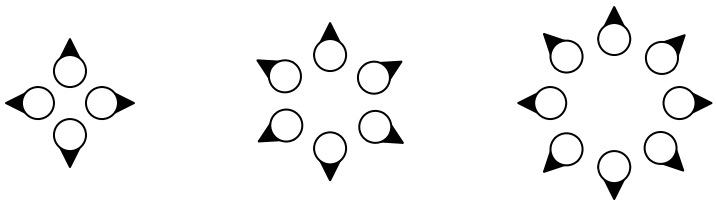
V robotickém fotbale záleží jak na schopnosti určení dobré dráhy robotů, tak na dobrém a rychlém vyhodnocení obrazu z kamery. Samozřejmostí je spolehlivá rádiová komunikace mezi robotem a řídicím počítačem. Hardwarové vybavení robotů hraje v posledních letech také velkou roli.

2.3 Přesun ve formaci

Skupinu robotů s hierarchickým systémem lze využít například při odstraňování min či prohledávání nebezpečných oblastí, kde dražší a kvalitnější kus je nadřízený ostatním a raději se drží stranou a udílí rozkazy. Podřízené roboty mohou mít jen omezené funkce a co budou dělat, za ně rozhoduje podle naplánování nadřízený robot [9].

Hierarchii lze však použít i pro řízení formace, kde si jsou roboty rovny, ale jeden (velitel) určuje cestu. Existují tři základní druhy formací: pevná, škálovatelná a vznikající. Při pevné formaci jsou přesně určená místa, kde má který člen být, přičemž jedno je označeno jako velitelova pozice. Její souřadnice jsou z pohledu formace nulové. Velitel se pohybuje podle plánované trasy a zbytek skupiny pouze drží formaci v závislosti na volitelově pohybu a natočení.

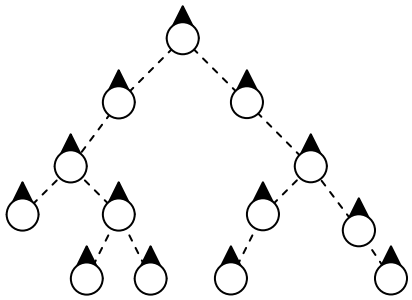
Škálovatelná formace se od pevné liší tím, že v závislosti na počtu robotů ji lze rozšířit. Ukázka obranné kruhové škálovatelné formace je na obrázku 8. Tento druh formace se ale pro robotiku využije málokdy, například při poruše některého člena skupiny.



Obrázek 8 – Škálovatelná obranná formace se 4, 6 a 8 členy

Vhodným příkladem vznikající formace je klínová (viz obrázek 9). Na špičce formace je pozice velitele. Ostatní členové se přiřazují do příslušného tvaru a snaží se o vyvážený poměr zaplněnosti stran. Pokud je ve skupině více členů, mohou uvnitř klínu vznikat menší elementy stejného tvaru, čímž vznikne rozvětvená struktura. Právě kvůli těmto nepravidelnostem je třeba vývoji algoritmů pro samonalezení místa ve formaci věnovat více pozornosti.

Ať už je zvolen jakýkoliv typ formace, pravidla pro přesun jsou stejná. Souřadnice polohy skupiny jsou totožné se souřadnicemi velitele. Ten se pohybuje směrem k cíli a členové skupiny ho následují a snaží se udržet formaci. Pro případ překážek v cestě má každý člen svůj vlastní algoritmus pro vyhýbání a jakmile se překážce vyhne, zařadí se opět do formace. Toho lze dosáhnout, pokud se velitel nebude pohybovat maximální rychlostí.



Obrázek 9 – Vznikající klínová formace

Pokud by se však vyhýbal překážce velitel, změnila by se relativní poloha celé formace, dokonce včetně natočení. Tento nedostatek lze vyřešit tak, že o natočení nebude rozhodovat přímo velitel, ale jeho imaginární zástupce, který se nemusí vyhýbat překážkám. Tím je zajištěn plynulý posuv celé formace.

Pro zajištění plynulého pohybu členů ve formaci je třeba posunout v každém časovém kroku cílové pozice mírně před současné pozice členů. V závislosti na vzdálenosti cíle budou členy měnit svou rychlost. Pokud je tedy požadováno, aby se formace přesunovala rychlostí v, pak pro cílenou pozici $p_{cíl}$ platí vzorec

$$p_{cíl} = p_s + k \cdot v\,, \tag{9}$$

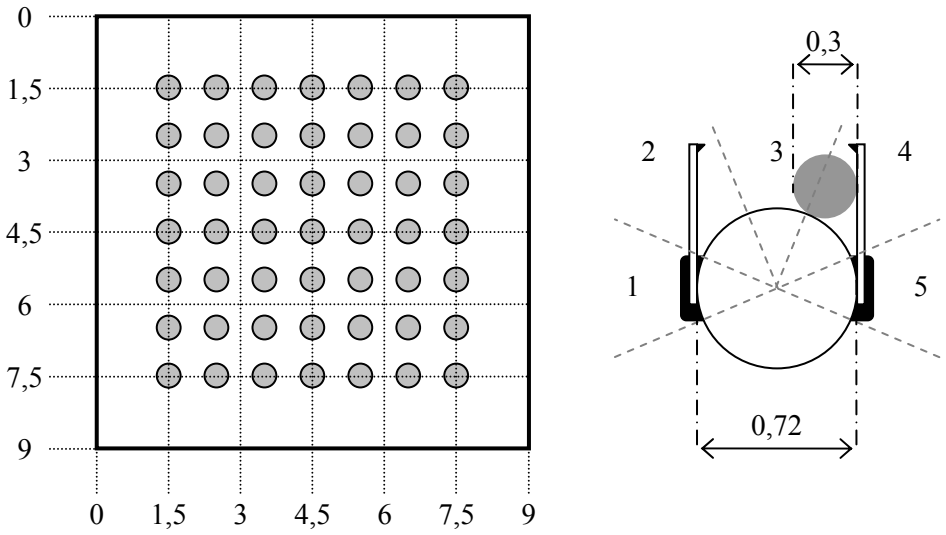
kde p_s je současná pozice a k je koeficient měřítka. Pokud má formace zůstat na místě, je lepší změnit přístup a nastavit polohu konstantní, aby nedošlo k případnému driftu [10].

3 POPIS VYBRANÝCH EXPERIMENTŮ

Z oblasti rojové inteligence byly vybrány dva experimenty, které budou zopakovány v RDS. V této kapitole budou oba experimenty popsány. První experiment se týká shromažďování objektů, konkrétně puků [2], a druhý experiment se týká vyhledávání cíle [3].

3.1 Experiment s puky

Jak je vidět vlevo na obrázku 10, puky o průměru 0,3 jednotek jsou pravidelně rozmístěny na zdmi ohraničené ploše 9×9 jednotek s rozestupem jedné jednotky. Pro experiment bylo zvoleno 49 puků, takže první leží na souřadnicích [1,5; 1,5], druhý [2,5; 1,5], atd. Další řádek bude pokračovat osmým pukem, a to na souřadnici [1,5; 2,5].



Obrázek 10 – Rozmístění 49 puků na ploše 9×9 a robot s chytačem

Dále se do plochy náhodně umístí jeden nebo dva roboty. Z každé varianty se budou vyhodnocovat data. Robot má kruhový půdorys o průměru 0,72 jednotek a je řízený diferenciálním podvozkem. Vpředu je na robotu umístěný chytač puků. Jak je vidět na obrázku 10 vpravo, puk se v chytači zachytí při pohybu dopředu a volně z něj vyklouzne, pokud robot zacouvá. Aby puk nemohl vyklouznout z chytače při otáčení robotu, musí být na koncích chytače drobné zobáčky.

Aby měl robot zpětnou vazbu s prostředím, potřebuje také snímače. Jeden snímač snímá přítomnost puku v chytači, druhý snímá přítomnost překážek v zónách 1-5 (viz obrázek 10 vpravo) a třetí snímač ve stejných zónách zjišťuje koncentraci puků. Ta je dána procentem puků v určité zóně a je využita k vylepšenému plánování trasy robotu.

Trasa robotů by mohla být náhodná, ale jelikož se ukázalo, že při použití informace o koncentraci puků probíhá experiment rychleji, bude využit tento princip. Roboty bez

puku budou mířit do místa s nejnižší nenulovou koncentrací, což reprezentuje nejmenší shluk puků. Roboty s pukem budou naopak mířit do místa s nejvyšší koncentrací, aby ještě zvětšily největší shluk puků. Tento pohyb a zároveň i sběr puku a vyhýbání překážkám (jiný robot či zeď) shrnuje 5 jednoduchých pravidel:

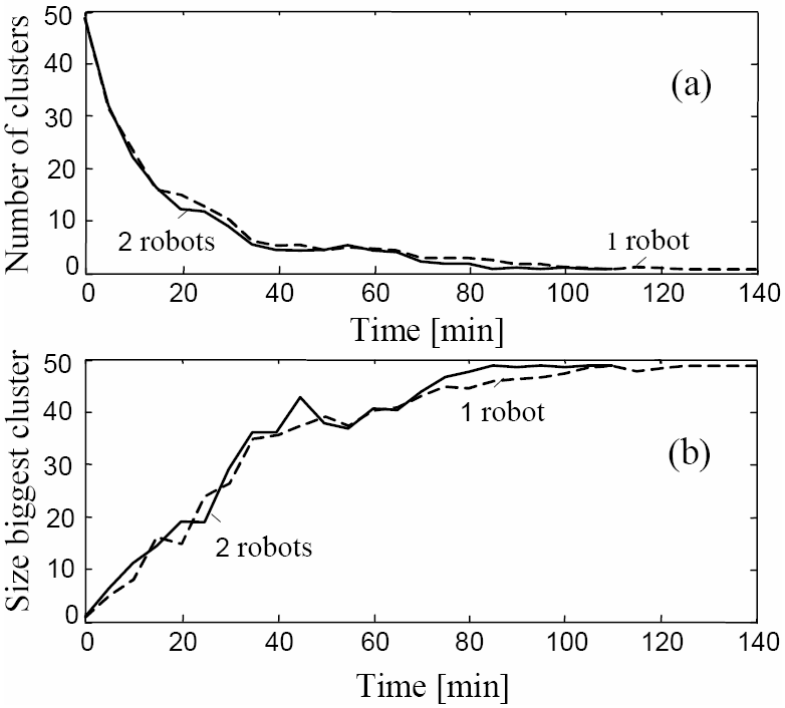
1. *pravidlo:* *if* (není puk v chytači) *and* (je puk před robotem) *then* (vezmi puk),
2. *pravidlo:* *if* (je puk v chytači) *and* (je puk před robotem) *then* (zacouvej),
3. *pravidlo:* *if* (není puk v chytači) *and* (není puk před robotem) *then* (hledej min),
4. *pravidlo:* *if* (je puk v chytači) *and* (není puk před robotem) *then* (hledej max),
5. *pravidlo:* *if* (před robotem je překážka) *then* (vyhni se překážce).

V práci [2] byly vyhodnoceny dvě varianty experimentů. První varianta byla prováděna s pomocí bezpečné (imunní) navigace a druhá bez ní. Pro jednodušší implementaci do RDS byla pro zopakování experimentu zvolena pouze varianta bez bezpečné navigace, která používá právě již zmíněných pět pravidel.

Tabulka 1 – Přehled parametrů simulace shromaždování puků

Rozměry hřiště	9×9
Počet puků	49
Souřadnice prvního puku	[1,5; 1,5]
Rozteč mezi puky	1
Poloměr puku	0,15
Poloměr robotu	0,36
Počet snímaných zón	5
Úhel jedné zóny	45°
Počet robotů	1 nebo 2
Pozn.: Rozměry jsou uváděny v obecných jednotkách	

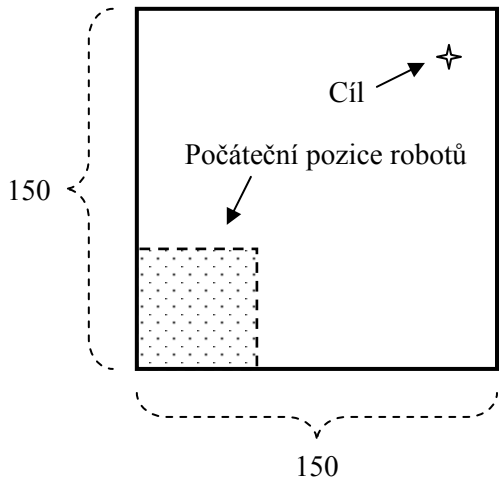
Pohybem robotů vznikají shluky puků. Jako shluk jsou považovány puky, které jsou od sebe vzdáleny maximálně o průměr jednoho puku, což je 0,3 jednotek. Základní parametry simulace jsou uvedeny v tabulce 1. V práci byl vyhodnocován časový průběh počtu shluků puků a časový průběh počtu puků v největším shluku. A to jak pro jeden robot, tak pro dva. Na obrázku 11 jsou zprůměrované výsledky z osmi experimentů (4 s jedním a 4 se dvěma roboty) provedených autory práce v prostředí MATLAB.



Obrázek 11 – Časový průběh počtu shluků (a) a časový průběh počtu puků v největším shluku (b) pro variantu se dvěma i jedním robotem (převzato z [2])

3.2 Experiment s vyhledáváním

V experimentu byla použita čtvercová plocha o délce hrany 150 jednotek, která je ohraničena zdmi. Deset pátracích robotů bylo náhodně rozmístěno do levého dolního rohu na prostor 50×50 jednotek. Do zbylého prostoru byl náhodně umístěn hledaný cíl. Publikovaná verze experimentu má cíl pevně umístěný v pravém horním rohu na souřadnicích [130; 130]. Počáteční sestava tohoto experimentu je na obrázku 12.



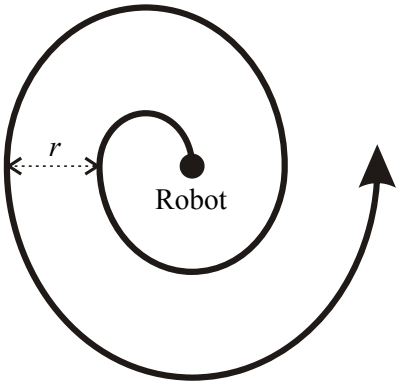
Obrázek 12 – Plocha pro vyhledávání cíle rojem robotů

Každý robot zná svou polohu a komunikuje s ostatními roboty a cílem. Narozdíl od signálu mezi roboty signál od cíle neobsahuje žádnou informaci (pouze identifikační číslo). Komunikační signál ale mění se vzdáleností svou intenzitu dle rovnice

$$I(d) = \begin{cases} 0, & d > r \\ \dfrac{P}{d^2} + \eta, & d \le \mathrm{r} \end{cases}, \tag{10}$$

kde d je vzdálenost od zdroje signálu, r je maximální dosah signálu (v simulaci roven 50 jednotkám), P je energie vysílače (1000 jednotek) a η je Gaussovský šum přidaný kvůli větší podobnosti realitě. Díky tomu lze spočítat, jak daleko od robotu se zhruba cíl nachází. Je patrné, že pokud nebude robot v dosahu signálu od cíle, žádný signál nedostane a musí cíl nejprve naslepo hledat.

Roboty mají dva typy pohybu. První typ je pohyb po spirále o rozteči r (viz obrázek 13), což je zároveň dosah signálů v simulaci. To proto, aby robot nemohl minout žádné místo, kde by mohl být cíl, ale nedosahoval by od něj signál. Tento typ pohybu vykonává robot jen tehdy, když k němu signál od cíle nedosahuje. Jakmile se robot ocitne v dosahu signálu, změní svůj pohyb na typ dle PSO. Informaci o přijetí signálu pošle i ostatním robotům v dosahu a ty také změní typ svého pohybu.



Obrázek 13 – Spirálovitá trajektorie záchranných robotů

Nový pohyb robotů se řídí rovnicemi PSO zmíněnými už v kapitole 2.1.1, a to rovnicí 1 pro rychlost a rovnicí 2 pro pozici. Tyto rovnice však mají příliš velký krok, a proto byly přepočítány na rovnice s krokem nižším než 1. Původní rychlost byla označena jako $v_{ijexpect}$ a z ní byla následně spočítána nová rychlost a pozice:

$$v_{ijexpect}(t+1) = wv_{ij}(t) + c_1 r_1 \left(x_{ij}^{\#}(t) - x_{ij}(t)\right) + c_2 r_2 \left(x_j^{*}(t) - x_{ij}(t)\right), \tag{11}$$

$$v_{ij}(t+\Delta t) = v_{ij}(t) + \frac{v_{ijexpect}(t+1) - v_{ij}(t)}{T}, \tag{12}$$

$$x_{ij}(t+\Delta t) = x_{ij}(t) + \Delta t \cdot v_{ij}(t+\Delta t), \tag{13}$$

kde t je pořadí předchozího časového vzorku, x_{ij} je poloha a v_{ij} je rychlost i-tého člena skupiny v j-té dimenzi. Poloha $x_{ij}^{\#}$ je nejlepší dosažená poloha i-tého člena ve skupině a poloha x_j^{*} je nejlepší dosud dosažená poloha z celé skupiny. Koeficient w realizuje

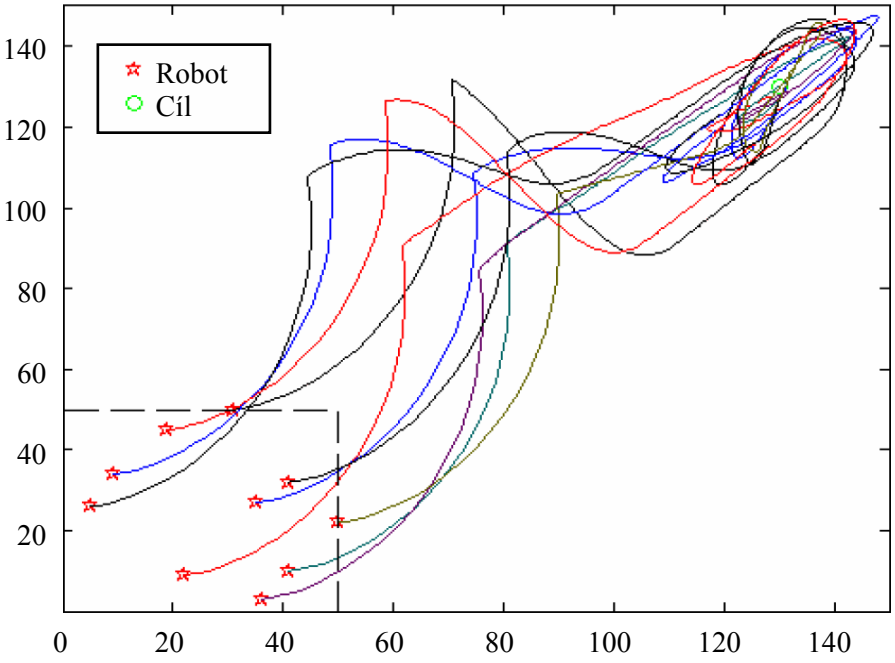
setrvačnost a jeho hodnota nebyla v práci definována. Proto ho bude třeba zvolit pokusně. Koeficienty r_1 a r_2 jsou náhodná čísla od 0 do 1. Konstanty c_1 a c_2 rovněž nebyly definovány, ale z [1] je známo, že se volí sobě rovny, a to buď 2 nebo 1,49. Časový parametr T = 0,6 s je perioda vzorkování simulace a Δt je perioda vzorkování pohybu robotů, kterou v simulaci volili autoři rovnu 0,4 s. Výsledná rychlost robotu je navíc omezena maximální rychlostí 2 jednotky za sekundu. Všechny parametry simulace jsou shrnuty v tabulce 2.

Tabulka 2 – Přehled parametrů simulace vyhledávání cíle

Rozměry hřiště	150×150
Rozměr startovní pozice	50×50
Počet pátracích robotů	10
Souřadnice cíle	[130; 130]
Maximální dosah signálu (r)	50
Energie vysílače (P)	1000
Perioda vzorkování simulace (T)	0,6 s
Perioda vzorkování pohybu (Δt)	0,4 s
Koeficienty c_1 a c_2	2 nebo 1,49
Maximální rychlost robotu	2/s
Pozn.: Rozměry a energie jsou uváděny v obecných jednotkách	

Na velikosti robotu pro účel simulace nezáleží, avšak je vhodné udržovat jeho velikost v rozumných mezích. Příliš malý robot by v realitě neuvezl dostatečné vybavení a příliš velký by mohl kolidovat s ostatními pátracími roboty. Typ pohonu robotu také nebyl specifikován, ale je třeba volit takový, aby co nejlépe sledoval požadavky na změnu rychlosti.

Výsledky původního experimentu jsou vyobrazeny pomocí různobarevných čar v prostoru pracovní plochy, které reprezentují trasy jednotlivých robotů. Při podobném rozmístění robotů by tedy průběhy tras měly být přibližně stejné. Hlavní však je, aby roboty nalezly úspěšně cíl. Publikovaný výstup ze simulace původního experimentu je na obrázku 14.



Obrázek 14 – Výstup ze simulace rojového vyhledávání cíle (převzato z [3])

4 REALIZACE V RDS

Tato kapitola pojednává o vývojovém a simulačním prostředí Microsoft Robotics Developer Studio (RDS) [11] a o realizaci obou experimentů z předchozí kapitoly v jeho verzi 2008 R3.

4.1 Microsoft Robotics Developer Studio

Microsoft RDS [11] je pro studenty volně stažitelné z Microsoft Developer Network Academic Alliance (MSDN AA). Po instalaci se studio propojí s Microsoft Visual Studiem, kde je možné programovat části kódu v jazyku C#. Další součástí instalace jsou mimo jiné ukázkové simulace a dva speciální vývojové programy pro RDS: Visual Programming Language a DSS Manifest Editor.

Chod RDS závisí na třech dílčích částech:

Concurrency and Coordination Runtime (CCR) – knihovna obsažená v .NET, která byla uvedena současně s RDS a umožňuje aplikacím asynchronní chod bez nutnosti vytvářet vlákna a hlídat blokující stavy. Díky CCR je možné dělat více operací najednou, čili například otáčet kolem robotu a zároveň snímat vzdálenost překážek.

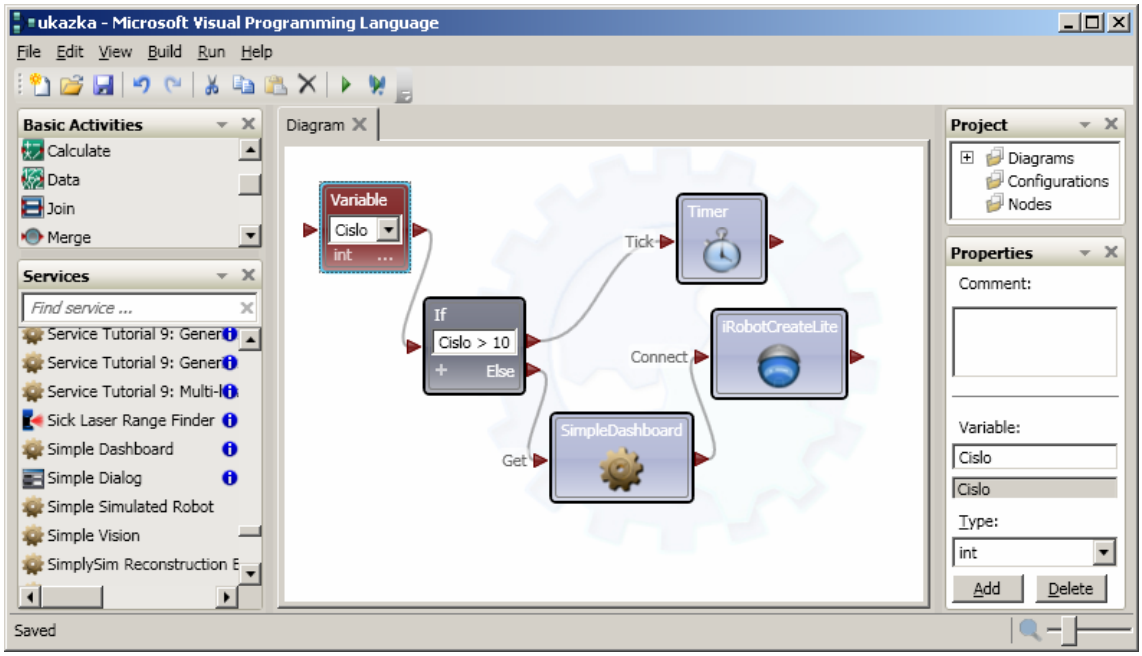
Decentralized Software Services (DSS) – RDS funguje na principu asynchronních služeb, jež lze sledovat právě pomocí DSS. DSS se inspiruje funkcí Representational State Transfer (REST), který je používaný pro chod internetu. REST využívá mimo jiné i formát XML (Extensible Markup Language). Jelikož je tímto pro formát XML ověřena rychlost, používá ho i RDS.

Common Language Runtime (CLR) *2.0* – základní stavební kámen celého RDS. Pomocí CLR přistupují služby k .NET knihovnám.

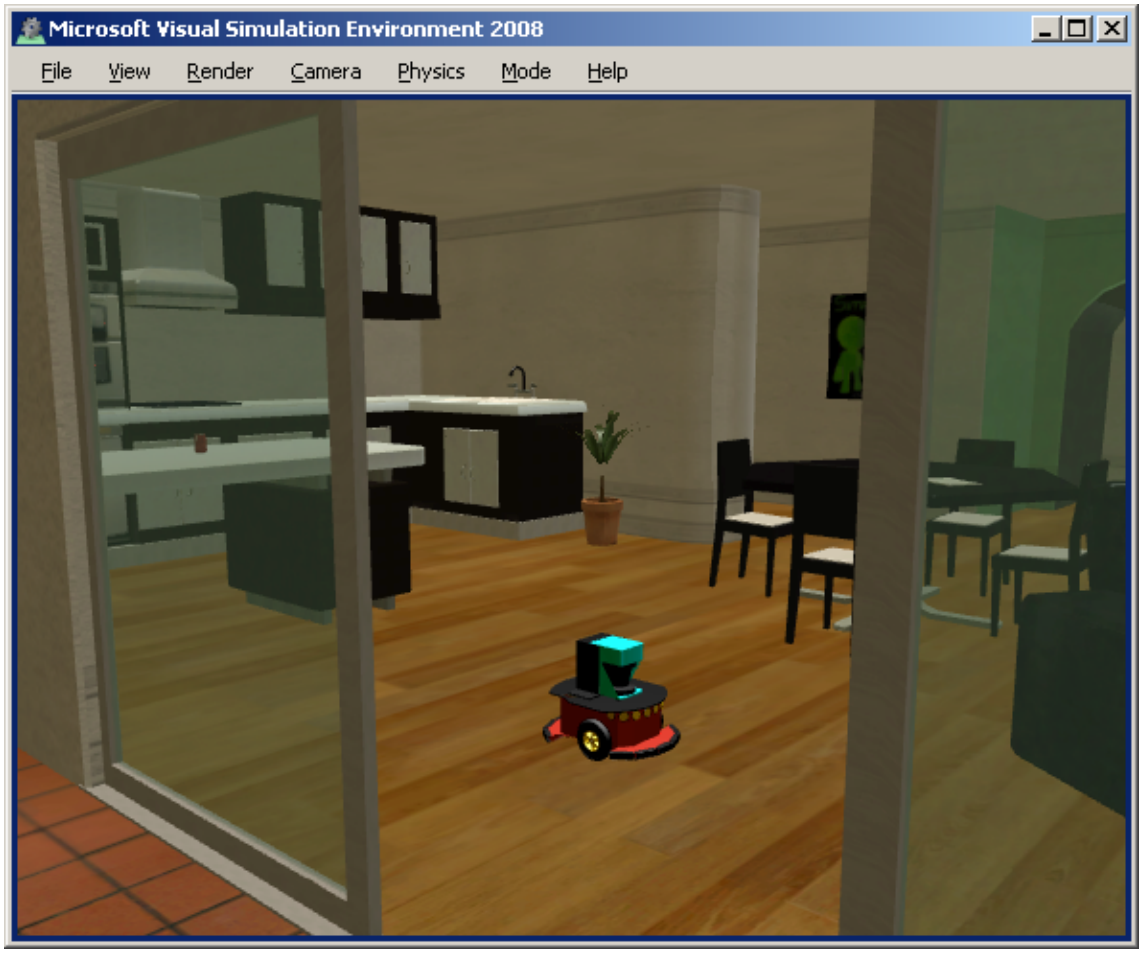
Jak již bylo zmíněno, mezi součástmi RDS se nachází DSS Manifest Editor. Zde lze editovat soubory XML, v nichž je uloženo, které služby budou propojeny.

Visual Programming Language (VPL) slouží k práci s RDS bez použití Visual Studia, avšak část kódu lze programovat v C# a následně využít ve VPL. Aplikace VPL je založena na grafickém programování. Vkládají se zde dílčí bloky a spojují se čárami. Na obrázku 15 je zobrazeno, jak vypadá uživatelské prostředí VPL.

Poslední částí RDS, která stojí za zmínku, je Visual Simulation Environment (VSE). Toto prostředí kombinuje model fyziky, který používá AGEIA PhysX engine, a 3D zobrazení simulace, které využívá Microsoft DirectX 9 a Microsoft XNA. Do VSE se vkládají jednotlivé 3D entity, na které působí fyzika. Ukázka, jak může vypadat simulace, je na obrázku 16. Každá židle, stůl i robot jsou volně přemístitelné objekty, na které působí gravitace. Naopak stěny či kuchyňská linka jsou pevně zapasovány a gravitace na ně nemá vliv.

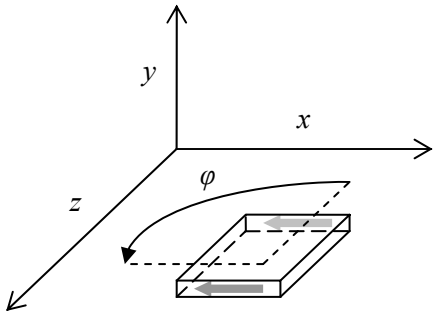


Obrázek 15 – Uživatelské prostředí Visual Programming Language



Obrázek 16 – Ukázka grafického prostředí simulace (VSE) v RDS

Jelikož je VSE trojrozměrné prostředí, sestává souřadnicový systém ze tří os. Osy určující rovinu, po které se roboty mohou pohybovat, jsou *x* a *z* (viz obrázek 17) a na ně je vzhůru kolmá osa *y*. Tíhová síla v simulačním prostředí RDS působí po směru osy *y* a má záporné znaménko, aby byly předměty přitahovány k podložce a ne vzhůru. Pro výpočet polohy a natočení robotu není třeba se osou *y* zabývat. Jakákoliv změna polohy nebo natočení vůči ose *y* by byla v simulaci dokonce nežádoucí. Proto bude v dalších kapitolách jako osa *y* označována vodorovná osa *z*.



Obrázek 17 – Souřadnicový systém v RDS

Na obrázku 17 je také vyobrazen objekt, který je otočen podél své osy rovnoběžné se svislou osou *y*. Při nulovém natočení směřuje objekt (například robot) proti ose *z*. Při otočení o libovolný úhel φ se objekt natáčí v kladném směru (proti směru hodinových ručiček). Na obrázku je ukázán případ, kdy se otočením o 90° změní směr, kam míří objekt, na opačný, než má osa *x*.

4.2 Realizace experimentu s puky

Původní experiment probíhal ve velice malém měřítku, kde byly jednotkami decimetry. Pro lepší simulaci bylo měřítko v RDS zvoleno v metrech. Vzniklo tak hřiště o rozměrech 9×9 metrů, ohraničené 60 cm vysokými a metr tlustými zdmi.

Dále je třeba vytvořit puky. V RDS jsou čtyři druhy základních objektů. Prvním je kvádr (*Box*), druhým je koule (*Sphere*), třetím je kapsle (*Capsule*) a posledním je kolo (*Wheel*). Válec, kterým by šel puk reprezentovat, mezi základními tvary není. Tvar kola je kvůli své specifičnosti v simulačním prostředí pro účel puku nepoužitelný.

Jako jedna z možností, jak tento problém vyřešit, je použití více kvádrů s různým natočením, které by vytvořily aproximaci válce. Kvůli vzájemnému silovému působení sil mezi dílčími částmi objektu však tento způsob nefunguje. Kvádry neustále kmitají a vzniká tak deformované těleso, které se rozkmitá a začne poskakovat.

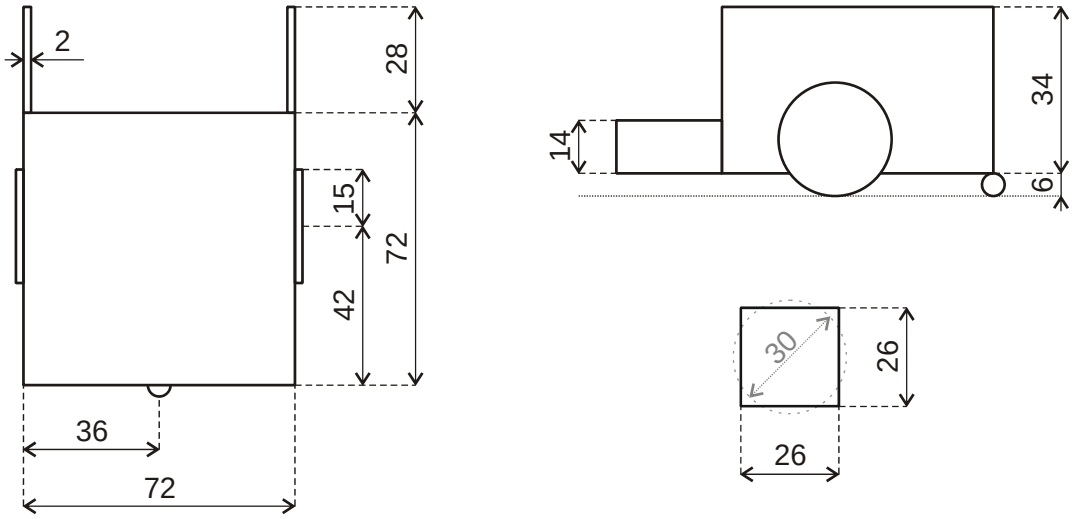
Čistší možností se zdá využití externího editoru 3D objektů a následné importování válce do simulačního prostředí RDS. To je možné, ale bohužel pouze co se týče grafické části. Vytvoří se objekt, který má například tvar kostky a přiřadí se mu vzhled válce ze souboru. Objekt následně vypadá jako válec. Jako válec se ale nechová. Fyzikální prostředí RDS s tímto objektem nadále zachází jako s krychlí. To je pro

simulaci puku nepoužitelné, protože by se puky buď prolínaly (když by byl kvádr menší než válec), nebo by se v místech rohů kvádru nikdy nemohly dotknout (kdyby měl kvádr stejný průměr jako válec).

V důsledku těchto omezení byly puky ve tvaru válce nahrazeny puky ve tvaru kostek. Aby bylo dosaženo přibližně stejných vlastností puků, byla jejich hrana (a) určena jako průměr mezi opsaným a vepsaným čtvercem do kružnice o průměru d, což je průměr puku (čili 30 cm):

$$a = \frac{1}{2}\left(d + \frac{d}{\sqrt{2}}\right) = \frac{1}{2}\left(30 + \frac{30}{\sqrt{2}}\right) = 25{,}6066 \doteq 26\,. \qquad (14)$$

Nový puk je tedy krychle o hraně 26 cm (viz obrázek 18 vpravo dole).



Obrázek 18 – Půdorys a nárys robotu a puku (rozměry v cm)

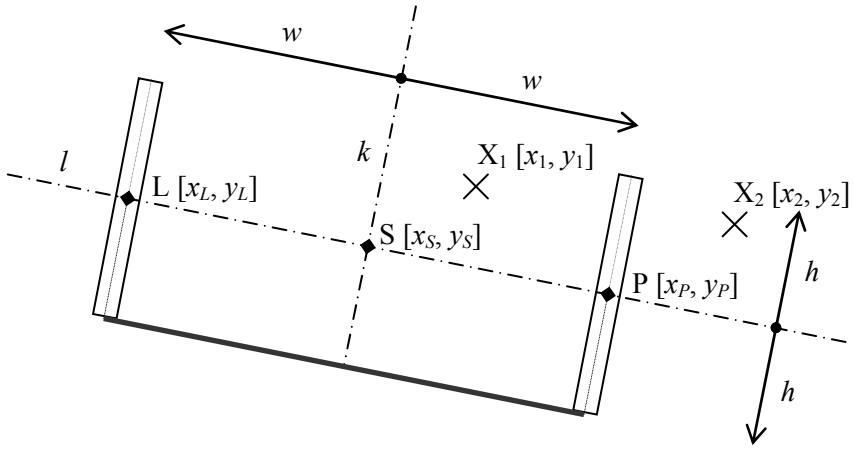
Původní robot měl také tvar válce. Ten je opět nahrazen kvádrem (viz obrázek 18). Rozměry jsou ale tentokrát zachovány, takže původní průměr je roven hraně robotu. Délka chytače byla zvolena 28 cm, aby byl o něco delší než je hrana puku, ale zároveň ne příliš dlouhý. Šířka byla určena na 2 cm, aby nebyl chytač příliš tlustý a zároveň aby nebyl tak tenký, že by byl v realitě buď ohebný nebo křehký. Háčky na konci chytače byly vynechány, protože pokud se robot neotáčí příliš prudce, hranatý puk z chytače vlivem tření nevyklouzne.

Oproti původnímu má nový robot kola tři. Levé a pravé o poloměru 15 cm slouží jako pohon a zadní kolečko s poloměrem 3 cm pouze vyvažuje rovnováhu. Přední kola byla od středu robotu posunuta o 6 cm dopředu, aby se robot nepřeklápěl dopředu vlivem váhy chytačů. Jejich hmotnost je spolu s hmotnostmi ostatních objektů v simulaci uvedena v tabulce 3.

Tabulka 3 – Přehled hmotností objektů v simulaci s puky

Zdivo	400 000 g
Puk	50 g
Robot	9 401 g
- tělo	9 000 g
- levé kolo	100 g
- pravé kolo	100 g
- zadní kolo	1 g
- levý chytač	100 g
- pravý chytač	100 g

Snímač přítomnosti puku v chytači funguje na principu výpočtu vzdálenosti bodu od přímky. Na obrázku 19 je vyobrazen chytač, u nějž známe vzdálenost *w* (polovina rozteče chytače), délku *h* (polovina délky chytače) a souřadnice středů levé (L) a pravé (P) části chytače. Body X_1 a X_2 reprezentují středy dvou puků, u nichž bude v ukázce zjištěna přítomnost v chytači.



Obrázek 19 – Detekce puku v chytači pomocí vzdálenosti bodu od přímky

Mezi body L a P lze uvažovat úsečku. Po dosazení souřadnic bodů L a P lze zjistit vektor *r*, reprezentující tuto úsečku, a také lze spočítat souřadnice jejího středu S:

$$\vec{r} = \left(r_x; r_y\right) = \left(x_P - x_L; y_P - y_L\right), \tag{15}$$

$$x_S = \frac{x_P + x_L}{2} \quad y_S = \frac{y_P + y_L}{2}. \tag{16}$$

Prodlouží-li se úsečka LP, vznikne přímka *l*, kterou lze popsat rovnicí

$$r_y \cdot x - r_x \cdot y + d = 0, \tag{17}$$

kde *d* je posunutí přímky. Lze ho spočítat dosazením souřadnic bodu, který leží na přímce. Pro bod S vyjde *d* následovně:

$$\begin{aligned} r_y \cdot x_S - r_x \cdot y_S + d &= 0 \\ d &= r_x \cdot y_S - r_y \cdot x_S \end{aligned}. \tag{18}$$

Výsledná rovnice přímky *l* je tedy

$$r_y \cdot x - r_x \cdot y - r_y \cdot x_S + r_x \cdot y_S = 0\,. \tag{19}$$

Vzdálenost bodu o souřadnicích [*X*, *Y*] od přímky ve tvaru $ax + by + c = 0$ lze spočítat podle vzorce

$$v = \frac{\left|a \cdot X + b \cdot Y + c\right|}{\sqrt{a^2 + b^2}}\,. \tag{20}$$

Dosazením z rovnice19 do vztahu 20 lze získat vzdálenost bodu X [*x, y*] od přímky *l*:

$$v_l = \frac{\left|r_y \cdot x - r_x \cdot y - r_y \cdot x_S + r_x \cdot y_S\right|}{\sqrt{r_x{}^2 + r_y{}^2}}\,. \tag{21}$$

Aby byl puk obsažen v chytači, nesmí vzdálenost v_l od přímky *l* překročit polovinu délky chytače, což je hodnota *h*. Při této podmínce by ale byl určen za přítomný v chytači i puk, který by měl střed v bodě X_2. Proto je třeba ještě porovnávat vzdálenost bodu od přímky, která bude na přímku *l* kolmá v bodě S. Ta je na obrázku 19 označena jako *k* a její rovnice má prohozené koeficienty a znaménko:

$$r_x \cdot x + r_y \cdot y + q = 0\,. \tag{22}$$

Stejným způsobem jako u předchozí přímky lze spočítat posunutí *q*:

$$q = -r_x \cdot x_S - r_y \cdot y_S\,. \tag{23}$$

Výslednou rovnici přímky *k* lze opět dosadit do vzorce 20:

$$r_x \cdot x + r_y \cdot y - r_x \cdot x_S - r_y \cdot y_S = 0$$

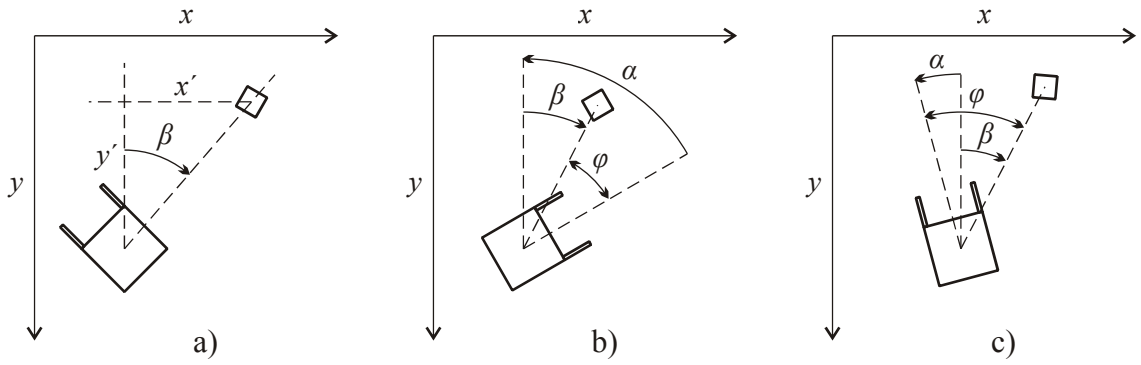
$$v_k = \frac{\left|r_x \cdot x + r_y \cdot y - r_x \cdot x_S - r_y \cdot y_S\right|}{\sqrt{r_x{}^2 + r_y{}^2}}\,. \tag{24}$$

Vzdálenost v_k nesmí překročit polovinu rozteče chytače *w*. Tím je dosaženo, že se detekují pouze body uvnitř libovolně natočeného chytače, čili na obrázku 19 pouze bod X_1. Změnou parametru *h* lze detekovat i puky před chytačem. Výsledná podmínka pro přítomnost puku se středem [*x*, *y*] v chytači je tedy:

$$\frac{\left|r_y \cdot x - r_x \cdot y - r_y \cdot x_S + r_x \cdot y_S\right|}{\sqrt{r_x{}^2 + r_y{}^2}} < h \quad \wedge \quad \frac{\left|r_x \cdot x + r_y \cdot y - r_x \cdot x_S - r_y \cdot y_S\right|}{\sqrt{r_x{}^2 + r_y{}^2}} < w\,. \tag{25}$$

Tato rovnice je aplikována na každý ze 49 puků. Počet kladných výsledků odpovídá počtu puků v chytači.

Snímače koncentrace puků v pěti zónách okolo robotu jsou realizovány pomocí výpočtu úhlu φ, který svírá směrnice natočení robotu a spojnice robotu s dílčím pukem. Tento úhel se počítá pro každý ze 49 puků. Podle úhlu se puk zařadí do příslušného intervalu (zóny). Úhel φ se počítá z úhlu natočení robotu (α) a úhlu β, který vůči souřadnicovému systému svírá spojnice robotu a puku. Ukázkové situace jsou vyobrazeny na obrázku 20 b) a c).



Obrázek 20 – Princip výpočtu úhlu φ, ve kterém bude zaznamenán puk

V RDS má každý objekt své 3D natočení uložené jako kombinaci osy natočení a úhlu natočení podél ní. Úhel natočení Φ objektu je vždy kladný. Natáčí-li se objekt kolem svislé osy, pak má osa natočení souřadnice buď [0, 1, 0] nebo [0, -1, 0]. Jestliže má osa natočení kladnou souřadnici, hodnota úhlu roste při otáčení v kladném směru (viz obrázek 21c), a když je souřadnice záporná, roste hodnota úhlu při otočení v záporném směru, čili po směru hodinových ručiček. Pokud se objekt otáčí plynule kolem své osy a otočí se o 360°, změní se osa otáčení z kladné na zápornou a úhel klesá ze 2π zpět k nule. Při první otočce kolem své osy tedy úhel roste a osa je kladná a při druhé úhel klesá a osa je záporná. Při třetí otáčce by se osa opět stala kladnou a úhel by rostl. Aby při libovolné otáčce šlo zjistit jednoduše otočení objektu v souladu s obrázkem 21c, byl použit přepočet pomocí vzorce

$$\alpha = \begin{cases} \phi, & y \geq 0 \\ 2\pi - \phi, & y < 0 \end{cases}, \tag{26}$$

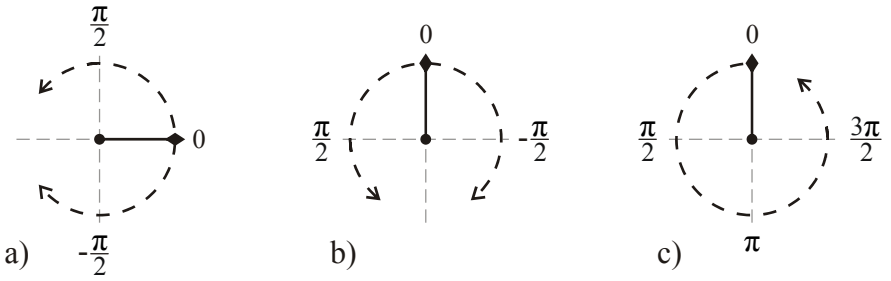
kde y je druhá (svislá) souřadnice osy otočení objektu, Φ je nepřepočtený úhel natočení objektu a α je výsledný přepočtený úhel natočení.

Pro výpočet úhlu β, který svírá spojnice středu robotu a středu puku vůči souřadnicové ose y, je vhodné použít arcus tangens. Z obrázku 20a je patrné, že

$$\begin{aligned} \operatorname{tg} \beta &= \frac{x'}{y'} \\ \operatorname{arctg} \frac{x'}{y'} &= \beta \end{aligned} \quad . \tag{27}$$

Arcus tangens je však definován pouze od -π/2 do π/2. Nabízí se použití vylepšené funkce *Atan2(y,x)*, která umožňuje vypočítat arcus tangens ze dvou parametrů a ne pouze jejich poměru. Tím lze pokrýt rozsah od -π do π. *Atan2* však počítá úhly dle obrázku 21a, zatímco pro výpočet náležitosti puku do zóny je potřeba, aby byly úhly podle obrázku 21b. Jelikož má souřadnice y opačné znaménko, než je běžné, stačí pouze prohodit souřadnice x a y, dát jim záporná znaménka a vznikne tak arcus tangens schopný spočítat úhel, který svírá spojnice robotu a puku vůči souřadnicové ose:

```
Beta = Math.Atan2(Robot.X - Puk.X, Robot.Y - Puk.Y);
```



Obrázek 21 – Hodnoty úhlu natočení pro tangens (a), spojnici (b) a robot (c)

Pro výpočet výsledného úhlu φ, čili příslušnosti do jedné ze zón, byl zvolen požadavek na úhel definovaný dle obrázku 21b, čili stejně jako úhel β. Jak správně vypočítat úhel z úhlů α a β, nejlépe ukazuje obrázek 20c. Na něm má úhel α kladnou hodnotu a úhel β zápornou. Úhel φ je v tomto případě větší než oba úhly. Aby to tak vyšlo, musí se odečíst záporný úhel od kladného. Z obrázku 21b je ale známo, že pokud úhel směřuje doprava, pak je jeho hodnota záporná. Proto platí, že

$$\varphi = \beta - \alpha \,. \tag{28}$$

Pokud by se však tento vzorec aplikoval na případ ilustrovaný na obrázku 20b, vyšel by úhel také záporný, protože hodnota α není nikdy záporná, ale vždy kladná. Pro tento případ by tedy vyšel úhel menší než -π, což nesplňuje požadavek na rozsah úhlu. Aby se tak nestávalo, úhlům nižším než tato hranice se připočte 2π. Vyjde tak konečný vztah pro výpočet úhlu, ve kterém robot „vidí“ puk:

$$\varphi = \begin{cases} \beta - \alpha, & \beta - \alpha \geq -\pi \\ \beta - \alpha + 2\pi, & \beta - \alpha < -\pi \end{cases}. \tag{29}$$

Posledním krokem je rozhodnutí, do které zóny puk patří. Úhlové rozsahy pěti zón (z_0, z_1, z_2, z_3, z_4) naznačuje rovnice

$$-\frac{5\pi}{8} \leq z_0 < -\frac{3\pi}{8} \leq z_1 < -\frac{\pi}{8} \leq z_2 \leq \frac{\pi}{8} < z_3 \leq \frac{3\pi}{8} < z_4 \leq \frac{5\pi}{8}. \tag{30}$$

Jestliže je tedy úhel φ například nulový, čili puk je přímo před robotem, pak je tento puk přičten do zóny 2. Když se takto zařadí všechny puky, lze rozhodovat, ve které zóně jich je nejvíce a ve které nejméně. Může se také stát, že úhel φ nebude náležet do rozsahu žádné zóny (například když bude puk za robotem). V takovém případě robot příslušný puk „nevidí“, a není proto registrován do žádné zóny. Součet puků ve všech zónách tedy nemusí být roven 49.

Posledním typem snímače je snímač překážek. Pro zjednodušení je při opakování experimentu realizován zvlášť snímač blízkosti zdi a snímač možné kolize mezi roboty. Snímač blízkosti zdi funguje pouze na principu porovnání úhlu natočení robotu a jeho vzdálenosti od zdi. Jakmile vzdálenost od zdi klesne pod 0,9 metrů, začne se porovnávat úhel, který robot se zdí svírá.

Jestliže se označí úhel, kdy robot míří přímo proti zdi, jako nulový, pak při úhlech nula až pí čtvrtin na jakoukoliv stranu robot zastaví a otáčí se směrem od zdi. Míří-li tedy robot ke zdi a je natočený mírně doprava, pak při přiblížení pod 0,9 metrů zastaví

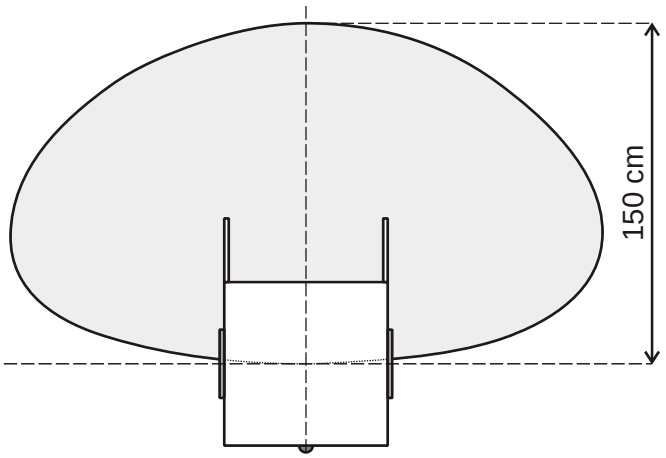
a začne se otáčet doprava. Když překročí úhel pí čtvrtin, mění se pravidlo na druhé, které říká, že když je úhel od pí čtvrtin do pí polovin (na obou stranách), pak je robotu umožněn původní směr jízdy, ale přidá se mu mírné natočení směrem od zdi. Robot už tedy nestojí, ale stále se snaží od zdi odvrátit.

Problematické jsou ovšem rohy. Aby v nich robot nekmital od jedné zdi ke druhé, mají přednost pravidla příslušející zdem podél osy *y* (ve 3D prostředí značené *z*). K tomu je ještě v rozích upraveno mírnější pravidlo (tj. otáčení za jízdy) na přísné (tj. otáčení bez jízdy), aby se robot při jízdě do rohu neotočil jen málo a nenarazil do zdi podél osy *x*. Jestliže robot ke zdi couvá, po poklesnutí vzdálenosti pod 0,9 se desetkrát sníží jeho rychlost. Robot couvá vždy jenom chvilku, a proto je tímto dosaženo, že do zdi nestihne narazit.

Snímače kolizí robotů jsou implementovány jako porovnání souřadnic dvou robotů. Robot, který zjišťuje, zda by mohl narazit do druhého, si zjistí vzdálenost *l*, v jaké se nachází střed druhého robotu vůči jeho vlastnímu středu. Dále se pomocí stejné metody, jaká je použita pro zjištění, pod jakým úhlem vidí robot puk, zjistí, pod jakým úhlem vidí robot druhý robot. Označí-li se tento úhel β, pak lze použít rovnici

$$l < 1{,}5 - 1{,}5 \cdot \left(\frac{2 \cdot \beta}{\pi} \right)^{10} \tag{31}$$

pro zjištění, zda se robot nachází v kritické zóně. Tvar této zóny včetně schematického obrázku robotu pro lepší představu je naznačen na obrázku 22.



Obrázek 22 – Dosah snímače zamezujícího kolizi s ostatními roboty (kritická zóna)

Jestliže se v této zóně objeví jiný robot, pak nastává úhybný manévr. Jestliže je úhel β kladný, pak robot zastaví a za mírného couvání se otáčí doprava, dokud se druhý robot nedostane z kritické zóny. Pokud je úhel β záporný, pak se druhý robot nachází vpravo, a tím pádem se za mírného couvání zatáčí doleva. Tato dvě pravidla ve většině případů zamezují kolizím při pohybu vpřed.

Při couvání (to nastane vždy po umístění puku ke shluku) by tento způsob vyhýbání nebyl vyhovující. Robot by rozhrabal právě vytvořený shluk puků. Proto při couvání snímá ve stejné zóně (ale za sebou) a když se v ní objeví jiný robot, pouze desetkrát

zpomalí svou plánovanou rychlost. Mezitím může druhý robot odjet z místa, kde překážel a první robot může bezpečně odjet.

Těmito všemi zpětnými vazbami od snímačů je definováno, jak se bude robot hýbat. Nejprve zjistí, zda má puk a podle toho buď míří k nejhustšímu (pokud puk má) rozložení puků nebo nejřidšímu (pokud puk nemá). Jakmile puk přiveze ke shluku, změní se chování na dočasné couvání (po dobu jedné sekundy zacouvá a otočí se náhodným směrem). Jestliže se ale přiblíží ke zdi, změní se směr pohybu podle pravidel vyhýbání zdem. A jako poslední se kontroluje případná kolize s jiným robotem. Ta se vyhodnocuje jako poslední a má nejvyšší prioritu.

Posledním krokem v pohybu robotů je zjištění pohyblivosti. Pokud robot za posledních 10 sekund neujel alespoň 2 metry, pak je pravděpodobně v nějaké patové situaci (například dva roboty usilující v rohu o stejný puk). V tomto případě se robot zachová stejně jako v případě umístění puku ke shluku. Zacouvá a natočí se náhodným směrem. Tím buď dočasně uvolní cestu druhému robotu, nebo zahlédne jiný puk a pojede raději k němu.

Aby bylo možno experiment s puky vyhodnotit, jsou každou minutu uloženy souřadnice puků do textového souboru (ve tvaru *x;y;x;y;x;y*...). Ten pak slouží pro zpracování v programu MATLAB. Cílem je určit, které puky tvoří shluk. Tím lze určit jejich počet a velikost největšího.

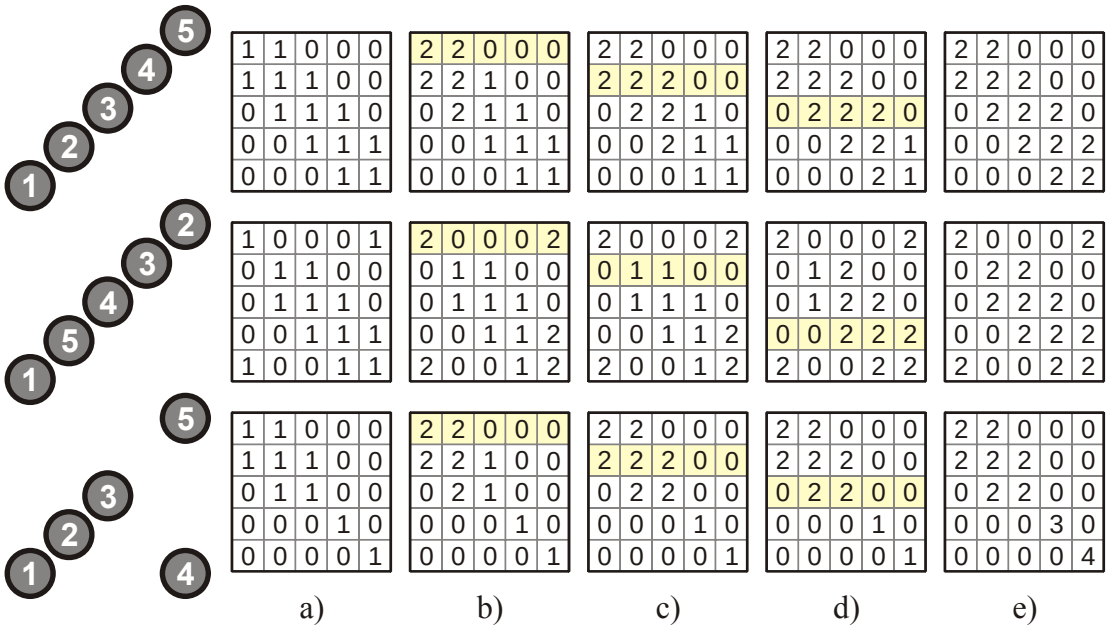
Algoritmus v MATLABu funguje tak, že nejprve načte ze souboru souřadnice všech 49 puků pro veškeré časové kroky. Nyní bude ukázán postup pro určení shluků v jednom časovém kroku. Nejprve se vytvoří binární matice 49×49 pomocí rovnice

$$\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 < 0{,}6^2 \qquad i = 1...49,\ j = 1...49\,, \tag{32}$$

kde x_i a y_i jsou souřadnice i-tého puku a x_j a y_j jsou souřadnice j-tého puku. Tato matice reprezentuje sousednost těchto dvou puků. Čili pokud jsou středy puků blíž než 0,6 metru (dvojnásobek průměru), sousedí puky spolu. Každý puk podle této matice sousedí minimálně sám se sebou, čili na diagonále jsou samé jedničky. Jestliže je tedy puk osamocený, má 1 pouze sám na sobě (na diagonále). Ukázkové příklady této matice jsou na obrázku 23a. Jak je vidět, když je pět puků vedle sebe, krajní puky přímo nesousedí. Díky prostředním ale patří do stejného shluku.

Proto prochází algoritmus matici a označuje skupiny číslem, přičemž první skupina má číslo 2. Algoritmus tedy začne prvním řádkem, všechny jedničky přeznačí za dvojky a tímto způsobem přeznačí i sloupce, ve kterých se na prvním řádku jednička nacházela. Jak vypadají matice po této úpravě, je naznačeno na obrázku 23b.

V dalším kroku projede algoritmus další řádek a zjistí přítomnost čísla 2. V takovém případě udělá to samé, co v předchozím kroku. Přečísluje všechny 1 na 2 a v příslušných sloupcích udělá totéž. Pokud dvojku nenalezne, nic se v matici nemění a algoritmus pokračuje na další řádek. Případy změn v tomto kroku jsou na obrázku 23c nahoře a dole. Prostřední matice ukazuje případ, kdy algoritmus nalezl jen jedničky, a tak nic nezměnil.



Obrázek 23 – Ukázkové polohy 5 puků a příslušné matice v různých krocích úprav

Tímto způsobem se projede celá tabulka. Vždy se pokročí na další řádek, zjistí se přítomnost čísla 2 a pokud je přítomno, přečíslovávají se jedničky a příslušné sloupce. Při horní variantě seskupení puků na obrázku 23 se takto označí všechny původní jedničky na dvojky, a tím je třídění hotovo. Výsledná matice je na obrázku 23e.

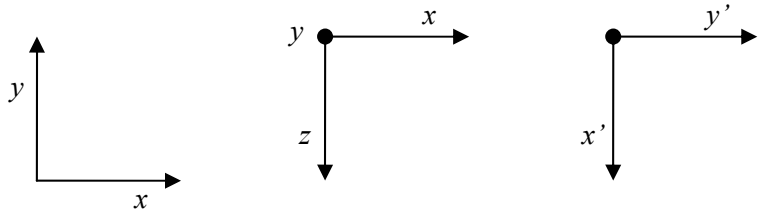
Pro prostřední případ však po projití celé matice zůstanou dvě jedničky (viz obrázek 23d uprostřed). Proto je potřeba projít tímto postupem matici znovu od prvního řádku. Jelikož nemusí být algoritmus rychlý, prochází matici více než dvakrát (maximálně 49×). Výsledný tvar matice je na obrázku 23e uprostřed.

Tímto způsobem byla ale určena pouze jedna skupina. Spodní ukázkový případ demonstruje, co se stane, když jsou puky rozděleny na více skupin. První tři puky byly při třetím kroku (viz obrázek 23d dole) určeny jako skupina. Další dva by však i po několikátém opakování postupu zůstávaly rovny jedné. Proto se musí změnit číslo 2 na číslo tři a projít matice znovu. Najde se první řádek, kde je na diagonále jednička, přečísluje se na trojky (včetně konkrétních sloupců) a projíždí se zbylé řádky, zda jsou na nich obtisknuté trojky. Pokud se tak několikrát nestane, zvýší se číslo skupiny na 4, opět se najde první řádek s jedničkou na diagonále, přečísluje se, atd. Výsledné rozdělení je na obrázku 23e dole.

Takto vyhodnocená data se již dají jednoduše zpracovat. Počet dvojek na diagonále matice se rovná počtu puků ve skupině „2“ a nejvyšší číselné označení skupiny je o jedno vyšší než celkový počet skupin (čili když je dole na obrázku 23 nejvyšší číslo skupiny 4, tvoří puky 3 shluky). Když se takto zpracují pozice puků ze všech časů, lze vykreslit požadované průběhy, tj. časový průběh počtu shluků a časový průběh počtu puků v největším shluku. Toto vyhodnocení bude v kapitole 5.

4.3 Realizace experimentu s vyhledáváním

Původní experiment probíhal v neurčených jednotkách. Při opakování byly zvoleny metry. Vznikl tak tedy čtvercový prostor 150×150 metrů, ohraničený zdmi. Aby byly souřadnice z experimentu v RDS stejné jako v původním, byla označena osa *x* jako *y'* a osa *z* jako osa *x'*. Jak je vidět na obrázku 24, nové souřadnice (c) přeznačené ze souřadnicového systému v RDS (b) jsou nyní pouze o 90° otočené oproti souřadnicím v původním experimentu (a). V dalším textu budou souřadnice uváděny podle přepočtených. Čili pokud budou souřadnice [10; 20], pak v RDS se jedná o [20; 0; 10].



Obrázek 24 – Souřadnicový systém v experimentu (a), RDS (b) a po přeznačení (c)

Stejně jako v původním experimentu se do oblasti 50×50 metrů umístí náhodně deset robotů s náhodným natočením. Aby byly roboty dobře rozlišitelné, má každý jinou barvu. Ta se vypočítá pomocí vzorce

$$\begin{aligned} \mathrm{R} &= i \bmod 2 \\ \mathrm{G} &= \left(\left(i \operatorname{div} 2\right) \bmod 3\right) \cdot 0{,}5\,, \\ \mathrm{B} &= \left(i \operatorname{div} 6\right) \bmod 2 \end{aligned} \tag{33}$$

kde *i* je pořadové číslo robotu, div znamená celočíselné dělení a výsledkem mod je zbytek po celočíselném dělení. Pro lepší představu výsledných RGB kombinací a příslušných barev jsou v tabulce 4 všechny možné kombinace výsledků.

Tabulka 4 – Přehled výsledků z výpočtu barev robotů

Číslo	**0**	**1**	**2**	**3**	**4**	**5**
RGB	0; 0; 0	1; 0; 0	0; 0,5; 0	1; 0,5; 0	0; 1; 0	1; 1; 0
Barva						
Číslo	**6**	**7**	**8**	**9**	**10**	**11**
RGB	0; 0; 1	1; 0; 1	0; 0,5; 1	1; 0,5; 1	0; 1; 1	1; 1; 1
Barva						

Jelikož je v simulaci robotů pouze deset a rozsah barev je pro dvanáct, počítají se roboty od čísla 1. V RDS jsou roboty indexovány od nuly, a proto je místo *i* do vzorců dosazeno *i* + 1. V MATLABu jsou pole implicitně indexována od jedničky, a tak při zobrazování výsledků simulace není třeba provádět žádné změny ve výpočtu barev. Z toho je zjevné, že mezi výslednými barvami nebude černá a bílá.

Každý robot má diferenciální podvozek o dvou kolech (průměr 0,5 m) se zadním kolečkem (10 cm) pro udržování rovnováhy s minimem tření. Roboty mají půdorys tvaru čtverce o straně dlouhé 1 metr. Stejné je i rozpětí kol. Narozdíl od experimentu s puky nemá robot žádné přidané součásti. Jeho cílem je pouze jet.

Snímač signálu, kterým každý z robotů disponuje, funguje pouze na principu výpočtu ze známé vzdálenosti vysílače. Čili podle rovnice 10. Dosazovaná vzdálenost je počítána ze známých souřadnic jako

$$d = \sqrt{(x_R - x_T)^2 + (y_R - y_T)^2 + (z_R - z_T)^2}\,, \tag{34}$$

kde R značí souřadnice robotu a T souřadnice vyhledávaného cíle. Ten se dle zadání nachází na [130; 130], čili v RDS na [130; 0; 130]. Rozsah Gaussovského šumu byl zvolen od 0 do 2.

Pohyb robotů se řídí dvěma způsoby. Nejprve jedou spirálovitě a poté dle PSO. Při obou pohybech jsou průběžně vypočítávány mezicíle, ke kterým má robot směřovat. Při pohybu po spirále jsou cíle generovány s desetistupňovým rozestupem. Jakmile se robot přiblíží k mezicíli, vypočítá se další bod o 10° spirály dál. Při pohybu dle PSO jsou mezicíle průběžně posouvány každou pětinu sekundy.

Aby robot k mezicíli mohl správně dojet, zjišťuje, jaký úhel vůči němu svírá. K tomu slouží stejný vzoreček (29), jako při výpočtu úhlu, který svírá robot a puk. Místo výpočtu zón je zde ale úhel φ násoben konstantou, a je tak dosaženo plynulého zatáčení až k mezicíli. Aby byla rychlost robotu rovna maximální povolené rychlosti 2 m/s, je třeba nastavit střední hodnotu požadovaných otáček obou kol rovnu 0,4. Toho je dosaženo při použití vzorce

$$P_L = 0{,}4 - 0{,}6 \cdot \frac{\varphi}{\pi} \qquad P_R = 0{,}4 + 0{,}6 \cdot \frac{\varphi}{\pi}\,, \tag{35}$$

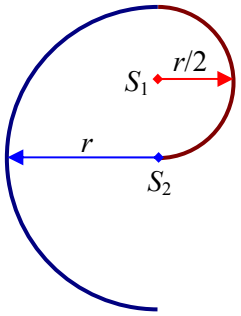
kde P_L jsou požadované otáčky levého kola, P_R pravého kola a φ je úhel ze vzorečku 29, udávaný v radiánech. Jestliže je φ kladné (viz obrázek 21b), pak je cíl více vlevo než kam robot míří. Musí se potom adekvátně snížit rychlost levého kola a zvýšit rychlost pravého, aby robot zatáčel doleva. Tyto výpočty jsou společné pro oba typy pohybů. Souřadnice mezicílů se však pro každý způsob pohybu počítají jinak.

Prvním způsobem je pohyb po spirále. Ta je tvořena půlkružnicemi se vzrůstajícím poloměrem a kmitajícím středem. Aby byla splněna rozteč r podle obrázku 13, musí mít první půlkružnice střed posunutý od robotu o $r/2$ a poloměr roven také $r/2$. Druhá, navazující půlkružnice má průměr zvětšený o r, čili poloměr o $r/2$. Její střed vůči robotu posunutý není. První dvě půlkružnice včetně středů a poloměrů jsou vyobrazeny na obrázku 25. Třetí půlkružnici bude opět narůstat poloměr o $r/2$ a střed bude opět v $r/2$.

Výpočet souřadnic bodů kružnice popisuje vzorec

$$\begin{aligned} x &= x_s + \frac{1+q}{2} \cdot r \cdot \sin\frac{\alpha \cdot \pi}{180} \\ y &= y_s + \frac{1-q \bmod 2}{2} \cdot r - \frac{1+q}{2} \cdot r \cdot \cos\frac{\alpha \cdot \pi}{180} \end{aligned}\,, \tag{36}$$

kde x_S a y_S jsou souřadnice startovní pozice robotu (počátek spirály, na obrázku 25 bod S_2), r je rozteč spirály (totožný s dosahem signálu), α je požadovaný dosahovaný úhel ve stupních a q je číslo symbolizující současnou půlkružnici. To se spočítá jako celočíselně vydělený úhel sto osmdesáti (α div 180). První půlkružnice tedy bude mít q rovno 0, druhá 1, atd. Adekvátně tomu pak q mod 2 bude nabývat hodnot nula nebo jedna, podle toho, zda se jedná o sudou nebo lichou půlkružnici.



Obrázek 25 – První dvě půlkružnice spirály a jejich středy a poloměry

Takto popsaný pohyb však neodpovídá pohybu, který byl prezentován v publikovaném výsledku. Po změření velikosti čtvrtkružnice v něm je patrno, že se nejedná o symetrickou spirálu a už vůbec ne s poloměrem $r/2$. Tato deformovaná spirála má na výšku (podél osy y) poloměr první půlkružnice 70 metrů a na šířku (podél osy x) pouhých 40 metrů. Jelikož původní spirála měla v obou směrech poloměr r/2, stačí v rovnici 36 změnit průměr r na příslušná čísla:

$$\begin{aligned} x &= x_s + \frac{1+q}{2} \cdot 80 \cdot \sin\frac{\alpha \cdot \pi}{180} \\ y &= y_s + 140 \cdot \frac{1-q \bmod 2}{2} - 140 \cdot \frac{1+q}{2} \cdot \cos\frac{\alpha \cdot \pi}{180} \end{aligned}, \tag{37}$$

Mimo tuto deformovanou spirálu se ještě nabízí jeden způsob, jak se spirálovitým pohybem naložit. Jedná se o spirálu s posunutým středem. K němu roboty nejprve ujedou určitý kus po přímce a od něj pak pokračují po nedeformované spirále [12]. Výpočet souřadnic takové trasy popisuje rovnice

$$\begin{aligned} x &= x_s + x_p + \frac{1+q}{2} \cdot r \cdot \sin\frac{\alpha \cdot \pi}{180} \\ y &= y_s + y_p + \frac{1-q \bmod 2}{2} \cdot r - \frac{1+q}{2} \cdot r \cdot \cos\frac{\alpha \cdot \pi}{180} \end{aligned}, \tag{38}$$

kde x_p je posunutí středu podél osy x a y_p posunutí podél osy y. Takovýto pohyb umožňuje projet spirálou větší prostor, protože roboty nebudou omezeny zdmi, které jsou blízko původní počáteční oblasti.

Po ukončení spirálovitého pohybu se roboty řídí pohybem podle PSO. Narozdíl od běžných simulačních prostředí v RDS nelze určit robotu přímo vektor rychlosti. Proto není třeba využívat rovnice se sníženým časovým krokem, ale pouze původní.

A namísto posunutí robotu podle jeho vypočtené rychlosti se posouvá jeho mezicíl. Při implementaci byly použity rovnice

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_1 \left(x_{ij}^{\#}(t) - x_{Pij}(t)\right) + c_2 r_2 \left(x_j^{*}(t) - x_{Pij}(t)\right), \tag{39}$$

$$x_{Tij}(t+1) = x_{ij}(t) + v_{\max} \frac{v_{ij}(t+1)}{\left|v_{ij}(t+1)\right|}, \tag{40}$$

kde w je koeficient setrvačnosti (experimentálně stanoven na 10) a v_{ij} je současná rychlost, která se považuje za konstantní (2 m/s), pouze se přepočítá do složek ve směrech x a y podle aktuálního úhlu natočení robotu. Poloha x_{Pij} je současná poloha robotu a poloha x_{Tij} je poloha vypočteného mezicíle. Maximální rychlost $v_{\max}$ je dána zadáním a je rovna dvěma metrům za sekundu. Ostatní parametry odpovídají původní teorii. Koeficienty c_1 a c_2 byly zvoleny rovny dvěma.

Nejlepší pozice se počítá dle nejvyšší dosažené síly přijímaného signálu od cíle. Jakmile robot přijede k pozici cíle (blíž než 2 metry od něj), zastaví se mu motory a jeho cesta k cíli končí.

Každé tři sekundy se do textového souboru zaznamenají pozice všech 10 robotů. Tento textový soubor slouží jako výstup ze simulace, který je dále zpracováván v MATLABu. Tam se pouze načtou souřadnice a vykreslí se z nich trasy robotů, jejich původní pozice a poloha cíle.

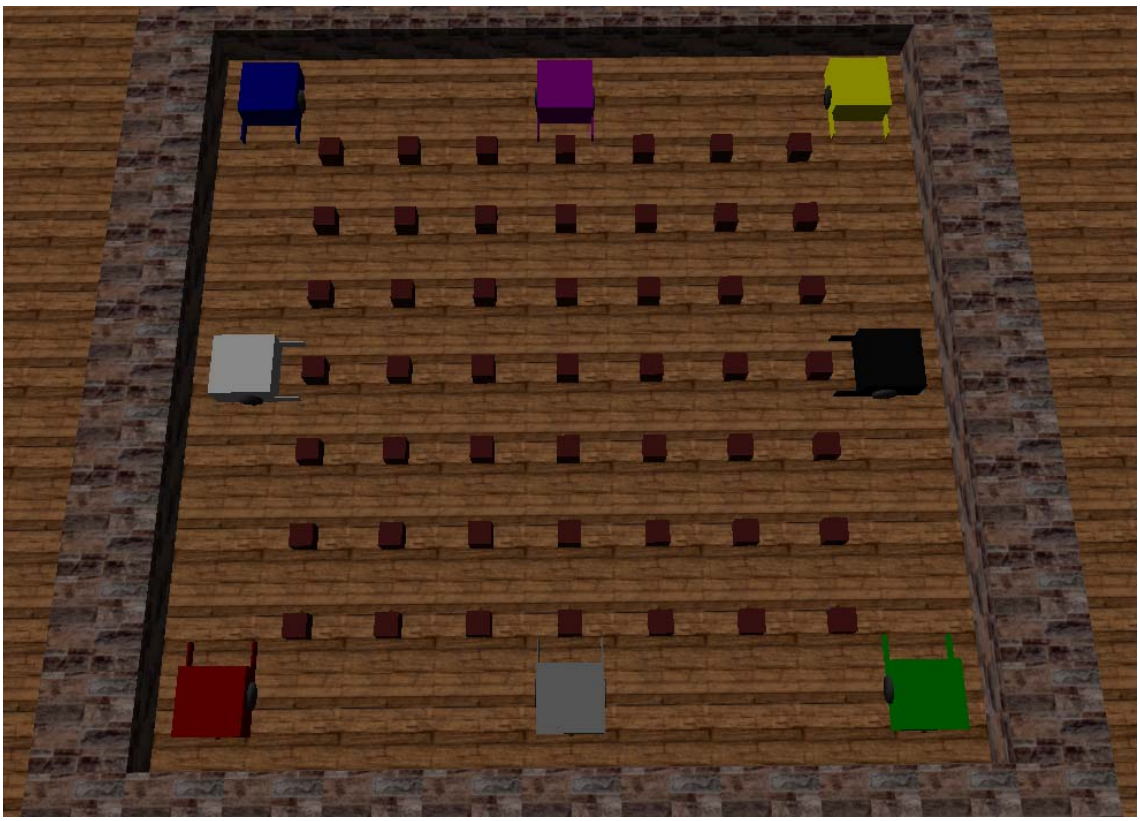
Aby se dal výsledek ze simulace dobře srovnat s publikovaným, byly na grafu v původním experimentu naměřeny startovní polohy robotů. V realizaci v RDS lze zapnout jejich použití namísto náhodného rozmístění. Tyto polohy jsou: [5; 26], [8; 34], [19; 45], [22; 10], [31; 50], [35; 27], [36; 4], [41; 11], [41; 32], [50; 22].

5 VYHODNOCENÍ EXPERIMENTU

Tato kapitola pojednává o výsledcích experimentů a porovnání s původními. Dále bude zhodnoceno Microsoft Robotics Developer Studio a jeho vhodnost pro další používání.

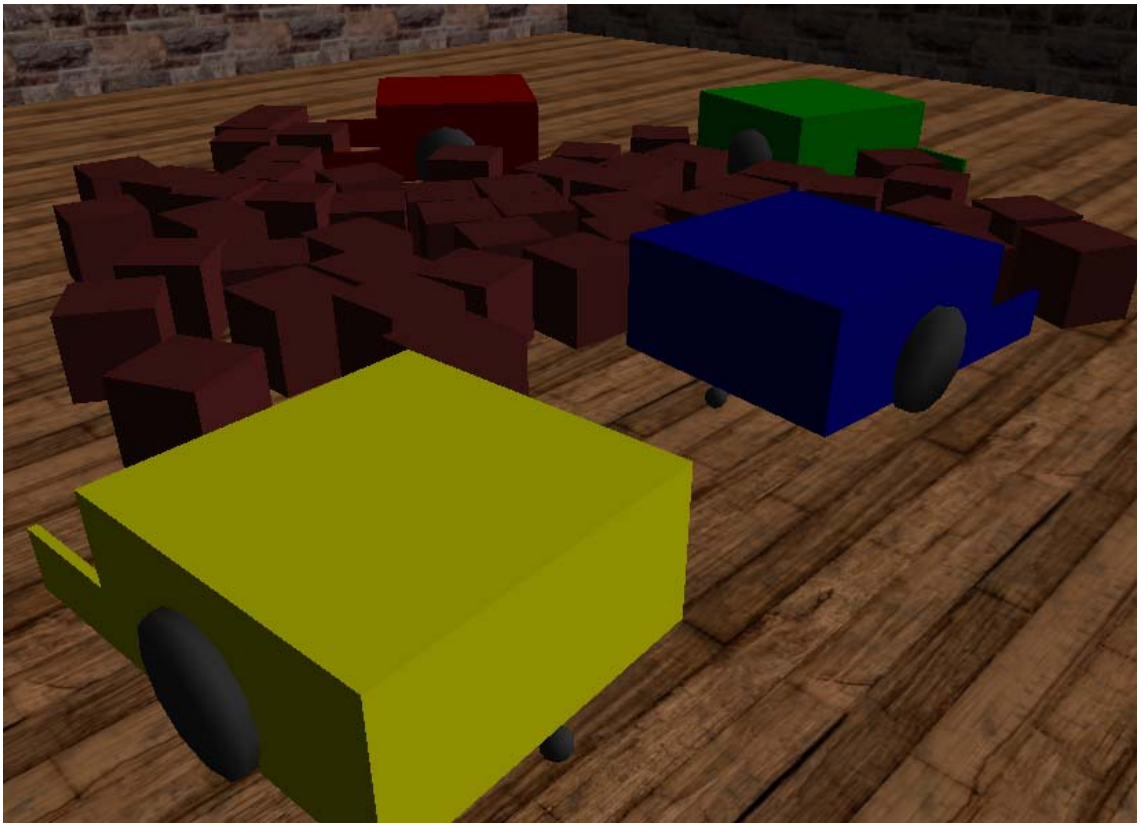
5.1 Vyhodnocení experimentu s puky

Původní experiment se shromažďováním puků byl prováděn pouze s jedním či dvěma roboty. Při opakování v RDS byl rozšířen na libovolný počet robotů. Z důvodu fyzického rozmístění robotů jich však bylo naprogramováno maximálně osm. Jak v RDS vypadá jejich počáteční natočení, rozmístění a barevné rozlišení, je ukázáno na obrázku 26. Nejvyšší počet robotů, s nimiž byl experiment vyhodnocován, je však šest. U vyšších počtů robotů docházelo již k příliš velkému množství kolizí a roboty nikdy nedosáhly požadovaného cíle, a to seskupit puky do jednoho shluku.



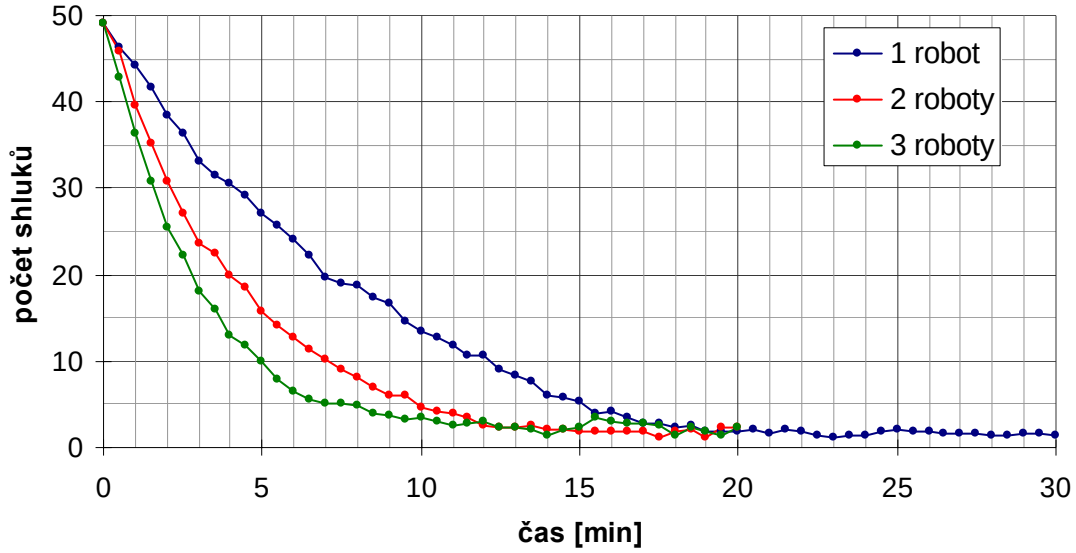
Obrázek 26 – Vzhled experimentu s puky a šesti roboty v prostředí RDS

Na obrázku 27 je zobrazen případ, kdy jsou již puky posbírány. Obrázek byl zaznamenán v případě se čtyřmi roboty a z jiného pohledu, než je základní, aby lépe vynikla trojrozměrnost VSE v RDS.

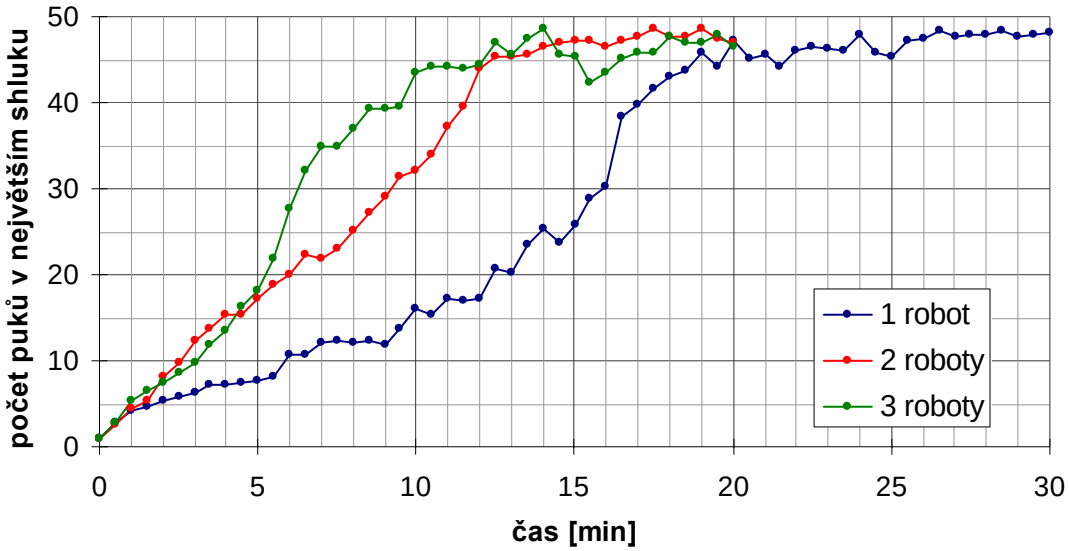


Obrázek 27 – Shluk puků shromážděných čtyřmi roboty

Jelikož nebylo v původní práci uvedeno žádné rychlostní omezení robotů, lze porovnávat pouze tvary průběhů výsledků experimentu. Na obrázku 28 jsou vykresleny časové průběhy počtu shluků pro 1–3 roboty. Každý z průběhů je průměr z 8 měření. Obrázek 29 zobrazuje průměrné průběhy počtu puků v největším shluku ze stejných měření a se stejnými počty robotů.



Obrázek 28 – Průměrné časové průběhy počtu shluků

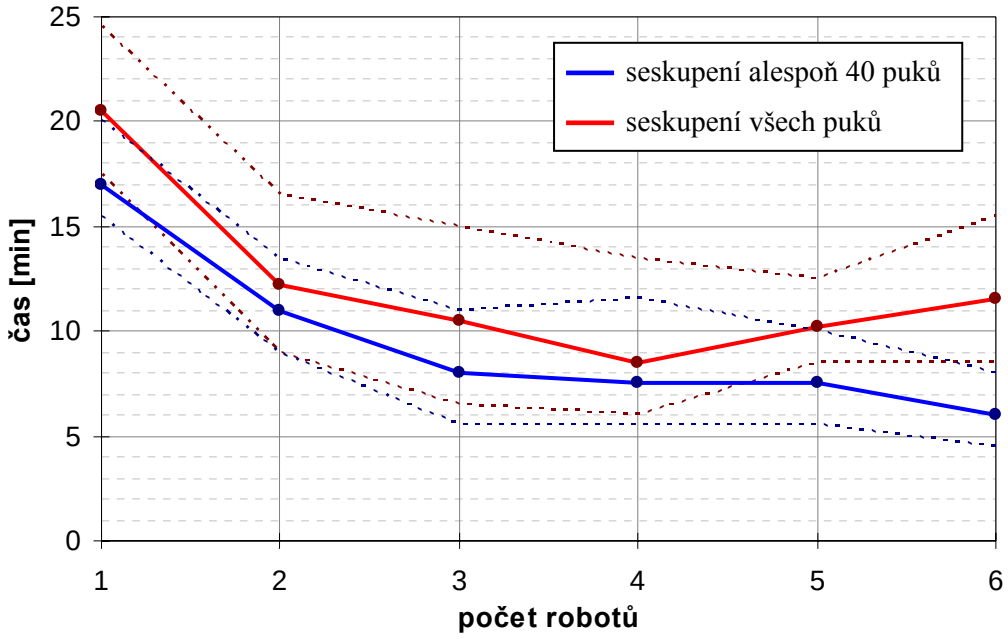


Obrázek 29 – Průměrné časové průběhy počtu puků v největším shluku

Srovnáním s původním experimentem lze zjistit, že průběhy mají podobný tvar, avšak roboty v RDS svůj úkol plní rychleji. Také stojí za povšimnutí, že oproti původnímu experimentu dochází přidáním dalších robotů k urychlení procesu. Bylo proto ověřeno, do jakého počtu robotů se jedná o zrychlení a kdy je již počet robotů natolik velký, že stráví příliš mnoho času vzájemným vyhýbáním se na úkor plnění úkolu. Pro detailnější analýzu byly stanoveny dva cíle, které se porovnávají. Prvním cílem je dosažení hranice alespoň 40 puků v největším shluku a druhým cílem je seskupení všech puků do jednoho shluku. Pro každý počet robotů bylo provedeno 8 měření a z nich určeny časy dosažení cíle. Z těchto časů byl následně vypočítán průměr, medián, maximum a minimum. Výsledky jsou zapsány v tabulce 5.

Tabulka 5 – Závislost rychlosti splnění dvou úkolů na počtu robotů

Úkol 1. – alespoň 40 puků v největším shluku						
Robotů	**1**	**2**	**3**	**4**	**5**	**6**
Průměr [min]	17	11	8	7,5	7,5	6
Medián [min]	16,5	11,25	7,75	6,5	6,75	5,5
Maximum [min]	20	13,5	11	11,5	10	8
Minimum [min]	15,5	9	5,5	5,5	5,5	4,5
Úkol 2. – všechny puky v jednom shluku						
Robotů	**1**	**2**	**3**	**4**	**5**	**6**
Průměr [min]	20,5	12,25	10,5	8,5	10,25	11,5
Medián [min]	19,75	11,75	9,75	7,5	10	11,5
Maximum [min]	24,5	16,5	15	13,5	12,5	15,5
Minimum [min]	17,5	9	6,5	6	8,5	8,5



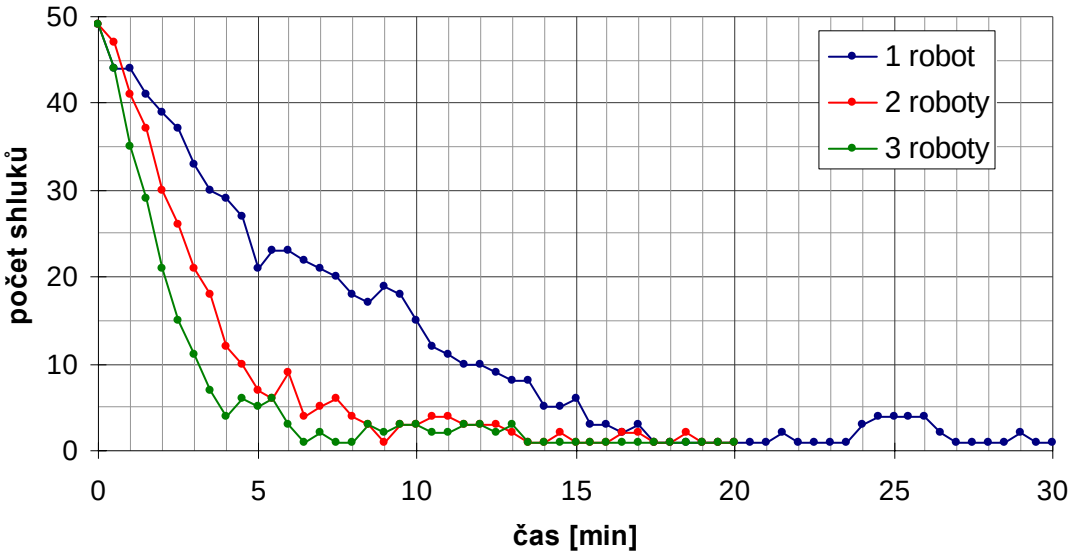
Obrázek 30 – Závislost rychlosti splnění úkolů na počtu robotů

Na obrázku 30 jsou vyobrazena data z tabulky 5. Jedná se sice o diskrétní hodnoty, ale pro lepší názornost byly v grafu propojeny čarami. Silné čáry určují průměrné hodnoty a slabé přerušované čáry jsou maxima a minima. Modře je zakresleno splnění úkolu seskupit alespoň 40 puků a červeně je zobrazeno seskupení všech puků do jednoho shluku.

Je vidět, že čím více robotů je do shromažďování zapojeno, tím rychleji se dosáhne seskupení čtyřiceti puků. O zbývající puky se však poté začnou roboty prát. Proto trvá seskupení všech puků od určitého počtu robotů déle než při počtu nižším. Tato hranice byla změřena u 4 robotů, kdy byl dosažen nejlepší průměr i medián výsledných časů při seskupování všech puků.

Při větším počtu robotů stráví jednotlivci více času vzájemným vyhýbáním než dovážením puků do shluku. Obzvlášť když jedou cíleně všechny roboty k místu s nejnižší koncentrací. Pak se setkají třeba tři roboty u jednoho puku a v další cestě jim brání zeď. Nakonec puk sice přesunou díky zabezpečení proti dlouhému stání na místě, ale i toto zdržení je značné.

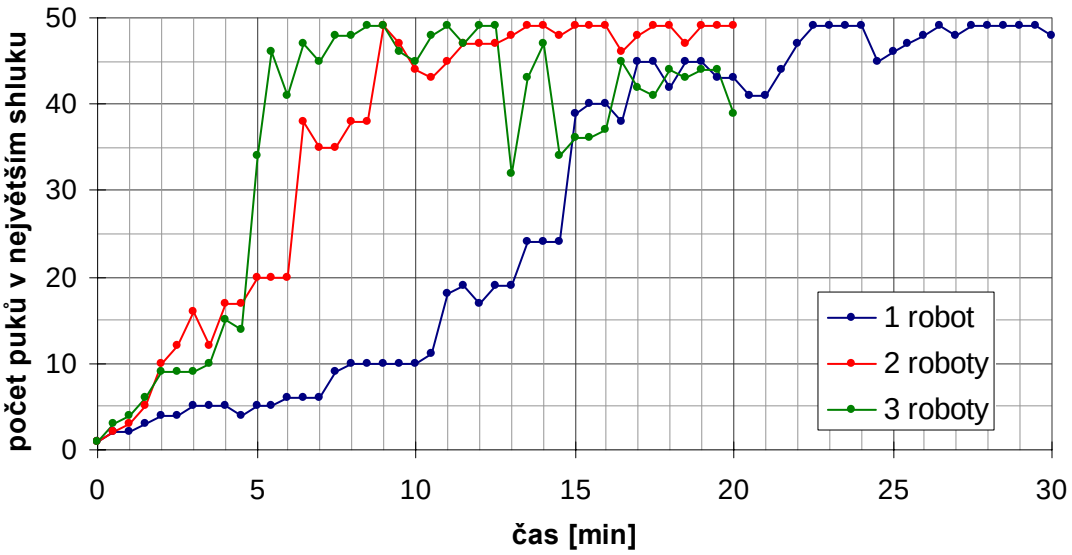
Pro lepší představu, jak probíhají jednotlivá shromažďování, jsou na obrázcích 31 a 32 zakresleny průběhy s nejlepšími výsledky. Jedná se vždy o verze s jedním, dvěma a třemi roboty. Na obrázku 31 jsou časové průběhy počtu shluků v případech, kdy došlo k nejrychlejšímu shromáždění všech puků do jednoho shluku. Na obrázku 32 jsou potom časové změny počtu puků v největším shluku, ale to pro případy, kdy bylo nejdříve dosaženo alespoň čtyřiceti puků v jednom shluku.



Obrázek 31 – Nejlepší časové průběhy počtu shluků

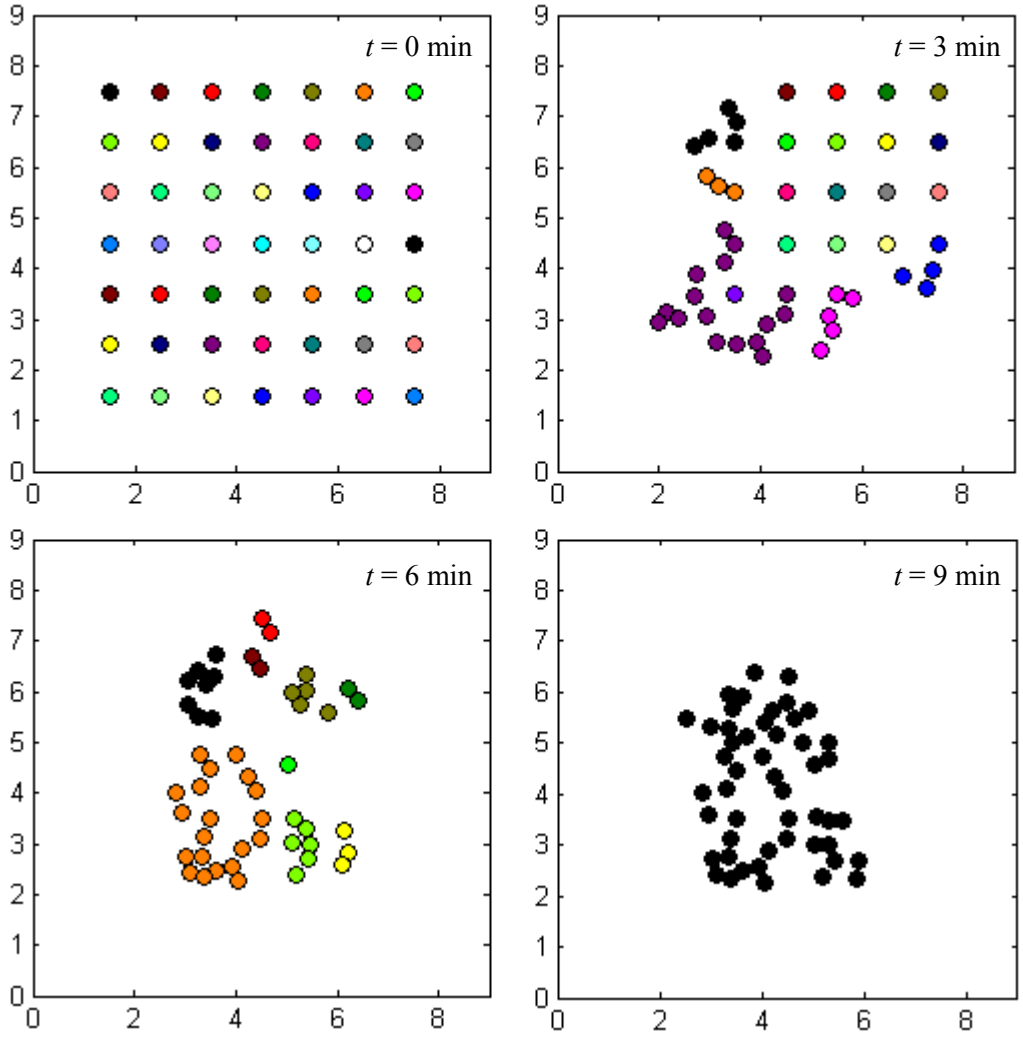
Na obrázku 31 na modrém průběhu je dobře vidět nedokonalost algoritmu po splnění úkolu. Robot totiž neví, že už jsou všechny puky seskupeny a snaží se je shromažďovat dál. Tím si ale rozhrne již hotový shluk. Například kolem času 25 minut je tento jev velmi dobře patrný. Robot puky oddělí a opět seskupí. A takto by mohl pracovat až do nekonečna. V realitě samozřejmě do vyčerpání zdroje energie.

Případ ještě většího rozhrnutí puků je vidět na obrázku 32 na zeleném průběhu ve 13. minutě. Jedná se o tři roboty, a tak je větší šance, že se shluk puků rozdělí. V největším shluku tak během půl minuty zůstane ze 49 puků pouhých 32. Do konce měření už pak v tomto případě roboty tento problém nestihly vyřešit. Avšak svůj úkol splnily už dávno předtím.



Obrázek 32 – Nejlepší časové průběhy počtu puků v největším shluku

Na závěr vyhodnocení experimentu je ještě dobré ukázat, jak vůbec vypadá průběh shromažďování vizuálně, a ne jenom číselně. To je vyobrazeno na obrázku 33, kde jsou rozložení puků v časech 0, 3, 6 a 9 minut. Jedná se shromažďování dvěma roboty, které má nejlepší výsledek pro oba úkoly, a to právě 9 minut. Detailnější průběh počtu shluků po půl minutách je červeně vykreslen na obrázku 31 a průběh počtu puků v největším shluku je červeně vykreslen na obrázku 32.



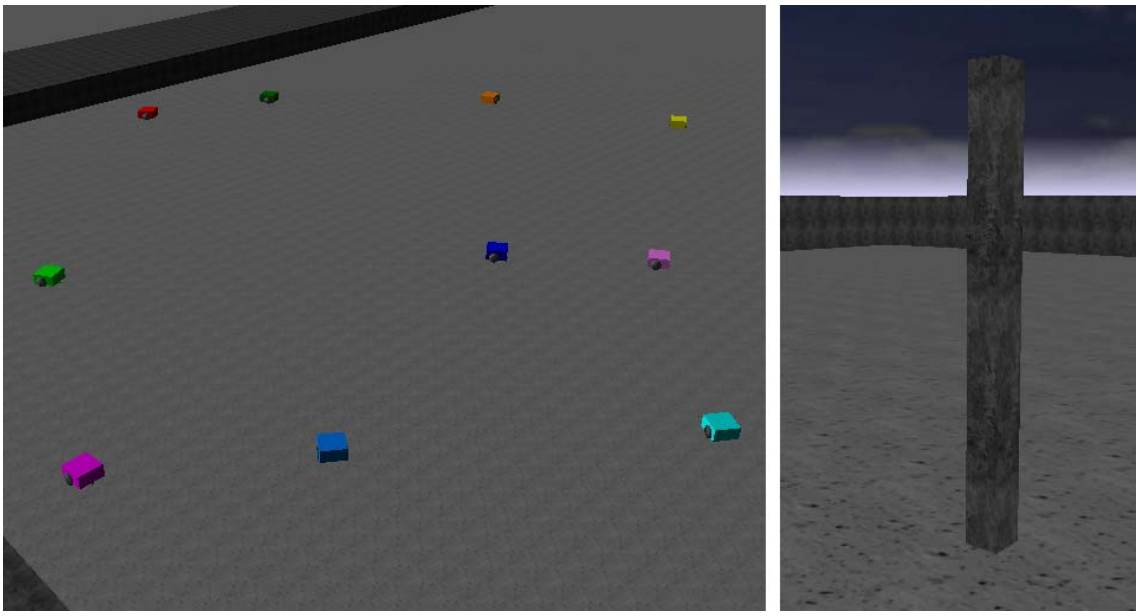
Obrázek 33 – Časové změny polohy puků včetně barevného rozlišení shluků

Na obrázku 33 jsou navíc barevně rozlišeny rozpoznané shluky. Například v čase 6 minut jsou puky rozděleny na 9 shluků, přičemž v tom největším (oranžovém) se nachází 20 puků. Skupina, která je algoritmem nalezena jako první, je označena černou barvou. Puky v čase 9 minut jsou všechny černé, protože společně tvoří jednu jedinou skupinu, čili detekovanou jako první.

Původní experiment byl tedy zopakován a bylo dosaženo podobného tvaru křivek. Dále byl rozšířen o vliv počtu robotů na rychlost plnění úkolů.

5.2 Vyhodnocení experimentu s vyhledáváním

Původní experiment vyhledávání dělníků uvězněných v dole byl prováděn s deseti roboty, které měly podle dokumentace nejprve jet po spirále o rozteči *R*, jež je totožná s dosahem signálu z RFID vysílače na helmách dělníků. Při opakování experimentu bylo nejprve použito také deset robotů a předepsaná rozteč spirály. Na obrázku 34 jsou vidět vlevo roboty rozmístěné na počáteční souřadnice a vpravo grafické zobrazení cíle – zjednodušeně pouze tyč na souřadnicích [130; 130]. Textura pro zdi a cílovou tyč byla vybrána z volitelné kolekce v RDS a textura pro zem byla přidána pomocí externího souboru.



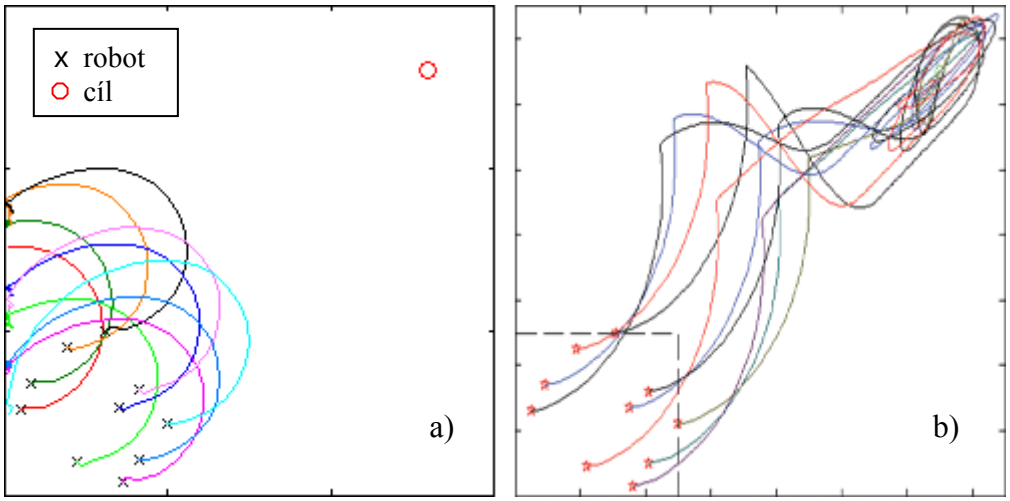
Obrázek 34 – Vzhled 10 robotů na počátku (vlevo) a vyznačení cíle (vpravo) v RDS

Obrázek 35a zobrazuje výslednou trajektorii robotů rozmístěných do míst, která byla změřena z původního výsledku, a pohybujících se podle předepsané spirály. Jak je vidět, roboty vyjedou s roztečí $R = 50$ m, udělají otočku a než stihnou zachytit signál od cíle, narazí do zdi. To se však v původním výsledku (pro porovnání je na obrázku 35b) nestalo. Roboty tam jedou po zdeformované spirále.

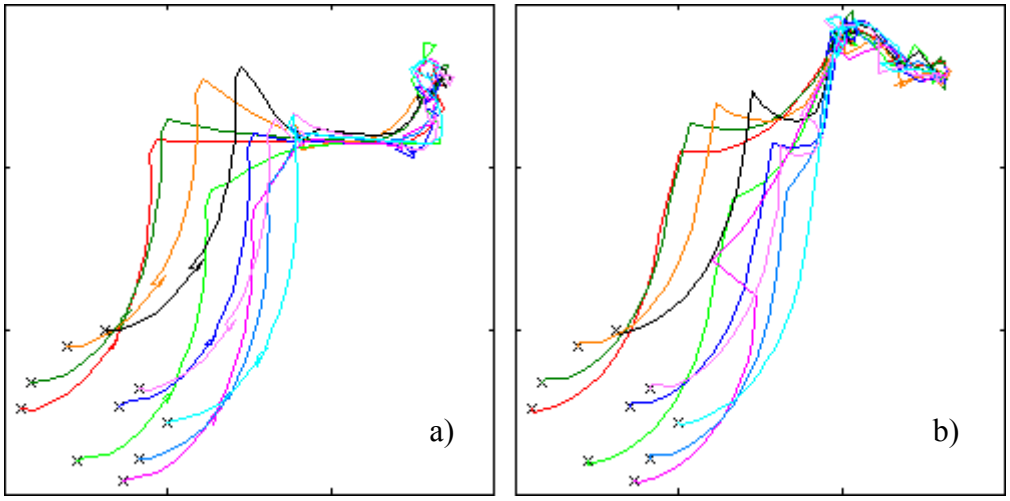
Jak vypadají trajektorie robotů v RDS při takovém tvaru pohybu, je vidět na obrázku 36. Zde již roboty dojely k místu, kde je dosah signálu od cíle, a přepnuly pohyb na PSO. Tento pohyb je závislý na náhodě, a tak jsou na obrázku ukázány dvě verze. Na obou jsou ale roboty umístěny na změřené souřadnice, nikoli na náhodné.

Jelikož je pohyb robotů v RDS definován otáčením kol a nelze přímo nastavit vektor rychlosti, nejsou trajektorie stejné jako v původním výsledku. Dále má na ně také vliv náhodných čísel v rovnicích, chybějící informace o koeficientech setrvačnosti a vzájemného působení v roji. Velké odchylky od hladkého průběhu však způsobuje metrová velikost robotů. Roboty totiž navzájem v určitých momentech kolidují

a vychylují se tak ze své původní trasy. To je dobře patrné například na obrázku 36b u fialové trajektorie. Robot se dostal do kolize s růžovým a ocitl se tak daleko od své původní trasy. Než se na ni stihl vrátit, byl již změněn pohyb na PSO. Tato ani podobné kolize však nalezení cíle nebrání, a protože v původní práci nic o vzájemném vyhýbání robotů psáno nebylo, jsou roboty ponechány vzájemným srážkám.



Obrázek 35 – Trajektorie dle teorie (a) a z výsledků (b) v původní práci



Obrázek 36 – Dvě ukázky trajektorie se zdeformovanou spirálou a PSO pohybem

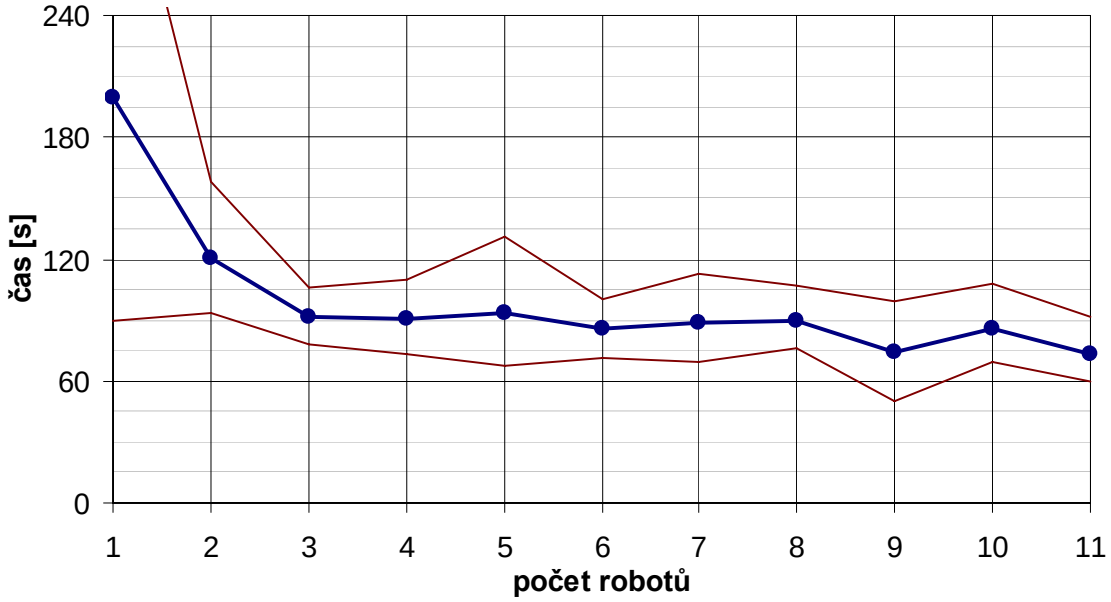
Posledním způsobem prvotního spirálovitého pohybu je pohyb po spirále s roztečí *R*, ale s posunutým středem. Tento střed byl posunut o [50; 50] od původního umístění každého robotu. Pokud je tedy robot generován na [41; 11], pak přejede rovně do [91; 61], odkud začne kroužit po spirále s roztečí 50 metrů.

Tento pohyb se zdá nejkorektnější. Nijak se v něm nedeformuje spirála a vede k nalezení cíle. Proto byl vybrán pro dodatečný experiment. Jeho cílem je odhalit, zda je opravdu k nalezení cíle potřeba 10 robotů. Byl tedy měněn počet robotů v roji od 1 do 11 a měřen čas, kdy dorazí první robot k cíli. Tím je totiž splněn úkol. Pro každý počet bylo naměřeno 8 pokusů s náhodným rozmístěním robotů v počátečním prostoru.

Vyhodnocen byl medián, průměr, minimum a maximum. Výsledné hodnoty zaokrouhlené na sekundy jsou zapsány v tabulce 6 a zakresleny na obrázku 37 (modře průměr a červeně minimum a maximum).

Tabulka 6 – Vliv počtu robotů na rychlost nalezení cíle (prvním robotem)

Robotů	1	2	3	4	5	6	7	8	9	10	11
Průměr [s]	199	121	91	90	93	86	89	90	74	86	73
Medián [s]	181	118	91	88	91	86	91	87	83	86	70
Minimum [s]	90	94	78	74	68	71	70	76	50	69	60
Maximum [s]	350	158	106	110	131	100	113	107	99	108	92

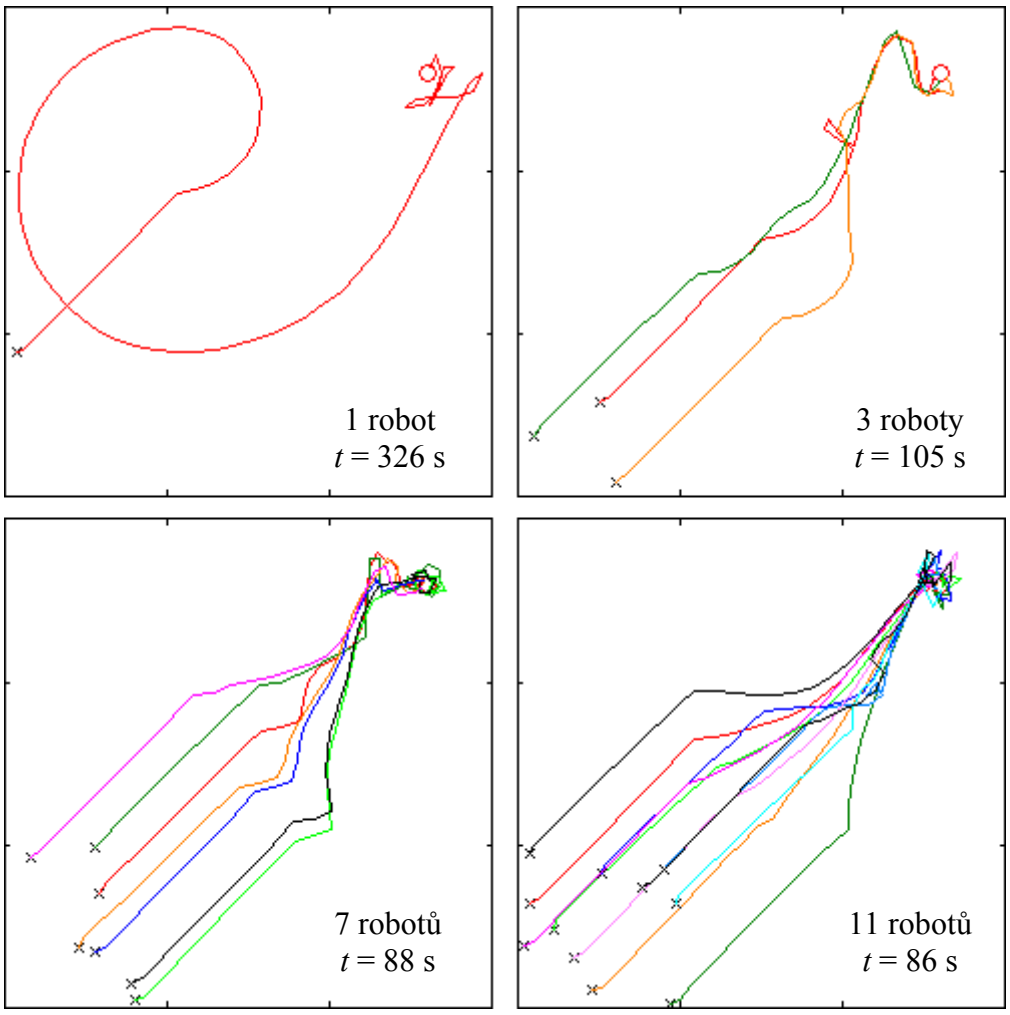


Obrázek 37 – Vliv počtu robotů na průměrnou rychlost nalezení cíle

Z obrázku je patrné, že rychlost nalezení cíle se od 3 robotů prakticky nemění. Tři roboty se tedy zdají jako ideální kombinace co největší rychlosti a nejmenšího počtu robotů. Publikovaných deset robotů bylo tedy zbytečně moc, protože cíl lze nalézt sice o 6 % pomaleji, ale s 30% počtem robotů. Výsledná úspora času se tedy nevyplatí vůči navýšené ceně kvůli množství robotů.

Proč simulace s jedním robotem funguje výrazně hůře, je naznačeno na obrázku 38 vlevo nahoře. Robot je sám, náhodně umístěn poměrně daleko od cíle, a tak musí udělat celou otáčku po spirále, než zachytí signál z vysílače u cíle. Následně pokračuje ve své trajektorii a PSO algoritmus funguje pouze na základě předešlých poloh jednoho robotu. To je sice možné, ale pomalé, a trajektorie není zrovna praktická.

Dále je na obrázku 38 případ se třemi roboty. Ty už si vzájemně předávají nejlepší polohu ve skupině, jež se liší od nejlepší polohy, které dosáhl každý robot sám. V důsledku toho je jejich výsledná trajektorie praktičtější. Protože jsou náhodně umístěné roboty tři, je mnohem větší pravděpodobnost, že jeden z nich zachytí signál již během první otočky. Tak se stalo i v případě na obrázku 38 vpravo nahoře.



Obrázek 38 – Trajektorie 1 – 11 robotů za použití spirály s posunutým středem

Dolní dvě situace na obrázku 38 ukazují již nadbytečné množství robotů. V takovém množství téměř nevykonávaly spirálovitý pohyb a hned zachytily signál od cíle. Pohyb PSO v takovém množství robotů dopadá dobře a rychle směřuje k cíli. Na obrázku 38 vpravo dole, čili pro případ 11 robotů, je okolo cíle patrné jakési klubko vytvořené z trajektorií. To vzniklo tak, že když všech 11 robotů dorazí k cíli, nevejdou se všechny do dvoumetrového okolí a navzájem do sebe naráží a občas se i odhodí.

V rámci opakování experimentu v RDS byla tedy zjištěna rozdílnost trajektorie popisované v teorii a trajektorie ve výsledku původního experimentu. Byla dohledána korektnější verze trajektorie a k ní byl posouzen vliv počtu robotů na rychlost nalezení cíle prvním z nich.

5.3 Zhodnocení RDS

Na první pohled se jeví Microsoft Robotics Developer Studio jako dokonalé. Bez nutnosti programovat fyzikální zákony a grafické prostředí lze se studentskou licencí zdarma ovládat simulované 3D roboty v graficky povedeném prostředí. S kamerou lze plynule pohybovat a sledovat tak roboty z různých úhlů, dokonce se dá kamera spojit s objektem a její pozice a natočení potom závisí na pohybu objektu. Takto lze například umístit kameru na robot.

Práce s RDS však není vůbec jednoduchá a v době psaní této práce bylo celkem málo literatury, ze které by se dalo v prostředí zorientovat. RDS lze sice programovat pomocí C#, ale místo jednoduché práce s třídami se vše řeší přes služby. Prostředí okolo robotu (VSE) je jedna služba, pohon robotu druhá, a s každým snímačem umístěným na robot vzniká další. Díky službám je zajištěna paralelní funkčnost veškerých objektů, ale programově se práce s RDS výrazně znepříjemní. Navíc RDS preferuje využívání grafického programování ve VPL, kde lze v důsledku toho většinu operací realizovat mnohem jednodušeji. Tyto faktory značně komplikují i realizaci jednoduchých úloh.

Fyzikální prostředí je v RDS sice funkční, ale některých věcí lze jenom těžko dosáhnout. V této práci bylo například potřeba vytvořit puky. Přes všechna očekávání však válec není mezi základními tvary. A fyzikální model počítá pouze s nimi. Funkci v C#, která by převedla grafický model válce importovaný z externě vytvořeného souboru na model fyzikální, se však nepodařilo nalézt. Kdo by ovšem chtěl vytvářet objekty ve tvaru pilulí, nemá se základními tvary problém. Další nepříjemností spojenou s fyzikálním prostředím je například neomezená síla motorů. Ačkoliv se robot zastaví o zeď, motor vykoná takovou sílu, že je robot odmrštěn.

Obrázek 39 – Model města v grafickém prostředí RDS

Na druhou stranu grafickému prostředí není co vytknout. Import libovolných modelů z 3D grafických editorů, použitelnost jakýchkoliv textur a volitelné nastavení kvality obrazu v závislosti na přítomné grafické kartě je vše, co stačí k příjemnému vizuálnímu zážitku. Pro slabší počítače nebo pro složitější simulace lze vypnout vykreslování, nebo přepnout na vykreslování pouhých obrysů objektů. Obrázky 39 a 40 ukazují, jak může obraz v RDS s plnou grafikou vypadat. Na obrázku 39 je vidět, že ani velké scény s četnými objekty, jako je například město, nejsou v RDS problém. V případě obrázku 40 je dokonce v prostředí RDS patrné drobné vlnění vody v bazéně. Jedná se ale pouze o grafickou animaci, nikoliv o fyzikální model vody. Fyziku pro lodě a letadla tedy v RDS rozhodně očekávat nelze.

Obrázek 40 – Model domu s přilehlým bazénem v grafickém prostředí RDS

Otázkou ale zůstává, zda je RDS vhodné na studentské projekty, které by svým vzhledem dobře reprezentovaly školu. Video ze 3D prostředí lze vyrenderovat téměř v kterémkoliv 3D grafickém editoru. RDS by mělo navíc přinášet roboty, které se budou pohybovat buď autonomně, nebo dle uživatelových příkazů.

Zdánlivě jednoduchý úkol, kde by robot jel prostředím a vyhnul se případným překážkám, nebo aby jel podél stěny, se však kvůli složité práci se službami značně komplikuje. V normálním robotu by šly po určitých intervalech číst ze snímačů údaje o vzdálenosti. Nikoliv však v RDS. Snímače jsou aktivními prvky a není možné si říct o údaj. Snímače ho samy posílají, a je tak nutno s nimi komunikovat a očekávat událost příchodu nových dat. Jiný způsob získání dat ze snímače není možný. Proto je mnohdy jednodušší a srozumitelnější číst v časových intervalech polohu robotu, což lze paradoxně velmi jednoduše, a z polohy a natočení zjistit vzdálenost zdi pomocí matematiky.

Ovládací zařízení, které by řídilo pohyb robotu, lze použít buď virtuální v počítači, nebo skutečný joystick. Opět se ale jedná o další službu, kterou je potřeba správně propojit s robotem. Celá věc lze naprogramovat vcelku jednoduše ve VPL. Vloží se dva

bločky, spojí se čarami a vše je hotovo. V C# je však tento úkol práce na hodně dlouho. A to se jedná o pouze jeden robot. Propojit s joysticky robotů více je v C# snad nadlidský úkol.

Když se však nahradí co nejvíce služeb matematickými modely, i pro více robotů RDS stíhá a program je srozumitelnější a lehčeji naprogramovatelný. Toho bylo také využito v této práci a přes ovládání v časových intervalech lze dosáhnout poměrně dobrých výsledků. Jediné, co tímto postupem není možné, je zrychlení chodu simulace.

Tabulka 7 – Přehled kladů a záporů Microsoft Robotics Developer Studia

Oblast	Klady	Zápory
Programování	Možnost C# i VPL	C# komplikovanější než VPL
Práce se službami	Zajištěna paralelizace	Nutné, velice náročné
Grafický model	Výborně provedený	–
Fyzikální model	Stačí pro základní simulace	Netradiční základní objekty

V tabulce 7 je přehled základních přínosů a nedostatků RDS. Jedinou bezchybnou částí RDS je tady jeho grafická stránka. Lze je proto spíše využít jako nástroj na prezentaci jednoduchých úloh v hezkém prostředí, než jako kvalitní simulátor robotiky. Pro tento účel je totiž RDS v C# pro běžné studenty příliš složité.

6 ZÁVĚR

V rámci diplomové práce byla provedena rešerše zaměřená na rojovou inteligenci a na základní typy pohybu skupin robotů. V další části byl zpracován stručný úvod do práce s RDS. Poté byly vybrány a důkladně rozebrány dva již publikované experimenty využívající rojovou inteligenci. V prvním experimentu mají roboty za úkol shromažďovat puky do shluků a ve druhém mají za úkol vyhledávat cíl. Tyto experimenty byly zopakovány v prostředí RDS.

Dokumentace k prvnímu experimentu obsahuje detailní popis prostředí, chybí však údaje o rychlostech robotů. Proto bylo možné porovnat pouze tvar výsledné křivky. Ten je velmi podobný publikovanému i přes omezující vlastnost RDS, kdy bylo nutno puky nahradit kostkami. Dále byl tento experiment rozšířen o vliv počtu robotů na rychlost splnění úkolu. Vyšší počet robotů znamenal rychlejší shromažďování, ale od 4 robotů výše roboty soupeřily o poslední puky, a výsledný čas tak zaznamenal nárůst.

Při opakování druhého experimentu bylo zjištěno, že dokumentace obsahuje chybné údaje o první fázi pohybu robotů. Proto byly v RDS pro první fázi realizovány 3 různé typy spirálovitého pohybu. První typ pohybu byl naprogramován podle teorie v dokumentaci k experimentu a selhal nárazem robotů do stěny. Druhý typ pohybu, vyčtený z výsledků původního experimentu, dosahoval podobných výsledků jako originál, pouze náhodný pohyb dle PSO se lišil. K cíli ale tentokrát roboty dojely. Na třetím typu pohybu, který byl publikován v literatuře odkazované v dokumentaci k experimentu, byl opět navíc vyzkoušen vliv počtu robotů na rychlost splnění cíle. Ukázalo se, že do 3 robotů má navyšování výrazný vliv, zatímco pro 4 a více robotů je urychlení nalezení cíle velmi pozvolné.

Na základě realizace těchto dvou příkladů je poukázáno na nevýhody (složitá práce se službami, netradiční základní tvary těles a upřednostňování VPL před C#) a výhody (pěkné grafické provedení a již funkční základní fyzika a modely robotů) simulačního prostředí RDS. Při omezení práce se službami je ve výsledku vcelku použitelné pro jednoduché úlohy, které dobře vypadají a jsou reprezentativní. Pro složitější úlohy, jejichž cílem je modelovat a simulovat chování skutečných robotů, je RDS kvůli velkému množství služeb a komunikaci mezi nimi pro běžné studenty příliš náročné.

Literatura

[1] ABRAHAM, Ajith – GUO, He – LIU, Hongbo. *Swarm Intelligence: Foundations, Perspectives and Applications*. Springer-Verlag Berlin Heidelberg, 2006. Dostupné z: <http://www.softcomputing.net/swarm-chapter.pdf>

[2] TSANKOVA, D. – GEORGIEVA V. – ZEZULKA F. – BRADÁČ Z. *Immune navigation control fro stigmergy based foraging behaviour of autonomous mobile robots*. Brno University of Technology, 2005. Dostupné z: <http://www.nt.ntnu.no/users/skoge/prost/proceedings/ifac2005/Fullpapers/02607.pdf>

[3] XUE, Song-dong – ZENG, Jian-chao. *Control over Swarm Robots Search with Swarm Intelligence Principles*. Lanzhou University of Technology, 2008. Dostupné z: <http://www1.tyust.edu.cn/yuanxi/yjjg/paper/2008/Control%20over%20Swarm%20Robots%20Search%20with%20Swarm%20Intelligence%20Principles.pdf>

[4] *Robot Soccer: YSR-A System Manual*. Seoul: Yujin Robotics Co., Ltd., 2005. Dostupné z: <http://www.ee.ust.hk/~mirosot/User%20Manual.pdf>

[5] PARK, Kui-Hong – KIM ,Yong-Jae – KIM, Jong-Hwan. *Modified Uni-Vector Field Navigation and Modular Q-learning for Soccer Robots*. Korea Advanced Institute of Science and Technology, 2001. Dostupné z: <http://rit.kaist.ac.kr/upload_files/conference/Modified_(ISR01).pdf>

[6] WOLF, J.C. – ROBINSON, P. – DAVIES, J.M. *Vector field path planning and control of an autonomous robot in a dynamic environment*. University of Plymouth, 2004. Dostupné z: <http://www.swrtec.de/swrtec/research/publications/VECTOR_FIELD_PATH_PLANNING_AND_CONTROL_OF_AN_AUTONOMOUS_ROBOT_IN_A_DYNAMIC_ENVIRONMENT.pdf>

[7] ROWSTRON, A. – BRADSHAW, B. – CROSBY, D. – EDMONDS, T. – HODGES, S. – HOPPER, A. – LLOYD, S. – WANG, J. – WRAY, S. *The Cambridge University Robot Football Team Description*. Springer-Verlag Heidelberg Berlin, 1999. Dostupné z: <http://www.ifm.eng.cam.ac.uk/automation/publications/documents/robot-football-description.pdf>

[8] KOO, Mi-Hoi – LEE, Yun-Ki – LEE, Kang-Hee – KIM, Tae-Hun – LEE, Jae-Kyung – KIM, Jong-Hwan. *Development of robot soccer system for 11-A-SIDE MiroSot*. Korea Advanced Institute of Science and Technology, 2004. Dostupné z: <http://rit.kaist.ac.kr/upload_files/conference/Development%20of%20Robot%20Soccer%20System%20for%2011-A-Side%20MiroSot.pdf>

[9] CAO, Wei – JASON, Yanqing Gao – MACE, Robert. *Formation and Cooperation for SWARMed Intelligent Robots*. American Society for Engineering Education, 2008. Dostupné z: <http://www.asee.org/documents/sections/middle-atlantic/spring-2008/02-Formation-and-Cooperation-for-SWARMed-Intelligent-Robots.pdf>

[10] MILLINGTON, Ian. *Artificial intelligence for games*. San Francisco, 2006. ISBN-10: 0-12-497782-0

[11] MORGAN, Sara. *Programming Microsoft Robotics Studio*. Washington, 2008. ISBN-10: 0-73-562432-1

[12] HAYES, Adam T., *Self-Organized Robotic System Design and Autonomous Odor Localization*. California Institute of Technology, 2002. Dostupné z: <http://thesis.library.caltech.edu/2544/1/aththesis.pdf>

Seznam obrázků

Seznam tabulek

Seznam zkratek

DSS – Decentralized Software Services
PSO – Particle Swarm Optimization
RDS – Robotics Developer Studio
SI – Swarm Intelligence
VPL – Visual Programming Language
VSE – Visual Simulation Environment

Obsah přiloženého CD

Elektronická verze práce (PDF)
Programy, skripty a výstupy ze simulací (ZIP)