Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Ilkovičova 3, 812 19 Bratislava

Tvorba softvérového systému v tíme

RoboCup – nové stratégie

(Dokumentácia k projektu)

Tím č. 12 – 12. hráč

Bc. Juraj Ligocký, Bc. Michal Hrubý, Bc. Gabriel Pán
Bc. Vladimír Oravec, Bc. Ján Hric, Bc. Marek Polák
Pedagogický vedúci: Ing. Ivan Kapustík

2008/2009

OBSAH

1 ÚVOD

1.1 Účel dokumentu

Tento dokument vznikol v rámci predmetov Tvorba softvérového systému v tíme a Tvorba informačného systému v tíme a zaoberá sa problematikou simulovaného robotického futbalu – RoboCupu [1]. Cieľom projektu je priniesť do sveta RoboCupu na našej fakulte kvalitného hráča, ktorý je výsledkom vylepšenia už existujúceho hráča pomocou rôznych algoritmov spolupráce agentov, rozhodovacích stromov, neurónových sietí a pod. Výsledkom nášho úsilia bude regionálne kolo turnaja RoboCupu, ktoré sa uskutoční na pôde FIIT STU v Bratislave.

Prvá kapitola dokumentu sa stručne zaoberá simulovanému robotickému futbalu.

Druhá kapitola je venovaná podrobnejšej analýze. Opisuje architektúru RoboCupu, predovšetkým simulujúci server. Okrem toho prezentuje poznatky získané jednotlivými členmi hlbšou analýzou fakultných a zahraničných tímov, zameriavajúc sa na ich jedinečné vlastnosti, výhody a nevýhody. Ďalej sú v kapitole zahrnuté výsledky z testovania a skúmania dvoch tímov.

Tretia kapitola obsahuje špecifikáciu riešení, ktorým sa budeme venovať najmä v rámci prototypu.

1.2 RoboCup [7]

RoboCup je medzinárodný projekt, ktorý má za úlohu podporovať výskum umelej inteligencie, robotiky a podobných oblastí. RoboCup sa to snaží pomocou futbalu, keď futbal sa stáva hlavnou témou výskumu.

Hlavným cieľom RoboCupu je do roku 2050 vytvoriť tím plne autonómnych humanoidných robotov, ktorí dokážu poraziť najlepších futbalistov (ľudí). Tento cieľ sa môže zdať príliš ambiciózny, ale je to výzva pre umelú inteligenciu na najbližších 42 rokov. Napríklad prešlo len 50 rokov od vynájdenia počítača k vytvoreniu superpočítača Deep Blue, ktorý dokázal poraziť aktuálneho svetového šampióna v šachu.

Na dosiahnutie tohto cieľa je potrebné vytvoriť nové technológie vo viacerých oblastiach ako napr. nové princípy autonómnych agentov, spolupráca v multiagentových systémoch, reagovanie agentov v reálnom čase, výskum senzorov a pod. Jednou z hlavných oblastí, kde sa dajú využiť technológie RoboCupu, je napr. hľadanie a záchrana zranených pri veľkých katastrofách. Touto oblasťou sa zaoberá projekt RoboCupRescue.

RoboCup má tri hlavné kategórie: RoboCupSoccer (simulácia futbalového zápasu), RoboCupRescue (hľadanie a záchrana ľudí pri katastrofách) a RoboCupJunior (určený na vzdelávanie študentov do 19 rokov). Tieto kategórie sa ešte delia na ďalšie. Náš tím sa bude zaoberať kategóriou RoboCupSoccer Simulation.

RoboCupSoccer Simulation slúži na výskum a vzdelávanie sa v oblasti umelej inteligencie a multiagentových systémov. Umožňuje dvom tímom jedenástich hráčov simulovať autonómnych robotov pri hraní futbalu.

Vlastná simulácia futbalového zápasu prebieha na servri, keď na server sú pripojení jednotliví hráči (agenti). Agenti získavajú zo servra rôzne informácie

a na základe nich reagujú. Prebiehajúci zápas sa dá prezerať pomocou tzv. monitora, ktorý na obrazovku zobrazuje prebiehajúcu (alebo uloženú) hru.

1.3 Zadanie

Téme RoboCup, presnejšie lige simulovaného robotického futbalu, sa naši študenti venujú už deviaty rok. Tímy študentov, či už v rámci umelej inteligencie alebo tímového projektu, sa snažia vytvárať a vylepšovať programy, ktoré simulujú správanie sa futbalového hráča. Každý tím sa v rámci obmedzení určených pravidlami futbalu a špecifikami simulačného prostredia snaží vytvoriť čo najlepšieho hráča. Mužstvo vytvorené z takýchto hráčov by malo vyhrať nad mužstvom súpera. O súťaži a doterajšej činnosti je dosť popísané aj na stránke STU turnaja v simulovanom robotickom futbale[1].

V rámci fakulty sa realizovalo viacero súťaží a posledné ročníky už boli oficiálnymi turnajmi iniciatívy RoboCup. Množstvo pozitívnych ohlasov priviedlo organizátorov k vyhláseniu ďalšieho regionálneho turnaja RoboCup v simulovanej lige, opäť na záver akademického roka. Práve množstvo nových prístupov a riešení, ktoré predviedli nielen študenti tímového projektu, ale aj študenti umelej inteligencie, ukázali, že možnosti na vylepšovanie hráča nie sú zďaleka vyčerpané a dokonca sa stále rodia prekvapujúce úspešné riešenia. Preto má téma RoboCupu podnázov „Nové stratégie“. Znamená to všeobecne hľadanie nových prístupov a stratégií nielen pre hráča, ale aj vo svojej práci, v úpravách zdrojového kódu, podporných aplikáciách, základných aj vyšších schopnostiach hráča, spôsobe učenia a ladenia počas simulácií. Nové stratégie sú komplexnou výzvou do nového kola víťazstiev!

Na spresnenie je vhodné povedať, že v tomto tímovom projekte budeme rozširovať možnosti a vylepšovať správanie hráčov, vytvorených vo vlaňajších tímových projektoch. Využije sa existujúci zdrojový kód, dokumentácia a aj vytvorené podporné aplikácie. Musí sa zachovať (a podľa možností aj zlepšiť) aj modularita a tým aj rozšíriteľnosť hráča. Zimný semester je vyhradený na oboznámenie sa s celým prostredím, najmä existujúcimi hráčmi a návrhu a prototypovej realizácii jeho vylepšení. Očakáva sa najmä návrh nových prístupov a stratégií vo všetkých už spomenutých oblastiach. Vybrané prístupy sa overia vytvorením jedného alebo viacerých prototypových rozšírení existujúceho kódu. Dôležitou súčasťou bude vytvorenie plánu implementácie a overovania nových stratégií v nasledovnom semestri. V letnom semestri nás čaká realizácia navrhnutých prístupov a stratégií a ich overovanie. Produkt by mal byť dohotovený v deviatom až desiatom týždni semestra, potom je potrebné venovať sa ladeniu a optimalizácii hráča na súťaž, ktorej výsledky idú do celkového hodnotenia tohto projektu.

[1] www.fiit.stuba.sk/robocup

2 ANALÝZA

Kapitola Analýza uvádza poznatky nadobudnuté naším tímom o prostredí simulovaného futbalu. V prvom rade ide o mechanizmus, na akom je celý systém postavený. Ďalej sa zaoberá situáciou vývoja v oblasti RoboCupu vo svete, no najmä na našej fakulte. Úlohou analýzy je získanie predstavy o možnostiach prínosu nášho tímu do sféry simulovaného robotického futbalu.

2.1 Prostredie simulovaného futbalu

Soccer server je systém predstavujúci simulačné prostredie pre virtuálne ihrisko. Celý mechanizmus simulácie je typu klient-server, hráči sa pripoja na server a komunikujú s ním pomocou protokolu UDP. Server poskytuje hráčom informácie o stave hry a senzorické informácie o objektoch na ihrisku. Hráči posielajú servru svoje požiadavky na akcie, ktoré chcú vykonať. Na externé sledovanie simulácie možno použiť program Soccer monitor. Existuje viac druhov s rozličnými vlasnosťami, základom je zobrazenie plochy ihriska, hráčov a lopty a možnosť spustenia zápasu.

2.1.1 Senzory

Senzory umožňujú hráčovi vnímať okolie. Sú ekvivalentom zmyslov u ľudí. Hráči majú tri druhy senzorov, na základe nimi prijatých informácií si vytvárajú predstavu o svete. Opis senzorov je prebratý z [7].

Zrak

Zrakový senzor dáva hráčovi informácie o objektoch, ktoré sa nachodia v jeho zornom poli. Informácie sú posielané pomocou funkcie `(see (Time) (ObjInfo) (ObjInfo) ... )` periodicky v stanovenom kroku. `ObjectInfo` je objekt, ktorý obsahuje napríklad informácie o relatívnej vzdialenosti (`Distance`), smere (`Direction`), natočení (`FaceDir`), čísle hráča (`UNum`) a mene tímu (`TeamName`) objektu. Informácie, ktoré hráč vidí, sú zašumené tým viac, čím je objekt ďalej od hráča. S väčšou vzdialenosťou je aj množstvo informácií menšie, teda postupne hráč prestane rozoznávať číslo dresu a príslušnosť k tímu. Kvalitu pohľadu hráča možno zmeniť pomocou funkcie `(change_view (ANGLE_WIDTH) (QUALITY))`, kde `ANGLE_WIDTH` môžu byť hodnoty `normal`, `wide`, `narrow` a `QUALITY high` a `low`. Po hracom poli je viacero značiek, ktoré hráč môže vnímať a podľa ktorých sa môže orientovať (bránky, rohy, stred ihriska).

Sluch

Sluchový senzor slúži na počúvanie správ od iných hráčov, kouča alebo rozhodcu. Samotné počúvanie je realizované funkciou `(hear (Time) (Direction) (Message))`, kde `Time` je aktuálny čas, `Direction` je smer zdroja. Ak chce hráč poslať správu ostatným hráčom, je mu to umožnené prostredníctvom funkcie `(send (Message))`. Táto správa nesmie byť dlhšia ako desať bajtov. Ak hráč bude môcť počuť naraz viacero správ, tak začuje náhodnú. Správu od rozhodcu začuje vždy. Správy povedané hráčmi majú obmedzený dosah. Hráč si však môže nastaviť, ktorého hráča

chce prednostne počúvať. V prípade, že bude počuť viacero správ naraz, rozozná správu práve od tohto hráča. Ak ten hráč nič nezakričal, tak opäť začuje náhodnú správu.

Pocit tela

Telový senzor umožňuje hráčovi zistiť svoj fyzický stav pomocou funkcie `sense_body`. Dal by sa prirovnať k rôznym pocitom u ľudí. Server poskytuje hráčovi informácie o jeho rýchlosti, výdrži, šírke pohľadu, počte poslaných správ a pod. Hráč by mal vedieť tieto informácie správne využiť a zaradiť ich do vyhodnocovacieho procesu svojej akcie (napr. nebude bežať plnou rýchlosťou, keď má málo energie).

2.1.2 Režimy hry

Podľa režimu hry majú hráči rôzne možnosti akcií. Server pozná tieto režimy hry:

Play on

Toto je základný režim hry. Nastáva, keď sa lopta začne pohybovať vďaka kopnutiu z iného hracieho režimu (rozohranie lopty, aut, výkop od brány, roh).

Kick off

Základné rozohranie lopty, pri ktorom musí byť každý hráč na svojej strane. Ak sa na nej nenachádza, server ho umiestni na náhodné miesto na príslušnej polovici hracieho poľa.

Gól

Nastáva v momente, keď sa lopta dostane svojím celým objemom do brány za bránkovú čiaru. Server preruší zápas na päť sekúnd, presunie loptu do stredu ihriska, pošle správu všetkým hráčom. Počas prerušenia sa hráči môžu presunúť na svoju polovicu pomocou príkazu `move`. V opačnom prípade ich presunie na náhodnú pozíciu server. Server následne zmení hrací režim na `kick_off`.

Aut, roh a výkop od brány

Nastáva, keď lopta svojím objemom opustí hraciu plochu. Server presunie loptu na príslušnú pozíciu (v prípade autu tam, kde opustila ihrisko; v prípade prekročenia bránkovej čiary buď do rohu, buď na hranicu výkopu od brány). Následne nastaví príslušný herný režim.

Presunutie

Nastáva pri rozohrávaní lopty (`kick_off`, `throw_in`, `corner_kick`, `goal_kick`, `offside`). Server presunie hráčov tak, aby boli minimálne vo vzdialenosti 9,15 metra od lopty. Hráči sú presunutí na obvod pomyselnej kružnice.

Presun hráča pomocou príkazu `move` je na pozíciu `(X, Y)`. Stred súradnicovej sústavy je v centrálnom kruhu, kladný smer osi `X` je určený smerom k súperovej bráne, kladný smer osi `Y` je určený smerom k pravej čiare (ktorá je určená vzhľadom na kladný smer osi `X`). Tento príkaz je k dispozícii iba v režime `before_kick_off`.

Polčas

Server preruší zápas v polovici plánovaného času trvania. Štandardná dĺžka polčasu je 3 000 simulačných taktov (okolo päť minút). Ak je po skončení zápasu skóre nerozhodné, predlžuje sa, kým sa nedosiahne gól – tzv. zlatý gól.

2.1.3 Akcie hráča

Pod akciami chápeme príkazy, ktoré môže hráč servru poslať na vykonanie určitej činnosti v nasledovnom simulačnom takte.

- Dash – zrýchlenie hráča v smere jeho tela. Zohľadňuje sa výdrž hráča.
- Kick – kopnutie do lopty určitou silou v určitom smere.
- Catch – chytenie lopty v určitom smere, používa len brankár. Server má definovaný interval, ako často možno príkaz chytenia opakovať.
- Say – poslanie krátkej správy spoluhráčom i protihráčom.
- Pozornosť na hráča – prednostné počúvanie určitého hráča. Ak daný hráč dačo povie súčasne s inými, práve jeho správu hráč zachytí.
- Otočenie hráča – zmena smeru hráčovho tela o určitý moment. V prípade stojaceho hráča je moment rovný uhlu otočenia. Zvyšovaním rýchlosti behu sa uhol zmenšuje.
- Otočenie hlavy – zmena smeru otočenia hlavy hráča relatívne k telu.
- Tackle – pokus o obratie súpera o loptu. Je to tzv. kĺzačka, hráč sa snaží z celej sily kopnúť do lopty. Desať taktov po vykonaní príkazu je hráč nečinný.

2.1.4 Heterogénni hráči

Server umožňuje hru s heterogénnymi hráčmi, čiže s hráčmi s rôznymi vlastnosťami. Na začiatku hry sa okrem základného hráča vygeneruje ďalších šesť náhodných typov. Podstatou heterogénnosti hráčov je zmena daktorých vlastností proti základnému hráčovi. Celkovo však platí, že čím lepšia jedna vlastnosť, tým horšia iná. Takto sa otvárajú nové možnosti na obsadenie rol v tíme. Napr. hráč s veľkou výdržou a nízkou presnosťou kopu je vhodným obrancom, no ťažšie sa uplatní v útoku.

2.1.5 Rozhodca

Rozhodcu predstavuje „obsluha“ simulácie, čiže je to externý zásah do systému. Rozhodca má právo zmeniť režim hry v prípade, ak nastala neštandardná situácia. Príkladom je faul. Faul síce nie je v simulácii definovaný, no sú určité dohody „správnej“ hry (napr. nesmie sa vytvoriť obranný múr z hráčov pred bránou), ktorých porušenie môže priniesť tímu penalizáciu. Rozhodca vie hráčom poslať zvukové správy. Hráči tieto správy zachytia v každom prípade.

2.1.6 Kouč a tréner

Kouč aj tréner slúžia na organizáciu hráčov, majú však rôzne úlohy. Kouč aj tréner dostávajú – na rozdiel od hráčov – neskreslené informácie o svete. Kouč je prítomný aj počas riadnych zápasov. Vie hráčom poslať informácie v podobe zvukových správ, ich frekvencia je však omnoho nižšia než komunikácia medzi hráčmi samými. Jeho hlavnou úlohou je analyzovať hru a na základe analýzy vyhodnotiť

najlepšiu stratégiu na najbližšiu etapu zápasu. Okrem dávania pokynov má možnosť vystriedať hráčov. Tréner, na druhej strane, slúži na trénovanie hráčov mimo riadnych zápasov. Veľmi vhodný je na učenie situácií.

2.2 Analýza tímov

Najprv predstavíme niekoľko súčasných zahraničných tímov. Keďže k zahraničným tímom často nie je dostupná kvalitná dokumentácia, ani plný zdrojový kód, ich analýza nie je hĺbková. Má poskytnúť najmä inšpiráciu pre našu prácu na vylepšení domáceho tímu. Fakultné tímy produkujú každoročne viac-menej podrobné dokumenty a zdrojové kódy sú plne dostupné, preto ich rozoberieme dôkladnejšie.

2.2.1 Nexus

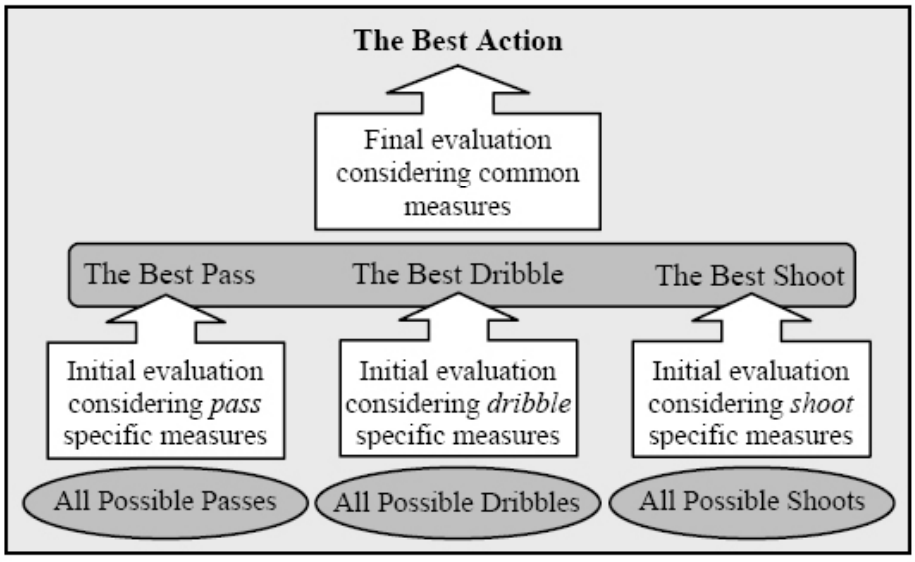
Tím Nexus [2] je arabského pôvodu. Zúčastnil sa minulého svetového ročníka v robotickom futbale a vypadol v druhom kole. Pri tomto tíme je zaujímavé, ako sa vyberá najlepšia nasledovná akcia pre určitého hráča. Pri algoritme sa používa špeciálna dvojfázová metóda, ktorú autori prebrali od tímu TsinghuAeolus – víťaza svetového turnaja v RoboCupe v rokoch 2001 a 2002. V robotickom futbale môže hráč urobiť s loptou tri kroky: nahrať, strieľať alebo driblovať. Informácie si delia na tie, ktoré sú špecifické alebo potrebné pre danú činnosť, a tie, ktoré sú spoločné pre všetky druhy akcií.

Parameter	S	D	P
Ball distance from the center of the opponent's goal	y	n	n
Relative direction of the goalie and the ball motion	y	n	n
Dribbling length	n	y	n
Relative angle of player's body and the dribbling direction	n	y	n
Distance between the passer and the ball	n	n	y
Movement direction of the pass receiver	n	n	y
Sensitivity of the target area	y	y	y
The density of the opponents in the target area	y	y	y
Probability of the ball interception by the opponents	y	y	y
Updating degree of the received information as to target area	y	y	y

Obr. 1. Rôzne parametre hráča tímu Nexus

```
00:  max_priority = 0
01:  selected_action = no_action
02:  for each feasible action (FA) do
03:    priority = 0
04:    for each evaluation measure (EM) do
05:      priority +=  EM.weight
06:    end for
07:    if priority > max_priority then
08:      max_priority = priority
09:      selected_action = FA
10:    end if
11:  end for
```

Obr. 2. Algoritmus pre jednofázovú metódu tímu Nexus



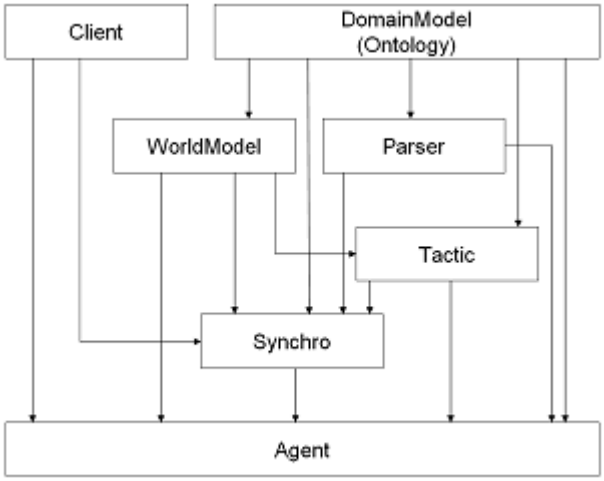
Obr. 3. Algoritmus pre dvojfázovú metódu tímu Nexus

2.2.2 DAInamite

Tím Dainamite [3] vyvíja svojho vlastného hráča od nuly. Ako programové prostredie si zvolili Javu. Výsledný framework je komplexný produkt, ktorý zahŕňa všetky potrebné časti pre správny chod. Zaujímavými sú vylepšenia tohto frameworku v aktuálnej verzii.

Architektúra hráča

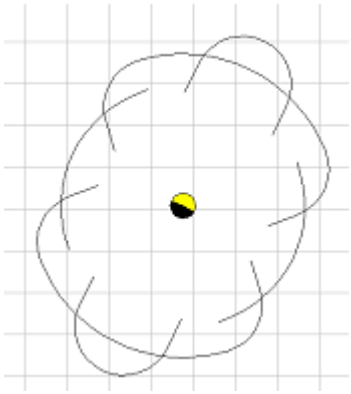
Na zachovanie čo najväčšej flexibility bol použitý Spring-Framework na konfiguráciu štruktúry hráča za behu. To im dovoľuje uloženie závislostí do súboru formátu XML namiesto ich uloženia priamo v kóde, takže výmena komponentov je iba jednoduchá záležitosť konfigurácie. Každý komponent má svoju vlastnú konfiguráciu, ktorá môže byť tiež rozšírená. Prehľad existujúcich modulov a ich závislosti možno vidieť na obrázku.



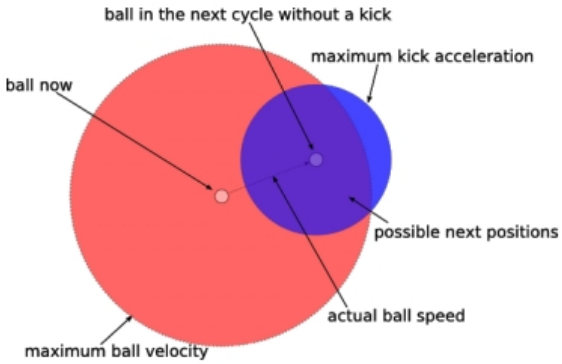
Obr. 4. Závislosti medzi jednotlivými modulmi architektúry hráča tímu DAInamite

Model akcie

Predchádzajúca metóda výberu akcie s loptou bola rozdelená na rôzne akcie ako driblovanie, podržanie, prihrávka, strela. Tieto akcie autori rozanalyzovali a zistili nedostatky. Po prvé bolo ťažko porovnať a následne rozhodnúť, ktorá akcia je najlepšia v danom okamihu. Po druhé nevieme povedať, či sme nezabudli na nejakú vhodnejšiu akciu. Na vyriešenie týchto problémov autori implementovali triedy `ReachableArea` a `BallArea`. `ReachableArea` je geometrický model plochy, ktorú môže hráč dosiahnuť za daný čas. `BallArea` reprezentuje, kde sa môže lopta nachádzať po náhodnom kope hráča.



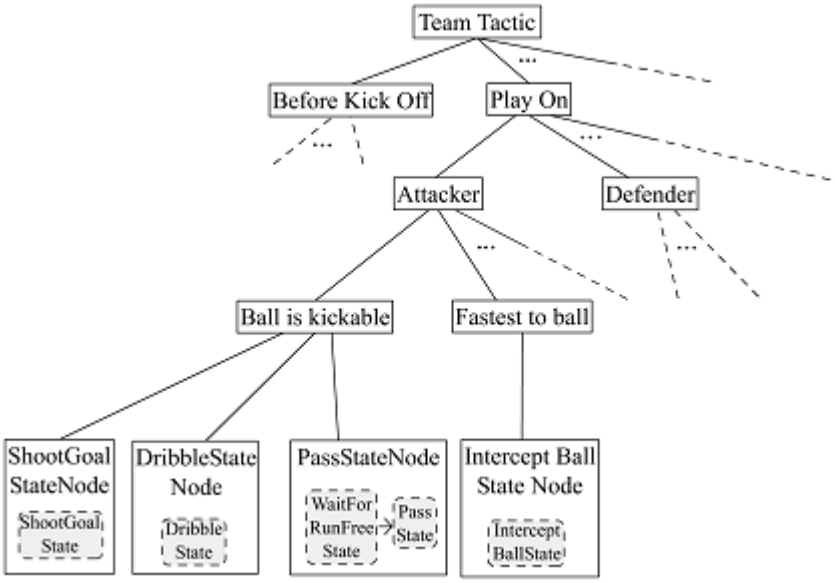
Obr. 5. Plocha, ktorú dokáže hráč dosiahnuť za čas – tím DAInamite



Obr. 6. Lopta s rýchlosťou a pozíciou – tím DAInamite

Taktika

Nový mechanizmus výberu akcie je zodpovedný za kalkuláciu najlepšej akcie na základe dostupného pohľadu. Tento mechanizmus je realizovaný pomocou rozhodovacieho stromu, ktorý obsahuje taktiku každej roly ako podstrom. Takže môžeme modelovať individuálne správanie sa každého spoluhráča a dynamické prideľovanie rol. Vrcholy stromu predstavujú podmienené varianty a listy predstavujú, čo sa stane. Listy obsahujú tzv. stavy (angl. states), ktoré sú zodpovedné za výpočet špecifickej akcie. Prechádzaním cez strom možno určiť najlepšiu akciu, ktorá sa má vykonať.



Obr. 7. Rozhodovací strom tímu DAInamite

Zhodnotenie

Pri tomto tíme nás zaujala ani nie tak jeho štruktúra či špeciálne vlastnosti ako jeho implementačné prostredie. Väčšina členov tímu má slabé skúsenosti s programovacím jazykom C++. Predpokladáme, že budúce roky sa situácia bude opakovať. Java predstavuje v súčasnosti jeden z najrozšírenejších programovacích jazykov, preto je prirodzené, že aj v oblasti RoboCupu sa začínajú objavovať hráči

v ňom napísaní. Na našej fakulte sa zatiaľ používa výhradne C++. Pokúsime sa zistiť, do akej miery sme schopní a či je účelné tento stav zmeniť.

2.2.3 Oxsy

Tím Oxsy [4] pochádza z Rumunska. Vznikol ako diplomový projekt študenta Mariana Sebastiana, no autor sa mu stále venuje. Na tohtoročnom turnaji skončil piaty.

Konaj prv, než udalosť nastane

Autor považuje túto metódu za základ fungovania tímu. Cieľom je, aby celý tím konal ako jedna myseľ, aby jednotliví hráči predpokladali akcie spoluhráčov i protihráčov bez nutnosti komunikácie.

Daj a bež

Jednoduchým príkladom predpovedania je akcia daj a bež, keď sa dvaja hráči snažia prejsť cez súperovho obrancu. Hráč nahrá spoluhráčovi a začne bežať na voľné miesto alebo jedným smerom, kde čaká spätnú prihrávku. Čiže musí predpokladať budúcu prihrávku od daného hráča a začne bežať skôr, než prihrávka nastane, dokonca ešte pred zachytením lopty spoluhráčom.

Výmena rol

Túto akciu využíva tím v obrane. Ak sa súperovmu hráčovi podarí prejsť cez obrancu po kraji ihriska ako dôsledok neúspešnej kĺzačky (tackle) krajného obrancu, ktorý potom nedokáže istý čas hrať, stredný obranca nečaká, kým súper príde do jeho oblasti. Hneď sa vyberie proti súperovi, aby ho zastavil, čím nahradí krajného obrancu. Pôvodný krajný obranca, keď sa už zregeneruje, beží na miesto stredného obrancu. Krajný a stredný obranca si tak vymenili roly. Je to ďalší príklad predpovedania udalostí. Podstatou je prekvapivý moment pre súpera, ktorý prihrávku nečaká.

Predpovedanie umožňuje tímu reagovať na nečakané udalosti, a na druhej strane, tím sa stáva menej predvídateľným, čo z neho robí silnejšieho súpera.

Postrehy zo zápasov

- Brankár vybieha aj do stredu ihriska, čím pomáha rýchlo založiť útok. Robí to len vtedy, keď si je istý, že k lopte dobehne prv, než súper.
- Tím je rozdelený na tri skupiny: štyria obrancovia, traja stredopoliari a traja útočníci. Hráči v skupine sa zdržujú takmer výlučne vo vertikálnom zástupe. Je to jednoduchá a pomerne účinná „formácia“.
- Obrancovia sa snažia vždy stáť pri svojom brankárovi bližšie než ktorýkoľvek súper. Sú však situácie, keď sa dostanú na úroveň súpera, pravdepodobne v snahe zachovať zástup, čo môže spôsobiť ťažkosti.
- Obrancovia idú len málo za stredovú čiaru, aj počas útoku sa zdržujú tesne pri čiare. Útočníci – podobne – vždy čakajú v súperovej polke, zriedka prejdú za čiaru.

Zhodnotenie

O kvalite tímu Oxsy svedčí jeho umiestenie na popredných miestach v posledných rokoch. Dostupná dokumentácia k tímu je veľmi slabá. Opísaná metóda predpovedania

udalostí by sa dala prirovnať k rozpoznávaniu akcií v návrhu tímu FIITMedia, hráči však medzi sebou nemusia komunikovať. Pozorovanie zápasov tímu Oxsy odhaľuje jednoduchú a účinnú hru: Hráči sú stále zoradení v akejsi formácii, ktorej zakomponovanie do nášho budúceho hráča by stálo za vyskúšanie.

2.2.4 Brainstormers

Brainstormers [5] je nemecký tím s desaťročnou tradíciou pôsobiaci na Univerzite Osnabruck v Nemecku. Zároveň je minuloročným víťazov majstrovstiev sveta, ktoré sa konali v čínskom Su-čou. Základné princípy, o ktoré sa opierajú sú:

- sú dva základné moduly: modul sveta a rozhodovací modul
- vstupom do rozhodovacieho modulu je približný celkový model sveta, ako ho poskytuje simulácia
- prostredie futbalu je modelované ako Markovov rozhodovací proces
- rozhodovanie je organizované v komplexných a menej komplexných typoch správania
- moduly správania sa učia pomocou učenia odmenou a trestom
- moderné metódy umelej inteligencie sú aplikované všade, kde je to možné a užitočné

Proces rozhodovania hráča je založený na robotických architektúrach založených na správaní (angl. behavior-based). Skupina viac či menej komplexných správaní sa podieľa na tvorbe rozhodnutia agenta. Do určitého stupňa je možné charakterizovať danú architektúru za hierarchickú, pričom je však možné, aby správanie nižšej úrovne zavolalo správanie vyššej a tým vyvolalo požadovanú akciu.

Tím Brainstormers sa zameral vo svojom poslednom výskume na implementovanie tzv. agresívneho správania. Ide o získanie lopty, ktorú má vo vlastníctve súper, a jej prihranie spoluhráčovi. Pri implementácii využili viacvrstvové perceptrónové neurónové siete, pričom úspešnosť zisku lopty zo všetkých pokusov bola 80%. 5% tvorilo chybné správanie.

Druhým prínosom tímu je vytvorenie a sprístupnenie nástroja nazvaného rcg2swf[2]. Ide o aplikáciu spustiteľnú z príkazového riadka, ktorá z logu zápasu dokáže vytvoriť pútavú animáciu formátu Flash. Program má množstvo nastavení a medzi jeho prednosti patrí realistický vzhľad trávnika, použitie avatarov namiesto kruhov predstavujúcich hráčov a automatickú zmenu strán po polčase. Bolo by výhodné použiť ho pri analýze zápasov, prípadne pri prezentácii výsledkov projektu.

Zhodnotenie

Na hráčovi tímu Brainstormers nás zaujala predovšetkým kvalitná neurónová sieť pre agresívne správanie. Keďže existuje podrobná dokumentácia tejto metódy a je dostupný aj zdrojový kód neurónovej siete, považujeme za reálne pokúsiť sa zahrnúť agresívne správanie aj do nášho budúceho hráča.

[2] http://www.ni.uos.de/index.php?id=1024

2.2.5 Jahodoví princovia

Tím Jahodoví princovia [6] je minuloročným víťazom fakultnej súťaže. Pri našej analýze, prototypovaní a implementácie sa pravdepodobne odrazíme od tohto tímu.

Tento tím mal hernú prevahu nad každým súperom, ale niektorým mal problém streliť gól, hoci mal vždy veľké množstvo šancí. Väčšinou strieľal z veľkej vzdialenosti, hoci mal voľný priestor a teda sa mohol presunúť bližšie k bráne, kde by sa šanca na strelenie gólu znásobila. Rozhodne je preto nutné zdokonaliť strieľanie na bránu v tomto zmysle.

Tím Jahodových princov sa inšpiroval hráčom Gang of Six. Hráča tohto tímu si zvolili, lebo dosahoval dobré výsledky. Rozhodli sa nerobiť zmeny v jadre hráča. Ich hlavným cieľom je oprava drobných nedostatkov a implementácia rozširujúcich vlastností, ktoré by mali zdokonaliť celkový výsledok. Ciele, ktoré si vymedzili, sú opísané ďalej.

Dopracovanie algoritmu vylučovania premenných

Problematika zahrávania autových situácii

V dokumentácii v zimnom semestri opisujú, že túto problematiku budú riešiť, v ich ďalších dokumentoch sme však o tom zmienku nenašli. Túto situáciu sa rozhodli riešiť, keďže hráč, ktorého prebrali, vhadzovanie lopty vôbec neriešil a dochádzalo k stratám lopty, čo bolo závažné najmä v obrane. Podľa nich je nájsť túto chybu veľmi náročné, lebo veľmi ťažko odsimulovať, kedy k nej dochádza.

Spôsoby zdokonalenia brankára

Tím vyladil nahrávania brankára. V prípade, že sú všetci hráči obsadení, nahrá ich brankár oproti predchádzajúcemu presnejšie a bezpečnejšie. Aj tak sa však niekedy stáva, že brankár nahrá protihráčovi pre hodnotu fitness. Upravili aj vybiehanie brankára a jeho návrat späť do bránkoviska, chytanie striel smerujúcih popri bránkovisku a držanie defenzívnej pozície.

Zavedenie a implementovanie kouča

Kouča sa autori rozhodli zaviesť preto, aby hráči nemuseli analyzovať súperových hráčov. Kouč vidí nezašumene celé ihrisko. Podarilo sa im rozbehať kouča tímu Gang of Six a UvaTrilearn. V robotickom futbale existujú heterogénni (nejaké vlastnosti majú potlačené, iné zplepšené) a homogénni hráči (všetky vlastnosti majú rovnaké). Viac sa oplatia heterogénni hráči. Tréner má za úlohu na začiatku zistiť atribúty hráčov a na základe toho ich zatriediť do rol. Roly:

- obranca
- stredopoliar
- stredný útočník
- útočiaci krídelník

Každá z týchto rol má pritom iné atribúty. Napríklad stredopoliar prijíma loptu od stredopoliarov, musí sa s ňou dostať ku bránke a kopnúť gól, preto by mal vedieť vyvinúť väčšiu rýchlosť (zrýchlenie) a mal by mať veľkú silu kopu. Takisto musí byť obratný, aby ho obrancovia ľahko neobrali o loptu. Tieto vlastnosti môžu byť zvýraznené na úkor presnosti prihrávky a takisto výdrže.

Implementácia zmeny formácií počas zápasu

Cieľom zmeny formácií počas zápasu je adekvátnejšie a pružnejšie reagovať na zmeny situácií. Túto funkčnosť sa rozhodli autori realizovať pomocou prebehu stredného hráča. Keď tím útočí, stredný hrač prebehne na útočnu polovicu, keď bráni, prebehne na polovicu obrannú.

Algoritmus eliminácie premenných

Algoritmus je založený na koordinačných grafoch. Cieľom je dosiahnuť, aby sa hráč rozhodoval tak, aby maximalizoval spoločnú výnosovú funkciu a teda robil to najlepšie pre tím. Autori nemenili cieľ tohto algoritmu, iba ho trošku upravili. Algoritmus je založený na výnosových funkciách. Napríklad zastavenie súperovej funkcie.

Zhodnotenie

Tím Jahodoví princovia je najpravdepodobnejším základom nášho hráča. V prípade pokračovania v hráčovi navrhujeme:

- skúsiť vyriešiť autové situácie, lebo tam zbytočne dochádza k strate lopty
- vylepšiť hráča, aby miesto strely z väčšej vzdialenosti išiel bližšie k bráne
- lepšie prispôsobovanie hráčov pri útoku
- urobiť diagramy UML štruktúry hráča
- skúsiť sa pozrieť na algoritmus eliminácie premenných

2.2.6 Loptoši

Tím Loptoši [7] si ako základ zvolil hráča predchádzajúceho projektu FIITMedia. Vo fáze prototypovania sa rozhodli zmeniť základ hráča a nadviazali na diplomový projekt FiitBA. Prínos hráča FiitBA je implementovanie vrstevnicovej mapy na rozhodovanie hráča a na reprezentáciu informácií o svete. Loptoši ako prvé vylepšili architektúru a prehľadnosť kódu. Implementovali návrhové vzory, ktoré sprehľadnili a zlepšili čitateľnosť kódu. Upravili niekoľko obsiahlych funkcií na viacero kratších. Odstránili duplicitný kód a abstraktné triedy SkillsInterface a AbilitiesInterface. Ďalej odstránili dedenie tried z Main od triedy Module. Ďalej nasleduje opis zmen, ktoré autori tímu navrhli a implementovali.

Vedúci tímu

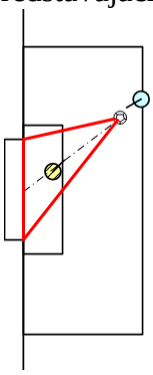
Úlohou vedúceho tímu je sledovanie okolia, aby mal čo najpresnejšie informácie o dianí na ihrisku a na základe týchto informácii mohol riadiť tím napr. zmenou formácie. Ako kandidát na vedúceho sa určil ten hráč, ktorý je vzdialený minimálne dvanásť metrov od lopty (autori vykonali simulácie troch odlišných metód určenia vedúceho). Tím Loptoši implementoval komunikačný protokol, pomocou ktorého si hráči medzi sebou vymieňajú správy o aktuálnom vedúcom tímu aj o jeho zmene. Protokol je navrhnutý tak, aby sa nestalo, že niektorý z hráčov má ešte starú informáciu alebo je viac vedúcich v jednom čase. Proces zmeny vedúceho je takýto:

1. Súčasný vedúci sa priblíži ku kandidátovi na vedúceho.
2. Nastaví si funkciu AttentionTo na kandidáta.
3. Počká určitý čas a na základe odpovede sa rozhodnutie prijme alebo odmietne.

Problém zmeny vedúceho na vzdialeného hráča sa vyriešil pomocou medzihráča, ktorý sa stane na určitý čas vedúcim a podá tento post tomu vzdialenému hráčovi.

Brankár

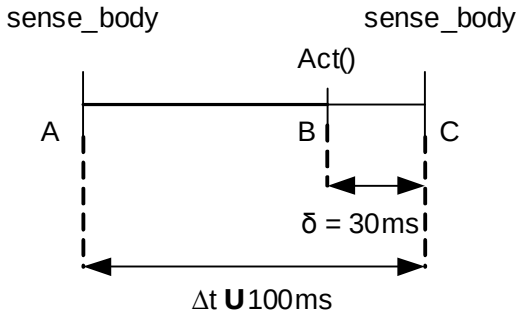
Tím sa v pôvodnom návrhu nezaoberal vylepšením samotného brankára. Avšak zistili, že brankár má výrazné nedostatky, napr. nevidí loptu, zle chytá strely a nedrží defenzívnu pozíciu. Nové riešenie, aby brankár videl loptu, spočíva v tom, že brankár, keď určitý počet taktov (konkrétne dvanásť) nevie, kde sa lopta nachádza, začne sa obzerať okolo seba a hľadať ju. Je zbytočné, aby išiel najprv na svoju domovskú pozíciu. Nová implementácia chytania lopty je taká, že hráč sa rozhodne loptu chytiť, ak vzdialenosť od lopty je parameter servra `catchable_area_l * k`, kde `k` je konštanta z intervalu <0, 1>. Ďalej autori vytvorili funkciu na určenie, či sa lopta môže chytiť o jeden alebo dva takty. Tento nový spôsob chytania lopty významne zlepšil uvedené problémy. Držanie defenzívnej pozície brankára je vyriešené tak, aby sa brankár nachádzal na ťažnici spájajúcej vrchol predstavujúci loptu s jeho protiľahlou stranou.



Obr. 8. Príklad dodržania defenzívnej pozície brankára tímu Loptoši

Synchronizácia

Pri skúmaní sa prišlo na problém synchronizácie hráča so servrom. Hráč veľmi často v nejakom takte vykonal akciu pred tým, ako dostal vizuálnu informáciu. Riešenie tohto problému bolo nastavenie času tesne pred koncom taktu, keď hráč vykonal danú akciu (**Obr. 9**).



Obr. 9. Znázornenie časovej synchronizácie hráča tímu Loptoši so servrom

Po vykonaní tejto úpravy autori zaznamenali veľké zlepšenie vo volaní funkcie `Act()`, ktorá zabezpečuje konanie hráča.

Situácie

Tím implementoval nové rozhranie, pomocou ktorého sa môžu doimplementovať nové situácie. Funkcia `LPT_processSituation` slúži na vykonávanie situácií

na základe aktuálnej situácie, ktorá je uložená v parametri `LPT_CurrentSituation`. Pre každú úlohu existuje zvláštny kód. Loptoši implementovali aj obrannú situáciu, v ktorej sa obranca z druhej strany postaví k vzdialenej žrdi, aby zabránil gólu v prípade, že by nasledovala prihrávka popred bránku na druhú stranu.

2.2.7 Sklo

Tím sklo [8] dobre analyzoval vtedajší stav systému, hráčov a predchádzajúce vylepšenia, ktoré mu poskytli odrazový mostík do ďalšej práce. Ako základ si autori zvolili zdrojový kód ich predchodcov – tímu Stjupit Dox. Pôvodne chceli použiť zdrojové kódy tímu Deravá kopačka, ale napokon sa im zdal Stjupit Dox vhodnejší pre prehľadnosť kódu, oddelenie hráča od vnútorného sveta a zmeny, ktoré by museli spraviť v tíme Deravá kopačka, by boli väčšie ako pri tomto tíme. Ďalej opíšeme hlavné vylepšenia:

Hráč

- Zmenený spôsob ukladania informácií o spoluhráčoch a súperoch vo vnútornom modeli sveta.
- Rozšírenie vnútorného modelu o možnosti na prihrávanie. Informácie o možných cieľoch prihrávky sa prenášajú zvukovou správou.
- Využitie heterotypov.
- Predpovedanie pozície lopty kvôli zníženiu času výpočtu.

Zvuková komunikácia

- Je založená na existujúcej komunikácií tímu Stjupit Dox.
- Pribudla tzv. urgentná správa.

Špeciálne taktické funkcie

- Dynamické určovanie cieľa hráča driblujúceho s loptou. Berie do úvahy rozloženie súperov.
- Výber miesta, kam hráč kopne loptu, keď už nemôže driblovať.
- Plánovanie výkopu.

Špecializácia taktiky hráčov

- Špecializácia hráčov na útočníka, stredopoliara a obrancu, pričom každý má svoju vlastnú taktiku.
- Hráč volí taktiku za behu a rozdeľuje stav s loptou a bez lopty.
- Šetrenie energie hráča, ak nemusí bežať naplno.

Brankár

- Vylepšený spôsob výkopu – brankár najprv vyhodnotí situáciu na ihrisku, zvolí pre seba najlepšiu pozíciu a až potom vykopne

Kouč

- Sleduje hráčov súpera a zisťuje, aké majú heterotypy. Následne túto informáciu odovzdáva svojmu tímu.

- Je to pomerne zložitá úloha, ale podarilo sa im implementovať rozpoznávanie niektorých heterotypov, a to najmä: rýchlosť, zrýchlenie, rýchlosť otáčania.
- Veľký problém robil šum.

V závere sa hovorí o tom, že najužitočnejší sú rýchli a vytrvalí hráči, preto voľba takýchto heterotypov je výhodná.

Odporúčané vylepšenia

- Plánovanie postupností akcií hráča.
- Predpoklad správania sa a umiestenia hráčov na základe ich umiestnenia vo formácii a stavu hry.
- Lepšia práca s údajmi o formácii.
- Koordinácia pri útoku.
- Uvoľňovanie spoluhráčov pri výkope a počas hry.

2.2.8 FIITMedia

Domáci tím FC Farmári spravil neurónovú sieť na vyhodnotenie vhodnosti prihrávky. Sieť sa inicializovala zo súboru s formátom riadka: d1 d2 d3 y, kde d1 je vziadelosť od cieľového hráča prihrávky, d2 je vzdialenosť protihráča v smere prihrávky, d3 je kolmá vzdialenosť protihráča a y určuje, či je prihrávka úspešná. Pri vyhodnocovaní vkladá hráč na vstup siete popri spoluháčovi polohy všetkých protihráčov, ktorých vidí. Prihrávku má hráč uskutočniť, len ak je pravdepodobnosť pre každý vstup vyššia než medzná hodnota. Autori FIITMedie [9] však zistili, že to tak nie je a hráč prihráva takmer vždy, keď nájde spoluhráča vzdialeného 5 až 30 metrov, ktorý je k bráne bližšie než on. Dokonca sa často prihrávalo aj cez protihráčov.

FIITMedia nahradila pre nedostatočnú funkčnosť a ťažkosť trénovania pôvodnú neurónovú sieť vlastnou.

Prihrávanie, kopanie

Autori FIITMedie pridali k vstupom neurónovej siete na vyhodnocovanie úspešnosti prihrávok parametre štýl kopania (na miesto alebo do behu) a stranu, z ktorej je postavený súper. Do vektora neurónovej siete vstupujú len určití spoluhráči, ktorý sa určujú algoritmom na zistenie okolia ohrozenia súpera. Okrem toho zjednodušili trénovanie prihrávok.

FC Farmári nedokázali driblovať dostatočne dlho na naviazanie súperových hráčov. FIITMedia to rieši zavedením tzv. nakopávaného driblovania. Hráč si v prípade dostatočne veľkého voľného priestoru nakopne loptu pred seba. V algoritme kopania autori tiež našli a odstránili daktoré chyby, keď hráč kopal loptu iným smerom, než má. Trénovaniu prihrávok a kopania venovali autori najviac času.

Oprava chýb v zdrojovom texte a implementácia neurónovej siete tvoria základ práce tímu FIITMedia. V dokumentácií sú spomenuté ďalšie zaujímavé nápady, ktoré autori realizovať nestihli.

Viac neurónových sietí

FIITMedia navrhla použitie viacerých neurónových sietí. Každá sieť by bola iná, a preto aj pri rovnakých vstupoch by vznikli iné výsledky. Hráč by mal možnosť použiť všetky siete a na základe presného algoritmu by sa rozhodol, ktorú bude považovať za smerodajnú. Každá sieť by sa natrénovala na inú formáciu súpera, celkove šesť. Viac

než šesť formácií videli autori ako neschodné pre nákladnosť a neefektívnosť. Ako alternatívu uvažovali s vytvorením neurónových sietí pre každú pozíciu hráča v tíme. Každý hráč by si podľa svojej pozície vyberal neurónovú sieť len z určitej množiny.

Navrhnuté algoritmy na výber vhodnej siete

- *Na základe informácii od kouča*

  Kouč posiela informácie len každých tristo taktov. Ak sa pomedzi rapídne zmení pozícia súpera, použitá sieť môže byť veľmi nevhodná.
- *Na základe rozhodnutia sa hráča*
  - *Pseudonáhodné rozdelenie*

    Hráč vyberie neurónovú sieť. Ak mu určí, že má vykonať prihrávku, vykoná ju. Podľa (ne)úspechu hráč zvýši/zníži dôveryhodnosť danej siete.
  - *Adaptívna kombinácia lokálnych expertov*

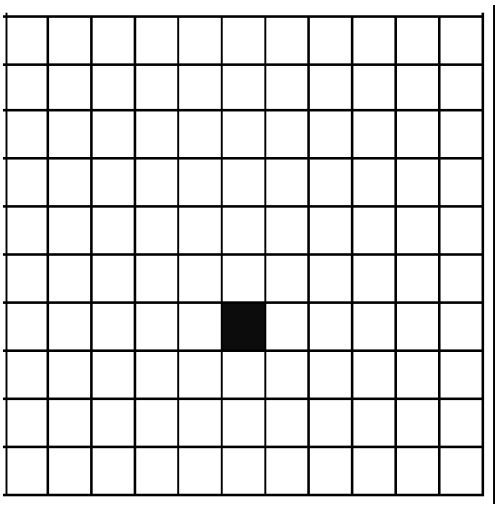
    Myšlienkou je mať nad danými sieťami rozhodovaciu sieť. Tá by sa učila pomocou spätného šírenia chýb (error backpropagation).

Akcie

Ďalším nápadom bolo pridanie akcií, t. j. postupností krokov, ktoré majú vykonať viacerí hráči. Hráč najbližšie k lopte by rozoznal akciu a oboznámil o tom zvukovou správou ostatných hráčov. Určil by pritom aj účastníkov akcie. Na rozoznanie akcie by sa používali vzory resp. masky. Každá maska by určovala rozmiestenie spoluhráčov, voľné miesto (tam nesmú byť protihráči, aby nemohli prerušiť prihrávku) a pozíciu brankára. Každý hráč prijímajúci prihrávku počas akcie by mohol v prípade neúspechu akciu zrušiť alebo zmeniť. Jendotlivé akcie by boli jednoduché, zložitejšia akcia by vznikla kombináciou jednoduchých (akcie by boli hierarchické).

Navrhovaný algoritmus porovnávania vzorov

Pred porovnaním si hráč rozdelí ihrisko na štvorce, čím vznikne mriežka. V nej bude hľadieť smerom nahor a bude sa nachádzať v jednej tretine odspodu. Do mriežky doplní zo svojich informácií o svete spoluhráčov. Mriežku potom prirovná k uloženým vzorom. Vyhovujú len vzory, pri ktorých sa všetky vyplnené pozície kryjú s vytvorenou mriežkou. Pre každý vzor existujú aj iné mriežky, určujúce prítomnosť súperových hráčov. Z tohto sa vyberú možné situácie. Navyše musia byť splnené ďalšie podmienky: pozícia braknára, pozícia v ofsajde a pod. Vyhovujúce situácie sa ohodnotia a vyberie sa najvhodnejšia.



Obr. 10. Umiestenie hráča v mriežke pri porovnávaní vzorov – návrh tímu FIITMedia

Libero

Libero je „futbalový obranca voľne zasahujúci do hry po celom ihrisku“. Nemá pevné miesto vo formácii, vyberá si pozíciu sám podľa vývoja hry. FIITMedia navrhla zakomponovanie libera do tímu. Pri ohrození by sa stiahol do obrany, pri útoku by pomáhal. Aby sa nevyčerpal behaním, obranné a útočné libero by boli dvaja rôzni hráči. Ich heterotyp by mal rýchlejšie dopĺňanie výdrže (staminy).

Štatistická analyáza rozloženia súperových hráčov

Návrh bol nechať kouča vyhodnocovať pravdepodobnosť výskytu súperových hráčov na rôznych miestach ihriska záplavovým alebo výškovým algoritmom. Dakoľkokrát za zápas by kouč informoval hráčov o pravdepodobných voľných miestach. Hráči by voľné miesta využívali ako miesta prihrávok.

Prihrávka obsadeným hráčom

Hráč beží s loptou vpred, no pred sebou má len protiháčov. Musí odkopnúť, ináč o loptu takmer isto príde. Zbadá spoluhráča na krídle, ten je však tiež obsadený. Zakričí mu, nech beží vpred, a kopne loptu na vhodné miesto záchytu prihrávky.

FM Farmári prihrávky do behu vôbec nevyužívali.

Zhodnotenie

FIITMedia priniesla do hráča prepracovanú neurónovú sieť s možnosťou jednoduchého trénovania. Vďaka tomu sa zlepšili prihrávky tvoriace základ pri (simulovanom) futbale. Mnohé z nahvrhovaných vlastností hráča nestihli zakomponovať, no pre náš tím môžu byť vhodným zdrojom nápadov.

2.2.9 UTTP

UTTP [10] bol tím pôsobiaci na fakulte v roku 2007. Ich hlavnou snahou bolo zlepšiť kvalitu pôvodného hráča tímu Loptoši, z ktorého vychádzali. Kvôli zvýšeniu kvality pristúpili k modularizácii správania, refaktorizácii kódu, dokumentovaní správania a dopĺňaniu komentárov. V projekte analyzovali správanie hráčov, od najvyššej až po základnú úroveň, ktorá zahŕňala komunikáciu so servrom.

Navrhli spôsob modularizácie správania, pomocou ktorého je možné rôzne druhy správania separovať a následne ich využívať v iných druhoch správania alebo v pôvodných triedach. Ako základ slúži trieda `ModuleManager`, ktorá poskytuje správu a rozhranie pre prácu s jednotlivými modulmi. Ďalej vytvorili rozhranie `ModuleInterface`, ktoré musia implementovať všetky vytvorené druhy správania a triedu `DataStorage` slúžiacu ako centralizované úložisko dát, ktoré potrebujú jednotlivé druhy správania. Nakoniec vyextraholi samotné druhy správania.

Hráč tímu UTTP obsahuje niekoľko modularizovaných druhom správania. Vytváranie nového modulu realizovali na základe analýzy zdrojových kódov hráča a identifikácie druhov správania.

Tím sa v podstatnej miere venoval oprave chýb, ktoré našli v pôvodnom kóde. Pôvodný hráč nebol kompilovateľný a po určitom čase vždy padol. Projekt následne konvertovali do prostredia Visual Studio 2005. Ďalej hráča optimalizovali aj z hľadiska veľkosti. Objavili a odstránili nepoužívané funkcie, prípadne časti kódu, ktoré nedávali požadované výsledky.

Na účely testovania tím vytvoril vlastný framework. Slúži na testovanie RoboCupového hráča. Aby ho mohli použiť ďalšie tímy, je nutné doladiť ho (ukončenie testu, kvalitné štatistiky, zber a vyhodnocovanie údajov z testov, a podobne).

2.2.10 Zhodnotenie analýzy tímov

Po oboznámení sa so situáciou na fakulte i v zahraničí, sme sa rozhodli v rámci prototypu odskúšať viac ciest:

- Zdokumentovať hráča tímu Jahodoví princovia

  Dôkladné oboznámenie sa s hráčom tímu Jahodoví princovia je dlhodoboou úlohou pre všetkých členov. Aby nebolo v budúcnosti (inými tímami) nutné nazrieť do zdrojového kódu na pochopenie štruktúry hráča, pokúsime sa vytvoriť akúsi používateľsky prívetivú príručku s názornými diagramami.
- Zhodnotiť vhodnosť použitia hráča napísaného v Jave

  Či sa pokúsime o implementáciu v Jave my, alebo až ďalšie tímy, preskúmanie tejto oblasti bude určite prínosom pre fakultný RoboCup.
- Implementovať agresívne správanie podľa vzoru tímu Brainstormers

  Agresívne oberanie o loptu je novinkou vo svete RoboCupu. Ak sa nám ho podarí zahrnúť do hráča, môže sa úroveň domácej ligy posunúť bližšie k svetovej.

2.3 Možnosti logovania

Odhaľovanie chýb je súčasťou vývoja každého softvéru. V krátkosti uvádzame, aké možnosti nám v tejto oblasti poskytuje hráč tímu Jahodoví princovia.

Logger

V rámci implementovaného hráča tímu Jahodoví princovia existuje trieda `Logger`. Táto trieda umožňuje logovanie informácií na rôznych úrovniach abstrakcie. Všetky správy sú odovzdané logovacej metóde `log` s indikáciou úrovne jednotlivých správ. Ak je dané, že by sa mali logovať správy s úrovňou n, použijeme metódu `addLogLevel` alebo `addLogRange`, ktorá zaznamená správy úrovne n a ostatné zahodí. Toto nám umožní výpis len určitých správ, ktoré nás zaujímajú. Existuje jedna globálna premenná

`Log`, ktorú používajú ostatné triedy využívajúce triedu `Logger` a jej metódy. Inštancia triedy `Logger` je umiestnená v súbore `Logger.cpp` a jej názov je `Log`. Všetky triedy, ktoré ju chcú používať, by si mali vytvoriť referenciu pridaním riadka `extern Logger Log;` a potom môžu používať jej metódy pomocou `Log.log( ... )`. Okrem toho trieda `Logger` obsahuje aj časovač, ktorý umožní výpis času od posledného reštartu časovača.

SituationsLog

Táto trieda slúži na zaznamenávanie situácií, ktoré nastali počas zápasu. Tvorcom je tím GangOfSix, autori programu SituationMonitor umožňujúceho graficky znázorniť jednotlivé situácie, ktoré sú zaznamenané v súboroch XML. Pre každého hráča sa generuje samostatný súbor XML v predpísanej forme. Máme tri typy objektov, ktoré sa zaznamenávajú: bod, čiara a kruh. Bod predstavuje hráčov v okolí. Sú dva typy bodov, a to spoluhráči a protihráči, ktorých odlišujeme farbou bodu. Čiary predstavujú vektory, ktoré nám ohraničujú priestor napr. na vykonanie prihrávky. Kruh predstavuje rádius, v ktorom je hráč schopný prebrať prihrávku. V programe SituationMonitor si môžeme otvoriť jednotlivé súbory XML. Ak sa v tom súbore nachádzajú nejaké záznamy, tak nám program ponúkne výber situácií, ktoré boli zaznamenané. Po ich výbere sa nám vykreslí konkrétna situácia.

2.4 Zbežný pohľad na štruktúru hráča Jahodových princov

Cieľom tejto kapitoly je podať prvé priblíženie pripravovaného dokumentu na jednoduché a rýchle vysvetlenie koncepcie, organizácie a hlavných tried robotického hráča tímu Jahodoví princovia naprogramovaného v programovacom jazyku C++. Dokument bude slúžiť najmä budúcim tímom robotickeho futbalu, ktorí sa rozhodnú pokračovať v implementovaní tohto hráča a nemajú dostatočné skúsenosti s programovacím jazykom C++.

2.4.1 Organizácia kódu

Zdrojové kódy robotického hráča obsahujú dva druhy zdrojových súborov. Prvým je typ súboru s príponou `.h` a druhý s príponou `.cpp`. Na čo slúžia a čo obsahujú jednotlivé súbory v týchto zdrojových kódov, si teraz vysvetlíme.

Súbory s príponou .h – obsahuje definíciu triedy. Nachádzajú sa v nej atribúty tejto triedy, metódy a konštruktory. Vo väčsine prípadov obsahuje iba prázdne metódy (definuje sa iba signatúra metódy, metódy nemajú telo). Tieto prázdne metódy potom implementuje súbor s príponou `.cpp`. Súbory s príponou `.h` môžu obsahovať aj viac tried, nie len jednu, ako sa používa vo väčsine iných objektovo-orientovaných jazykoch.

Nasleduje príklad krátkeho úrivku z triedy `BasicPlayer`. Červenou farbou sú zvýraznené poznámky k významu častí kódu.

```
#ifndef _BASICPLAYER_
#define _BASICPLAYER_

#include "ActHandler.h"  // includovanie potrebných súborov

#include "fuzzyobj.h"     // includovanie potrebných súborov

extern Logger Log;

/*! Tu sa nachádza komentár k danej triede*/
```

```
class BasicPlayer
{
//jednotlivé atribúty (premenné) tejto triedy
protected:
  ActHandler      *ACT; /*!< ActHandler to which commands can be
sent        */
  WorldModel      *WM;  /*!< WorldModel that contains information
of world   */
  ServerSettings  *SS;  /*!< All parameters used by the server
*/
  PlayerSettings  *PS;  /*!< All parameters used for the player
*/

  ////////////////////// LOW-LEVEL SKILLS//////////////////////////

//jednotlivé metódy (funkcie) tejto triedy
  SoccerCommand   alignNeckWithBody       (
);
  SoccerCommand   turnBodyToPoint         ( VecPosition   pos,
                                            int           iPos = 1
);

} ;

#endif
```

Súbory s príponou .cpp – obsahujú implementácie jednotlivých metód, ktoré boli zadefinované v súbore s príponou `.h`. Taktiež obsahujú konštruktor.

```
#include "BasicPlayer.h"   //includovanie potrebných súborov
#include "Parse.h"         //includovanie potrebných súborov

/********************** LOW-LEVEL
SKILLS*********************************/

/*! Tu sa nachádza komentár k danej metóde (funkcie)*/
//implementácia metódy, ktorá je zadefin. v súbore s príponou .h
SoccerCommand BasicPlayer::alignNeckWithBody( )
{
  return SoccerCommand( CMD_TURNNECK,
                        WM->getAgentBodyAngleRelToNeck( ) );
}

/*! Tu sa nachádza komentár k danej metóde (funkcie)*/
//implementácia metódy, ktorá je zadefin. v súbore s príponou .h
SoccerCommand BasicPlayer::turnBodyToPoint( VecPosition pos, int
iCycles )
{
  VecPosition posGlobal = WM->predictAgentPos(iCycles, 0);
  AngDeg angTurn        = (pos - posGlobal).getDirection();
  angTurn              -= WM->getAgentGlobalBodyAngle();
  angTurn               = VecPosition::normalizeAngle( angTurn );
  angTurn               = WM->getAngleForTurn( angTurn,
                                               WM->getAgentSpeed(),
                                               WM->getAgentObjectType()
);

  return SoccerCommand( CMD_TURN, angTurn );
}
```

Všetky zdrojové súbory sa nachádzajú v jednom zdrojovom adresári, čo nie je ideálne. Buduce tímy, ktoré preberú tohto hráča, by sa mohli zamerať na lepšie zorganizovanie kódu, aby všetko nebolo pokope.

2.4.2 Najzákladnejšie triedy

V tejto kapitole sú opísané najzákladnejšie triedy, predstavujúce hrubý pohľad na štruktúru hráča. Pri každej triede je uvedených len pár metód ako príklad.

`BasicPlayer` – toto je zakladná trieda hráča. Obsahuje základné metódy (funkcie), ktoré hráč môže urobiť. Môžeme ich rozdeliť na tri úrovne. Sú to:

- *Low level skills* – sú to akoby najnižšie funkcie, ktoré hráč ovláda. Ide napríklad o funkcie, ktoré umožnujú otočiť telo k nejakému bodu, komunikovať s ostatnými hráčmi a počúvať ich, nájsť loptu ap.
- *Intermediate level skills* – ide o funkcie, ktoré napríklad umožňujú driblovať s loptou rýchlejšie alebo pomalšie, natočiť telo k danému objektu, pohnúť sa k danej pozícii po čiare, kopnúť do lopty ap.
- *High level skills* – ide o funkcie, ktoré umožňujú napríklad driblovať s loptou, dať prihrávku dopredu a do voľného priestoru, prehrať súpera ap.

Väčšina týchto metód vracia objekt typu `SoccerComand` (nachádza sa v súbore `SoccerTypes.h`). `SoccerComand` obsahuje všetky informácie o príkaze, ktoré sa majú poslať serveru. Tieto príkazy sú nezávislé od implementácie servera a predtým, ako sa pošlú, sa skovertujú na reťazec znakov.

`Player` – táto trieda dedí od `BasicPlayer`. V jednoduchosti sa dá povedať, že táto trieda obsahuje komplexnejšie metódy na výber nasledujúcej akcie. Obsahuje metódu `mainLoop`, kde sa rozhoduje, ktorú akciu najbližšie vykoná.

`WorldModel` – obsahuje funkcie, ktoré umožňujú vrátiť informácie o súčasnom alebo búducom stave sveta (futbalové ihrisko). Zároveň sa stará aj o obnovenie informácií. Tieto informácie môžeme rozdeliť do viacerých kategórii:

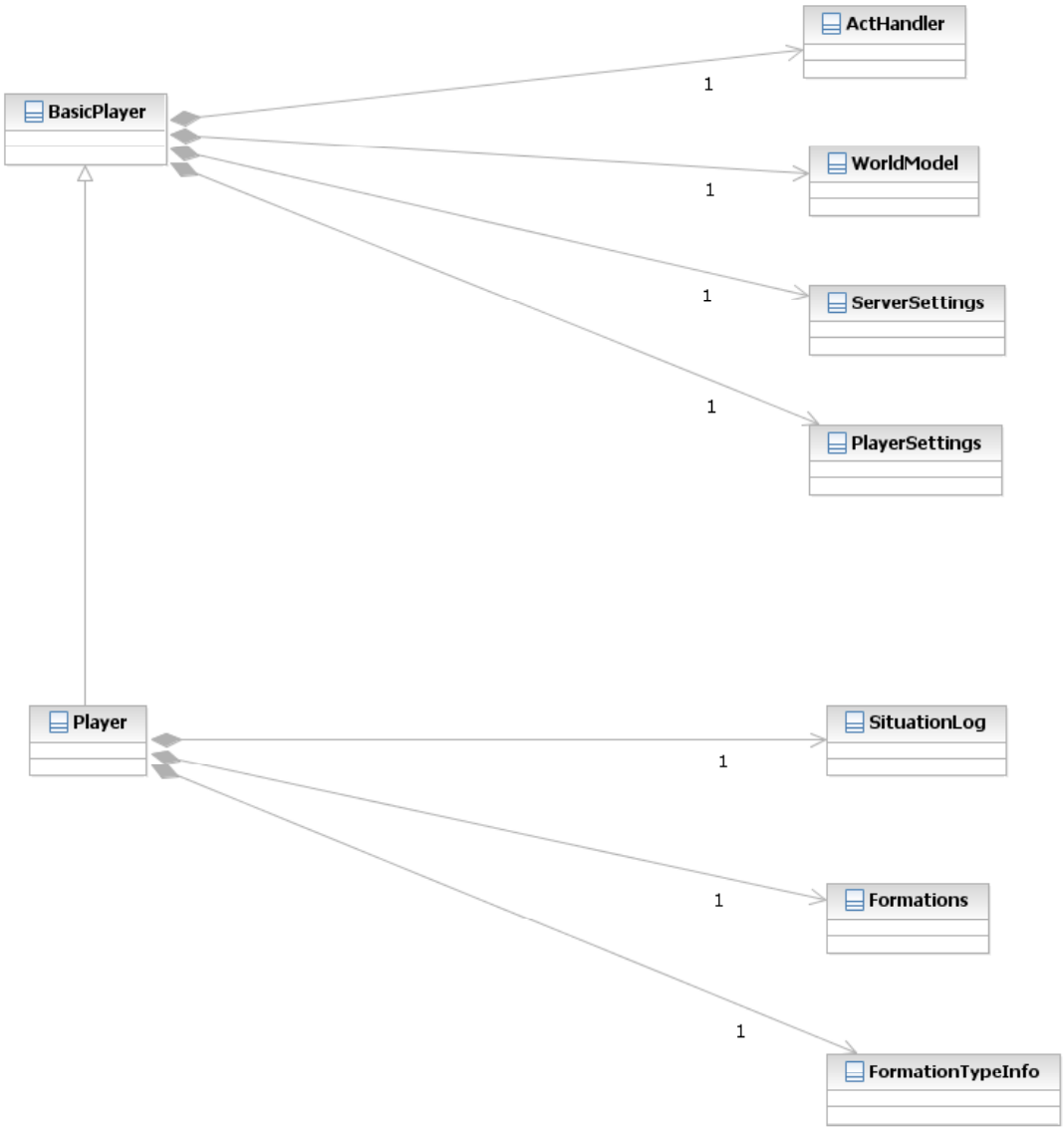
1. Enviromentálne (informácie o serveri)
2. Informácie o zápase (základné informácie o súčasnom stave zápasu)
3. Informácie o objektoch na ihrisku
4. Informácie o akciách, ktoré hráč urobil

`ActHandler` – posiela jednotlivé príkazy na server. `ActHandler` obsahuje rad týchto príkazov. Keď príde signál, prekonvertujú sa tieto príkazy na textové reťazce a pošlú sa na server.

`SituationLog` – táto trieda vytvára logy pre situácie. Obsahuje napríklad metódy, ktoré umožňujú pridávať objetky, ktoré maju byť logované (kapitola 2.3).

`ServerSettings` – táto trieda obsahuje všetky parametre Soccer servera. Ide napríklad o paramatre pre hráča ako jeho váha, maximálna rýchlosť, parametre pre únavu, parametre pre loptu, kouča a i. Všetky atribúty majú nastavenú nejakú konkrétnu hodnotu.

`PlayerSettings` – táto trieda obsahuje atribúty, ktoré sú dôležité pre hráča vykonať ďalšiu akciu. Atribút tejto triedy je napríklad rýchlosť lopty pri rýchlej nahrávke. Všetky atribúty majú nastavenú nejakú konkrétnu hodnotu.



Obr. 11. Základný diagram tried hráča Jahodových princov

2.5 Testovanie nárokov hráča tímu DAInamite

Našou úlohou bolo otestovať javovú implementáciu hráča a porovnať výkonnosť s implementáciou iného tímu v C++ na dvoch rôznych strojoch. Prvé testy prebiehali tak, že všetky potrebné súčasti bežali na jednom stroji. To znamená, že RoboCup server, tím Squirell Squadron a aj javový tím DAInamite bežali na jednom počítači naraz. Výsledky boli nasledovné:

Intel Core II Duo E6750 @ 2.66GHz, 2.67GHz, 2 GB RAM – Desktop

Vykonali sme šesť simulácií, každá prebehla bezchybne až do konca polčasu – 1 000 simulačných taktov. Vykonanie 1 000 taktov trvalo dve minúty aj 40 sekúnd,

z čoho vyplýva, že desať taktov zodpovedá 1,6 sekundy reálneho času. Tento čas bol rovnaký pre všetky simulácie.

Zaťaženie stroja, čo sa týka nárokov na procesor, bolo pomerne nízke – zaznamenali sme približne 5% nárast zaťaženia procesora počas behu simulácie oproti pokojovému stavu.

Intel Core II T7200 @ 2.00GHz, 996MHz, 2 GB RAM – Notebook

Rovnako sme vykonali šesť simulácií, pričom väčšina nebola úspešne ukončená. Simulácie prebiehali nasledovne:

1. 446 taktov za 1,17 minúty
2. 674 taktov za 1,49 minúty
3. 104 taktov za 0,18 minúty
4. 360 taktov za 0,58 minúty
5. 1 000 taktov za 2,40 minúty
6. 1 000 taktov za 2,40 minúty

Spolu 2 584 taktov za 422 sekúnd, z čoho vyplýva priemerný čas 1,63 sekundy na desať taktov. Avšak simulácie vo väčšine prípadov neprebehli do konca a nepodarilo sa nám zistiť dôvod, prečo to tak bolo.

Zaťaženie procesorov kolísalo okolo hodnoty +40% počas behu simulácie. Nároky na pamäť boli na oboch strojoch približne rovnaké:

- RoboCup server približne 11 MB
- jedenásť hráčov tímu Squirell Squadron 10 MB
- jedenásť hráčov v Jave + monitor tímu DAInamite 603 MB

Spoločná záťaž bola približne 625 MB pamäte.

Viac počítačov

Ďalšie testy prebiehali s použitím silnejšieho stroja ako servera, na ktorom bol spustený RoboCup server a javový tím. Na slabšom stroji bolo spustených len jedenásť hráčov tímu Squirell Squadron. Simulácia na serveri prebiehala približne rovnakým spôsobom ako pri prvých pokusoch (zaťaženie procesora +4% až +5%) a na slabšom stroji spotrebovalo jedenásť hráčov približne 15MB pamäte a vyťaženie procesora sa pohybovalo okolo +8%.

Zhodnotenie

Z predchádzajúceho jasne vyplýva, že implementácia v Jave má oveľa vyššie nároky na pamäť aj silu procesora ako C++ implementácia. Vyťaženie procesora ani v jednom prípade nebolo nad 50%. Čas 1,6 sekundy na desať taktov v oboch prípadoch hovorí o tom, že simulácia prebiehala celkovo pomalšie, ako by mala, ale nebolo to spôsobené nedostatkom výpočtovej sily.

3 ŠPECIFIKÁCIA A HRUBÝ NÁVRH

3.1 Agresívne správanie hráča

Jedným z cieľov nášho tímového snaženia je implementácia agresívneho správania do hráča tímu Jahodoví princovia na zlepšenie obranných schopností. Ako vzor slúži viacnásobný víťaz RoboCupu 2D – nemecký tím Brainstormers. Výsledkom práce bude navrhnutá, vytvorená a naučená neurónová sieť plus nevyhnutné rozhranie na jej použitie v zápasoch. Nasledujúce riadky obsahujú opis a vysvetlenie problému, ako aj návrh riešenia.

3.1.1 Agresívne správanie

Agresívnym správaním sa rozumie snaha hráča prerušiť rozohrávku súperiaceho tímu tak, aby zabránil vzniku útočnej situácie a – naopak – získal loptu na budovanie vlastného útoku. V praxi ide o obranné správanie: pristúpenie k hráčovi s loptou a jej odobratie povoleným spôsobom.

Väčšina tímov robotického futbalu rieši proces získania lopty naivným spôsobom: hráč najbližšie k protivníkovi s loptou sa snaží dostať na dosah lopty, aby ju mohol nahrať spoluhráčom. Táto stratégia často býva neúspešná, hlave ak súperiaci hráč je z tímu s dobre vyvinutou schopnosťou driblovať [5].

Získanie lopty je vysoko netriviálna úloha, pričom jej zložitosť závisí vo veľkej miere od správania súpera. Pri postupe klasickými programátorskými metódami môže nastať problém vytvorenia špecializovaného hráča, ktorý by fungoval dobre pri niektorých tímoch, ale pri iných vôbec. Okrem toho musí hráč pri procese získania lopty dbať aj na svoje postavenie a postavenie ostatných spoluhráčov, aby pri možnej strate lopty nedošlo k prečísleniu a súperiaci tím nezískal útočnú výhodu.

Riešenie, ktorým sa zaoberal viacnásobný víťaz RoboCupu 2D, nemecký tím Brainstormers, spočíva vo využití neurónových sietí, presnejšie metódy „reinforcement learning“. Reinforcement learning je metóda učenia sa vďaka interakcii s prostredím. Agent sa učí na základe výsledkov jeho predchádzajúcej činnosti. Nasledujúcu akciu vyberá z množiny predchádzajúcich výsledkov a nových možností, čiže postupuje metódou pokus-omyl. Učiacim signálom je numerická hodnota určujúca úspech akcie, pričom agent sa snaží vyberať v ďalších krokoch akcie, ktoré maximalizujú kumulovanú odmenu v priebehu času [12].

3.1.2 Modelovanie problému

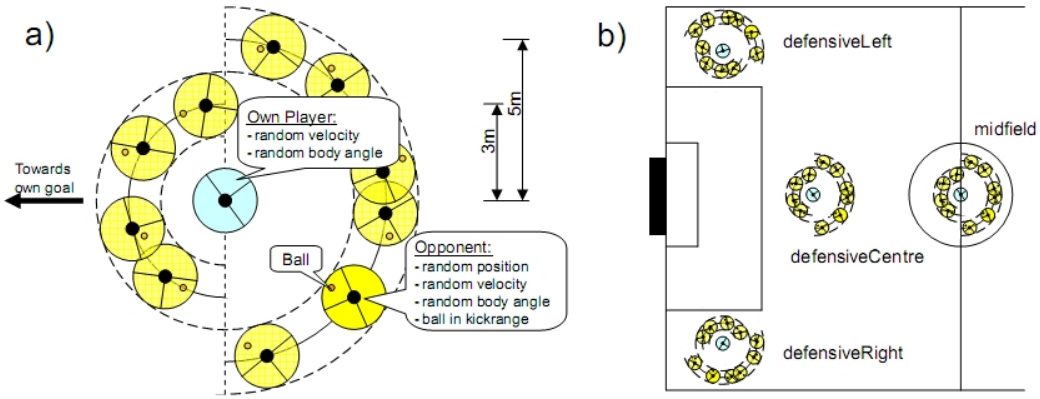
Stavový priestor robotického futbalu je mimoriadne rozsiahly a mení sa v každom takte hry. Z toho dôvodu je nutné určiť, ktoré parametre sú nutné na identifikáciu vhodnosti agresívneho správania. Tím Brainstormers sa rozhodol obmedziť problém na deväť dimenzií [5]:

- Vzdialenosť d medzi naším a súperiacim hráčom s loptou
- Rýchlosť nášho hráča (v_x a v_y zložky pohybu)
- Absolútna hodnota v_{opp} – rýchlosť súperiaceho hráča s loptou
- Pozícia lopty na mape (b_x a b_y)

- Natočenie tela hráča relatívne k súperovej pozícii (uhol α)
- Natočenie tela útočiaceho hráča vzhľadom na našu bránu (uhol β)
- Hodnota strategického uhla γ = ∠ GOM (kde G je pozícia našej bránky, O pozícia súpera a M pozícia nášho hráča)

Hráčovi je povolené použiť príkazy `dash(x)` a `turn(y)` z ohraničených intervalov tak, že existuje 76 vykonateľných akcií. Intervaly pre jednotlivé príkazy sú:

- $x \in \langle -100,100 \rangle$
- $y \in \langle -180°,180° \rangle$



Obr. 12. a) Vymedzený priestor v okolí hráča s vyznačenými príkladmi umiestnení súpera (žlté kruhy), b) tréningové regióny na mape [11]

Tréningové stavy, slúžiace na učenie hráča, boli modelované podľa nasledujúceho modelu. V okolí hráča si možno predstaviť dva polkruhy, jeden s polomerom 3 m, druhý 5 m. Polkruh s menším polomerom je orientovaný smerom k našej bránke, polkruh s väčším k súperovej. Do takto vymedzeného priestoru je umiestnený súperiaci hráč a lopta, pričom lopta je v blízkosti súpera a má nulovú rýchlosť. Rýchlosti hráča a súpera, ako aj ich vzájomné natočenie, sú zvolené náhodne.

Okrem toho tím Brainstormers vybral štyri tréningové regióny na mape. Jeden v strede ihriska, dva v jednotlivých rohoch našej polovice ihriska a posledný v strede našej časti ihriska. Pri návrhu týchto pozícií vychádzali z idey, kadiaľ súperiace tímy vedú útok (všeobecne stredom, alebo po krídlach plochy).

Navrhovaná sieť pozostáva z 28 neurónov pričom obsahuje jednu skrytú vrstvu (architektúra po vrstvách: 9-18-1). Ako aktivačná funkcia je použitá sigmoidálna funkcia a učiacim algoritmom je RPROP algoritmus (Resilient backpropagation). Za rýchlosť učenia α bola zvolená hodnota 1,0.

Chybová funkcia RPROP algoritmu je:

$$E = \sum (d_k - out_k)^2 + 10^{-\alpha} \sum w_{ij}^2$$

Pritom out_k je výstupom siete, d_k je požadovaným výstupom, w_{ij} je j-tou váhou neurónu i a α parametrom váh.

Pri trénovaní siete sa jednotlivé stavy nevymazávajú, ale prebieha ich pretrénovanie po 250 spustených inštanciách.

Dosiahnuté výsledky Brainstormers

Tím pri vyhodnocovaní výsledkov identifikoval päť možných koncových stavov:

1. *úspech* – Lopta sa dostala do hrateľnej oblasti nášho hráča, hráč obral súpera a môže nahrať voľnému spoluhráčovi.
2. *neúspech* – Neúspešná epizóda môže vzniknúť v dôsledku straty lopty hráčom, príp. keď lopta opustí hraciu plochu a pod.
3. *súper spanikári* – Prejavuje sa nezmyselným odkopnutím lopty súpera mimo svojho hracieho priestoru, často dopredu alebo do voľného priestoru. Takúto epizódu možno považovať aj za remízu.
4. *chyba* – Protihráč prešiel cez nášho atakujúceho hráča a je od neho vzdialený viac ako sedem metrov.
5. *time-out* – Počas trvania epizódy (35 krokov) nenastala ani jedna z vyššie uvedených situácií.

Tím dosiahol po naučení úspešnosť siete presahujúcu 80%, pričom množstvo chybových epizód sa držalo pod úrovňou 5%.

V priebehu klasických zápasov (6 000 taktov) mali hráči približne 66 epizód, v ktorých použili agresívne správanie. Pri konzervatívnom odhade úspechu 50% to znamená viac ako 30 získaní lopty, čím zabránili gólovým situáciám a – naopak – mohli prejsť do útoku.

3.1.3 Prevzatie správania

Náš tím nemá ambíciu navrhnúť vlastnú neurónovú sieť ani vlastný algoritmus, lebo to by sme z časových dôvodov nestíhali. Zameriame sa na preskúmanie voľne dostupného kódu tímu Brainstormes a jeho nasadenie do nášho hráča. V rámci prototypu nás zaujíma predovšetkým kompatibilita riešenia s kódom tímu Jahodoví princovia. Je potrebné zahrnúť do hráča neurónovú sieť tak, aby sme boli schopní trénovať ho na agresívne správanie.

3.2 Úroveň dostupného kódu tímu DAInamite

Súčasťou prototypu bude overenie vlastností voľne dostupnej verzie zahraničného tímu DAInamite napísaného v Jave. Nároky na hardvér sme už otestovali (kapitola 2.5), zaujíma nás predovšetkým, či sme schopní hráča použiť ako kostru pre náš vývoj. Voľne dostupná verzia neobsahuje vyššie správanie. Práve to je sféra, v ktorej by sme pokračovali. Vyššie správanie by sme prebrali z hráča tímu Jahodoví princovia (aj preto je potrebné, aby sme sa s ním podrobne oboznámili). Nemôžeme však predpokladať úspešné zvládnutie tejto úlohy, ak kód tímu DAInamite neposkytuje dobrý základ. Pod základom rozumieme funkcie nižšieho „správania“, ako je komunikácia so servrom, udržiavanie si modelu sveta, schopnosť plniť jednoduché úkony typu kopni, bež a pod. V prototype sa zameriame na postupné odskúšanie týchto funkcií pomocou trénera:

– *Kop*

  Zaujímajú nás možnosti kopnutia istým smerom, určitou silou, na konkrétnu pozíciu.

- *Beh*

  Aj pri behu je podstatná schopnosť bežať určitým smerom a na konkrétny bod. Zároveň je zaujímavé narábanie s výdržou (staminou). Beh je činnosť na viac taktov, preto je potrebná aj možnosť prerušiť ho v prípade potreby.

- *Nahrávka*

  V podstate ide o špeciálny prípad kopu na jednej strane (prihrávajúcim hráčom) a špeciálny prípad behu na druhej (cieľových hráčom pri prihrávke do behu).

- *Pohľad*

  Odskúšame pohľady hráča s rôznou šírkou a rôznym smerom. Overíme aj príjem zrakových správ a ich spracovanie.

- *Komunikácia*

  Pod komunikáciou chápeme výmenu krátkych správ medzi hráčmi. Popri schopnosti vyslania správy nás zaujíma aj možnosť počúvať iného hráča (príkaz `AttentionTo`).

- *Prehľad o dianí na ihrisku*

  Toto už patrí k stredným až vyšším schopnostiam hráča, no napriek tomu je existencia modelu sveta nutným predpokladom na pokračovanie v kóde.

Úroveň týchto funkcií nám poskytne prehľad o tom, či má význam pokračovať v tíme DAInamite. Ide totiž o základné funkcie, a ak nie sú dostatočne implementované, pravdepodobne by sme neboli schopní nahradiť ich vlastnými, čo by mohlo mať za následok neschopnosť nášho hráča zúčastniť sa turnaja.

POUŽITÁ LITERATÚRA

1. Oficiálna stránka RoboCupu.
   http://www.robocup.org
2. Amin Milani Fard et al.: *Nexus 2D 2008 Team Description*, IEEE Latin American Robotics Competitions - RoboCup Brazil Open 2008, Oct, 2008, Rio de Janeiro, Brazil.
   http://nexus.um.ac.ir/Nexus-2d-2008.pdf
3. Endert et al.: *DAInamite 2008 Team Description*. In Proceedings RoboCup 2008, Suzhou.
   http://www.dainamite.de/fileadmin/Dainamite-Dateien/Papers/DainamiteTDP2008.pdf
4. Marian Sebastian: *OXSY 2006 Team Description*, OXygen-SYstems laboratory, Str. Constantin Noica, Bl.5, Sc.C, Ap.36, C.P. 550169, Sibiu, ROMANIA.
   http://navid.alamati.googlepages.com/Oxsy2006TeamDescription.pdf
5. Riedmiller M. et al.: *Brainstormers 2D — Team Description 2008*, Institute of Cognitive Science, Universität Osnabruck.
   www.ni.uni-osnabrueck.de/fileadmin/user_upload/publications/riedmiller.gabel.trost.bs08tdp.pdf
6. Ladislav Borženský et al.: *RoboCup S – nové stratégie*, Bratislava: FIIT STU, 2008.
   http://labss2.fiit.stuba.sk/TeamProject/2007/team16is-si/dokumenty/odovzdane2/Implementacia.pdf
7. Peter Cséfalvay et al.: *RoboCup – nové stratégie*, Bratislava, FIIT STU, 2007.
   http://labss2.fiit.stuba.sk/TeamProject/2006/team08/public_html/docs/technicka_dokumentacia.doc
8. Serguei Gorbatchev et al.: *RoboCup – simulácia robotického futbalu*, Bratislava, FIIT STU, 2004.
   http://labss2.fiit.stuba.sk/TeamProject/2003/team08/download/doc/final04/v_software_final.pdf
9. Ľubomír Hromádka et al.: *RoboCup – nové stratégie*, Bratislava, FIIT STU, 2006.
   http://labss2.fiit.stuba.sk/TeamProject/2005/team02/DokumentaciaFinal.pdf
10. Peter Kajsa et al.: *RoboCup – nové stratégie*, Bratislava, FIIT STU, 2007.
    http://labss2.fiit.stuba.sk/TeamProject/2007/team06is-si/projektova_dokumnetacia-UTTP_a.doc
11. Riedmiller M. et al.: *Brainstormers 2D — A Case Study on Improving Defense Behavior in Soccer Simulation 2D*, Institute of Cognitive Science, Universität Osnabruck.
    www.ni.uos.de/fileadmin/user_upload/publications/gabel.riedml.trost.robocupsymposium08.pdf
12. Florentin Woergoetter and Bernd Porr: *Reinforcement learning*, Scholarpedia, 3(3):1448, 2008.
    http://www.scholarpedia.org/article/Reinforcement_learning