TECHNICKÁ UNIVERZITA V KOŠICIACH
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a umelej inteligencie

Robotic Football Team TUKE
(Záverečná práca)

Vypracovali:
Veronika Szabová
Adam Bugan
Michal Puheim
Tomáš Sabol
Dominika Fodorová

Akademický rok: 2009/2010
2. ročník Bc. štúdia
Odbor: Inteligentne systémy

OBSAH

Úvod

Obsah tohto reportu je založený na poznatkoch získaných počas trvania predmetu Aplikácie umelej inteligencia pri práci v simulovanom prostredí programu Webots od firmy Cyberbotics. Pracovali sme na regulátore robota futbalistu pre účely turnaja Robotstadium Contest, kde proti sebe na futbalovom ihrisku stoja na oboch stranách tímy simulovaných robotov Nao od firmy Aldebaran Robotics.

Tento súboj na simulátore je inšpirovaný medzinárodnou súťažou zvanou Robocup. Táto súťaž sa prvýkrát konala v meste Osaka v Japonsku v roku 1996. Odvtedy je hrávaná každý rok v rôznych krajinách sveta.

Dlhodobým výhľadom tejto súťaže je podporovať výskum a vzdelanosť v oblasti robotiky. Dokonca existuje ambícia do roku 2050 postaviť tím robotických futbalistov schopných poraziť tých najlepších ľudských protivníkov.

Cieľom projektu bolo vytvoriť vyššie spomínaný regulátor, zaradiť sa do súťaže a dostať sa do umistnenia v rebríčku na stránke www.robotstadium.org. Keďže turnaj nanešťastie neprebiehal tak, ako sme čakali, regulátor sme prevažne overovali na vlastných počítačoch.

Pri našich počinoch sme narazili na široké spektrum problémov, ktoré je potrebné v robotike riešiť. Od spracovania senzorických informácii, cez dizajn pohybov, až po tímovú spoluprácu v každom bode vyskakujú problémy, s ktorými sa robot musí vysporiadať. Zistili sme, že pre ohromnú komplexnosť riešených úloh je pochopiteľné, že aj napriek polstoročiu dynamického vývoja v odboroch robotiky a umelej inteligencie, stále neexistuje autonómny robot, ktorý by sa vyrovnal svojmu ľudskému protikladu.

Hoci o tomto projekte by bolo možné napísať veľmi rozsiahly článok, v nasledujúcom sa obmedzíme na to najdôležitejšie, s čím sme sa pri programovaní regulátora stretli.

1 Pohyb

Jednou z najdôležitejších častí prípravy robota na robofutbal je spracovanie a úprava pohybov. Rýchlosť, efektivita a stabilita sú najdôležitejšie sledované parametre. Po otestovaní defaultných pohybov poskytnutých aplikáciou Webots bola zistená relatívna pomalosť pohybov a dokonca istá miera zakrivenosti. Z toho dôvodu bolo potrebne získať nové pohyby.

Spôsoby získavania nových pohybov:
- Tvorba nových pomocou Motion Editora
- Úprava aktualných pohybov

Pri tvorbe nového pohybu sa nastavuje základny krok Step, ktorého základná hodnota je 40. Následne sa pridávaju jednotlivé pózy, ktoré obsahujú polohy servomotorov, ktoré medzi pózami menia svoju hodnotu. Alternatívny prístup je zameraný na zefektívňovanie už existujúcich pohybov. Dve najrýchlejšie a najefektívnejšie cesty sú:
- Znižovanie veľkosti kroku (Step), pri tejto metode sa zvyšuje rýchlosť pohybu servomotorov. V závislosti od druhu pohybu však existuje hraničná hodnota, kde po prekročení dochádza ku strate stability robota. Tiež je potrebne pristupovať k zmene veľmi opatrne, pretože by mohlo dôjsť pri použití v praxi k poškodeniu robota, ak pohyb servomotora nie je bezpečnostne obmedzený
- Vynechávanie nadbytočných krokov. Tento prístup vie byť tiež veľmi užitočný, ale v tomto prípade je potrebné dane pohyby podrobne testovať, pretože napriek vysokej úspešnosti zrýchlenia pohybov, nastal v jednom prípade problem s nepravidelným zakopávaním.

1.1 Vývoj pohybov

Názov pohybu	Popis	Step	Počet póz	Čas	Hodnotenie
Forward	1 krátky krok	40	66	02:600	Stabilný, Pomalý
Forward50	5 krokov, zakrivený	40	170	02:600	Stabilný, Nepresný
Forwardsb	1 krátky rýchly krok	40	34	01:320	Stabilný, Rýchly
Forwards50b	5 rýchlych krokov	40	86	03:440	Stabilný, Rýchly
Forwards50c	5 rýchlych krokov	40	95	03:760	Stabilný, Rýchly
Shoot	defaultny kop	40	121	04:800	Pomerne slabý kop
Shootb	zrýchleny kop	40	61	02:400	Stabilný
ShootC	rýchly kop	40	61	02:840	Stabilný, Silný

1.1.1 Pohyb dopredu

Po testovaní a úprave pohybov dopredu sme získali lepšie, ktoré dosahovali neporovnateľne lepšie výsledky. Krátky pohyb dopredu bol zrýchleny na polovicu pri zachovaní stability. Dlhý pohyb dopredu, pozostávajúci zo série piatich súvislých krokov sme tiež zrýchlili, zabezpečili stabilitu a upravili chybné zanášanie.

1.1.2 Streľba

Vzhľadom na strelbu, pri porovnaní základneho kopu a posunu lopty pomocou narazenia pohybom dopredu, bola zistená malá vzdialenosť dostrelu. Upravený pohyb ShootC je aktuálne nielen zrýchlený, ale aj dostatočne silný a presný. V opakovaných pokusoch boli úspešne strelené góly z polovice ihriska.

1.1.3 Ďalšie pohyby

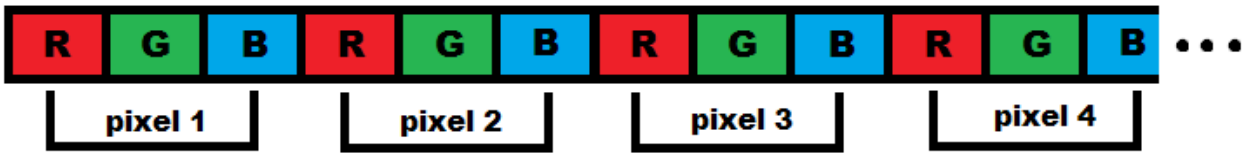
Metódou vynechávania krokov sa nám podarilo zrýchliť všetky ostatné pohyby pri zachovaní stability. Pre potrebu presnejšieho nastavovania hráča k lopte bolo potrebne vytvoriť ešte zmenšené pohyby. Pri zmenšení kroku do boku nastal problem s pravom stranou, pretože dochádzalo k strate rovnováhy a následnym pádom. Problém bol úspešne vyriešený inverzným prepísaním posunu doľava.

2 Rozpoznávanie obrazu

Rozpoznávanie obrazu sme v našom projekte riešili cestou porovnávania objektov s ich referenčnými farbami. Tou najzákladnejšou časťou pri snahe získať obraz okolia bolo pochopenie práce s kamerou a s ňou spojených základných funkcií. Medzi tieto funkcie patria, mimo iných, tieto:

- `void wb_camera_enable(WbDeviceTag camera, int ms)` – inicializácia kamery so snímkovaním každých ms milisekúnd.
- `double wb_camera_get_fov(WbDeviceTag camera)` – funkcia vracia zorné pole kamery
- `int wb_camera_get_width(WbDeviceTag camera)` – funkcia vracia šírku obrazu
- `int wb_camera_get_height(WbDeviceTag camera)` – funkcia vracia výšku obrazu
- `const unsigned char *wb_camera_get_image(WbDeviceTag camera)` – dôležitá funkcia, ktorá vracia jednorozmerný vektor farieb pixelov v obraze

Azda najdôležitejšou funkciou je načítanie obrazu. Tu treba poznamenať, že obraz je uložený ako vektor trojíc farieb, teda pre každý pixel pripadajú tri zložky – červená (R), zelená (G) a modrá (B).



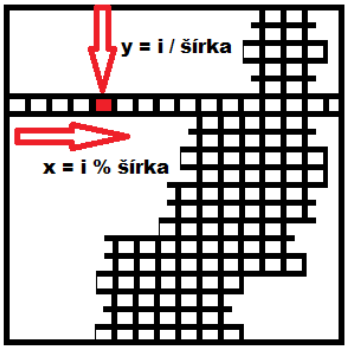
Obrázok 1: Grafické znázornenie vektora obrazu z kamery

Keďže RGB komponenty môžu nadobúdať hodnoty od 0 do 255, výhodné je tieto trojice normovať, čo čiastočne odstráni nepriaznivé dôsledky rozdielneho osvetlenia. Vyberie sa zložka s najväčšou hodnotou a ňou sa predelia hodnoty všetkých troch farebných zložiek. Získame tak trojicu hodnôt medzi 0 a 1, čo značne uľahčuje porovnávanie s referenčnými farbami.

Hľadanie objektu v obraze sa robí jednoducho porovnaním všetkých jeho pixelov s s danou referenčnou referenčnou farbou s určitou toleranciou. Teda v cykle sa vyhodnocujú hodnoty nasledovného logického výrazu:

```
(fabs(pixel[0] - ref[0]) < tolerance &&
fabs(pixel[1] - ref[1]) < tolerance &&
 fabs(pixel[2] - ref[2]) < tolerance)
```

Ak výraz platí, teda vieme, že daný pixel patrí objektu a teda zaznamenáme si jeho súradnice v obraze. Keďže obraz je uložený nie ako dvojrozmerná matica, ale ako jednorozmerný vektor, je pri použití prehľadávacieho cyklu s počítadlom i vhodné použiť matematické operácie celočíselného delenia pre určenie y-ovej súradnice a modusu pre určenie x-ovej súradnice daného pixelu.



Obrázok 2: Súradnice

Pre presné určenie polohy objektu stačí zistiť priemer súradníc x, y pre všetky vyhovujúce pixely. Uholy smeru a elevácie voči stredu obrazu získame pomocou vzorcov:

- `smer = ( priemer(x) / šírka - 0.5) * zorné pole kamery`
- `elevácia = - ( priemer(y) / výška - 0.5) * zorné pole kamery`

Ak chceme objekt sledovať aj pri dynamickom pohybe, potrebujeme kamerou pohybovať. Na pohyb kamery pri robotovi Nao slúžia dve krčné servá – „HeadYaw“ a „HeadPitch“. Pohyb serva do danej polohy učiníme použitím funkcie:

- `void wb_servo_set_position(WbDeviceTag servo, double position)`

Ak už objekt máme na obrazovke, kameru na neho vycentrujeme použitím dvojriadkového príkazu

- `wb_servo_set_position(head_yaw, wb_servo_get_position(head_yaw)-objectDirectionAngle);`
- `wb_servo_set_position(head_pitch, wb_servo_get_position(head_pitch)-objectDirectionAngle);`

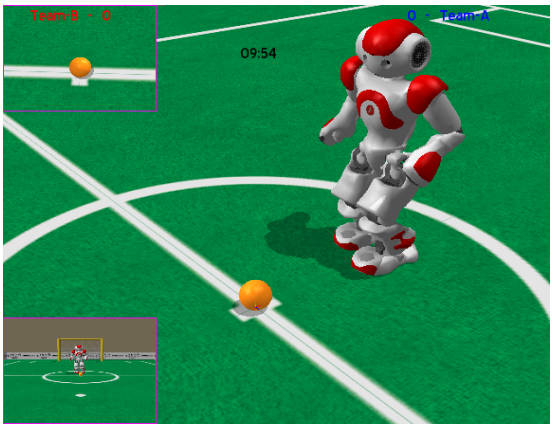
Podobným spôsobom môžeme sledovať akýkoľvek farebne konzistentný objekt.

Ak však daný objekt nemáme na obrazovke, je potrebné ho nájsť. Preto je dôležité vytvoriť algoritmus procedúry, ktorá postupným otáčaním hlavy prehľadá celé okolie robota a dokáže dostať hľadaný objekt na obrazovku (tj. „Head Scan“, „Ball Scan“ alebo „Goal Scan“). Keďže servá majú obmedzený rozsah pohybu, je treba umožniť robotovi v prípade potreby aj otáčanie celého tela o 180°.

2.1 Rozpoznávanie lopty

Ak už vieme rozpoznávať objekty vyššie popísanou metódou, pre rozpoznávanie lopty stačí iba dosadiť vhodné referenčné farby. Je veľmi dôležité správne nakalibrovať tieto hodnoty, pretože už pri malej nepresnosti je možné, že robot bude namiesto oranžovej lopty sledovat svojho „červeného“ spoluhráča. Podobnou chybou trpela referenčná farba v pôvodnom kóde. Metódou pokusu a omylu sme stanovili vhodné referenčné farby pre loptu na 1 pre červenú, 0.6 pre zelenú a 0.2 pre modrú zložku:

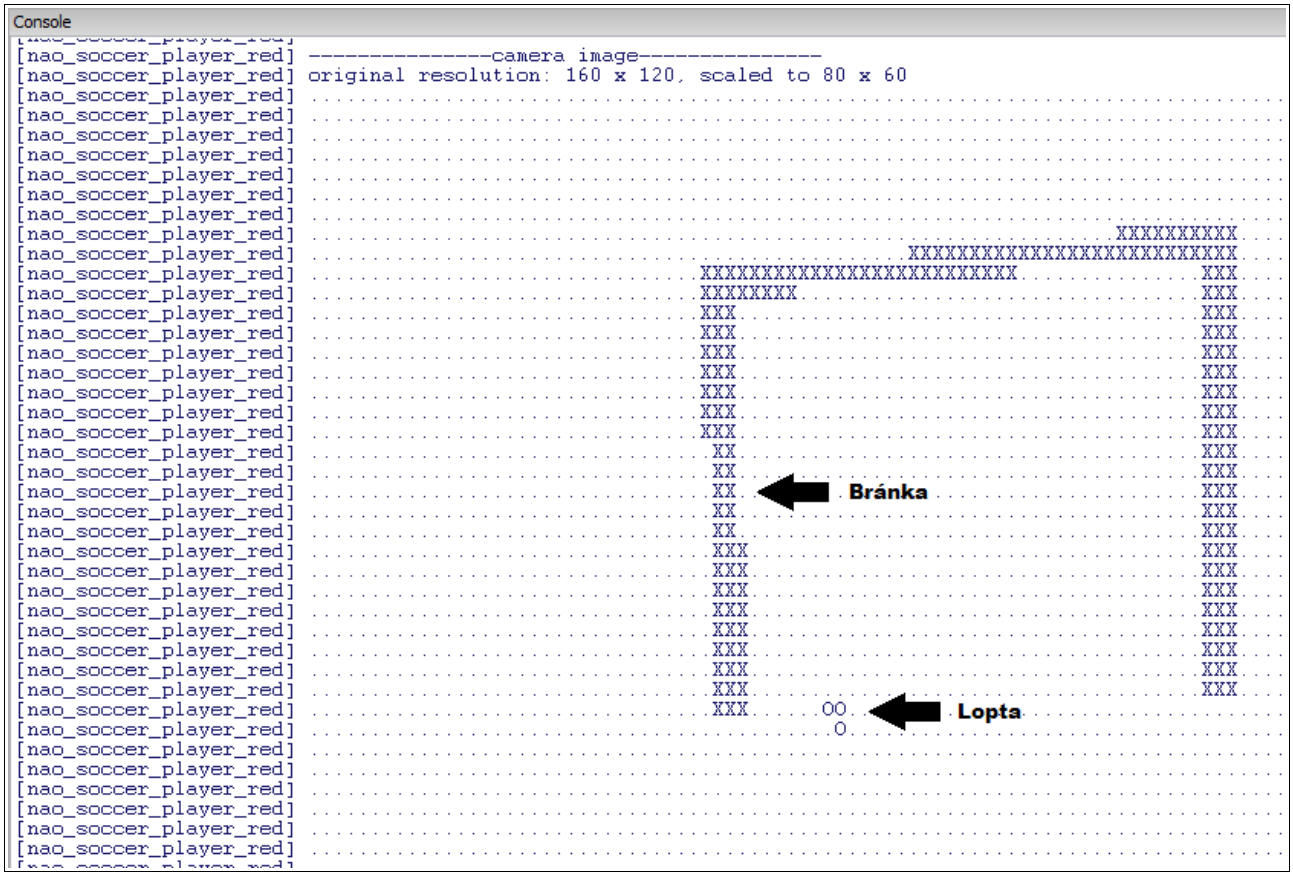
- `const double BALL_COLOR[3] = { 1.0, 0.6, 0.2 };`



Obrázok 3: Robot sledujúci loptu

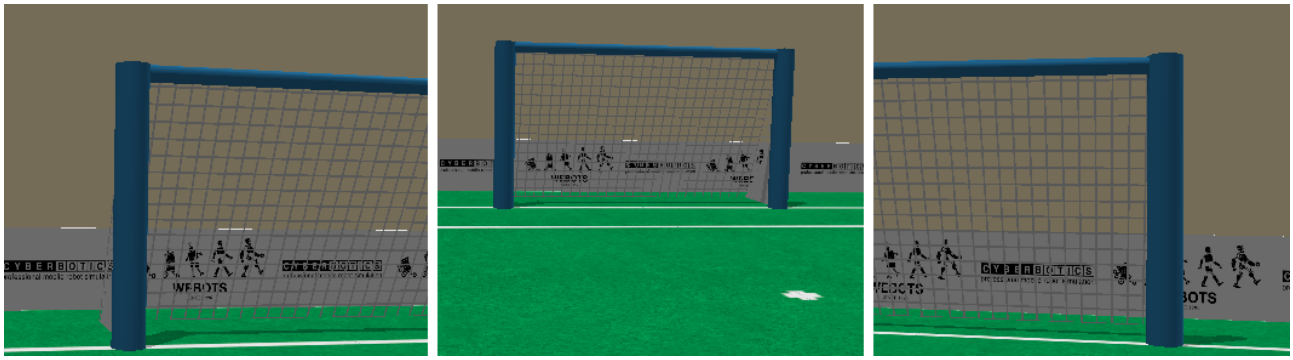
2.2 Rozpoznávanie brány

Ďalšou dôležitou úlohou bolo naprogramovať rozpoznávanie brány. Pôvodný algoritmus detekoval bránu podobne, ako ostatné objekty, čo zväčša nespĺňalo všetky potreby. Nami vytvorené algoritmy dokážu presne určiť stred brány, ako aj vzdialenosť robota k nej, čo veľmi rozširuje taktické možnosti počas hry.



Obrázok 4: Výpis obrazu z kamery v textovom móde

Pri hľadaní brány môžu v princípe nastať dve prípady. Buď robot vidí celú bránu, alebo ak je príliž blízko, len jej časť.

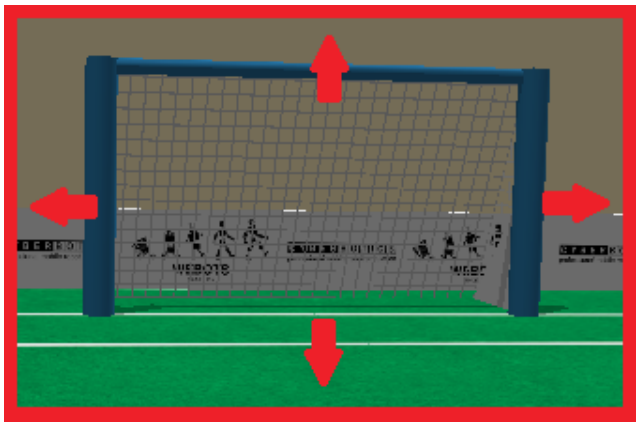


Obrázok 5: Možnosti výhľadu na bránku

Tento problém sme vyriešili vytvorením dvoch samostatných prístupov. Prvý, presnejší, je využiteľný, ak je robot vzdialený a vidí celú bránku a druhý prichádza na pomoc v prípade, že prvý prístup nedokáže poskytnúť potrebné údaje.

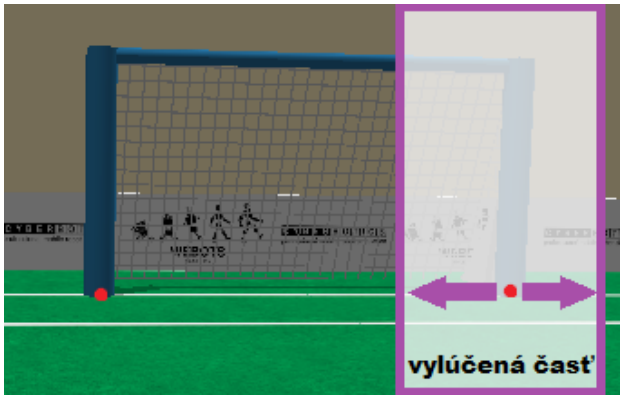
2.2.1 Vzdialená brána

To, či robot vidí celú bránku, overíme tak, že zistíme, či všetky okrajové pixely obrazu NEsúhlasia s referenčnou farbou brány.



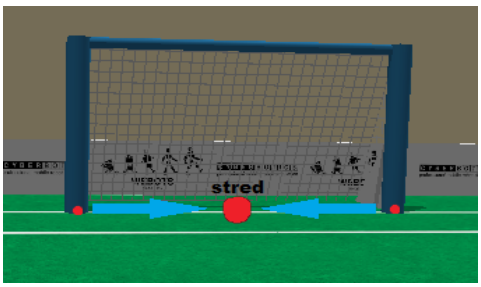
Obrázok 6: Kontrola okrajových pixelov

Ak je táto podmienka splnená, môžeme sa pustiť do hľadanie stredu bránky. Ako prvú vec, nájdeme pixel referenčnej farby, ktorý je na obraze najnižšie, teda má najvyššiu hodnotu súradnice y. Ak ho nájdeme, vyčleníme z prehľadávania jeho okolie a nájdeme druhý pixel referenčnej farby, ktorý je najnižšie v ostávajúcej časti obrazu. Pre oba pixely je vhodné učiniť kontrolu, či susediace pixely tiež referujú s farbou bránky. Získame tak istotu, že dané pixely sú skutočne koncami žrdí brány.



Obrázok 7: Systém prehľadávania obrazu s vylučením okolia prvej žrďe

Ak máme určené súradnice oboch žrdí, tak môžeme jednoduchým spriemerovaním určiť stred brány, respektíve určitou modifikáciou určiť ako cieľ priestor medzi stredom brány (kde zvyčajne stojí brankár) a žrďou. Rovnako vieme pomocou goniometrie určiť pomerne presnú vzdialenosť robota k jednotlivým žrdiam a teda aj jeho pozíciu na na ihrisku. Robot vie presne určiť, či je od brány naľavo, alebo napravo, čo len ďalej nahráva jeho možnostiam.



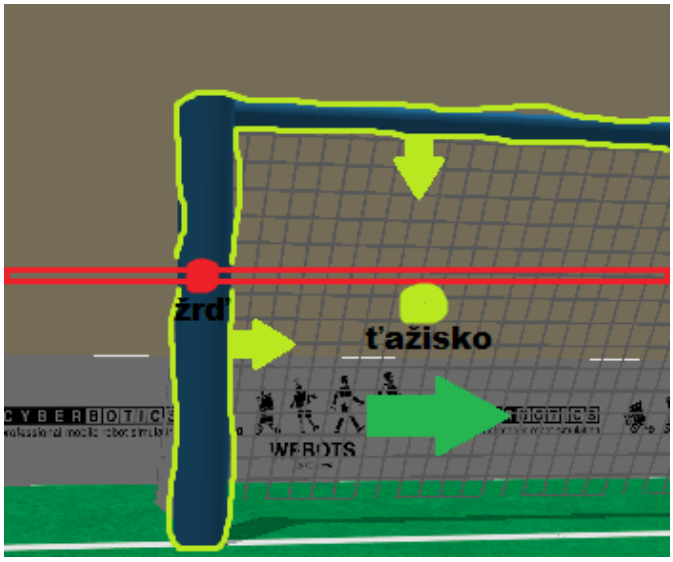
Obrázok 8: Výsledok analýzy obrazu metódou Vzdialenej brány

2.2.2 Blízka brána

Nevýhodou vyššie popísného algoritmu je jeho nefunkčnosť, ak nie je možné zobraziť celú bránu do jedného obrazu. Preto bolo nevyhnutné vytvoriť aj alternatívne riešenie, ktoré hoci nedokáže presne určiť vzdialenosť jednotlivých žrdí, umožňuje zistiť ich smer, ako aj smer stredu brány.

Algoritmus pre rozpoznanie bránky na blízku vzdialenosť využíva skutočnosť, že ak nie je možné zobraziť celú bránu v jednom obraze, je isté, že jednotlivé žrďe majú šírku aspoň 8 pixelov. Princíp spočíva v preložení obrazu vodorovnou čiarou priamo v jeho strede. Preskúma sa iba tento jeden riadok a nájdu sa v ňom referenčné pixely bránky. Tieto referenčné pixely tvoria žrď brány, ktorej pozíciu voči žrďi určíme spriemerovaním pozície všetkých referenčných pixelov na celom obraze. V závislosti od pozície ťažiska týchto pixelov voči žrdi určíme približný stred brány:

```
• if (pozicia_taziska > pozicia_zrde)
    pozicia_brany = ( pozicia_zrde / sirka_obrazu - 0.5) * fov + 0.2;
• else
   pozicia_brany = ( pozicia_zrde / sirka_obrazu - 0.5) * fov - 0.2;
```



Obrázok 9: Analýza obrazu metódou blízkej brány

Hoci táto metóda robotovi umožňuje s určitosťou určiť smer brány, nedokáže určiť vzdialenosť. Alternatívne je možné určovať vzdialenosť podľa šírky žrde (počtu referenčných pixelov prechádzajúcich horizontálne skrz ňu), ale to je úloha na dlhšie testovanie. Naša funkcia vracia v každom prípade hodnotu orientačných 0.5 metra.

3 Stratégia

Popis stratégii sa dá rozdeliť zvlášť na rozbor činnosti brankára a činnosti hráčov v poli.

3.1 Stratégia strelenia gólu

Ide o najdôležitejší bod, ktorý bolo treba naprogramovať a predsa sa riešil ako jeden z posledných. Keď sme už mali k dispozícii dokonalé ovládanie kamery a relatívne dobré pohyby, pustili sme sa aj do samotnej hry.

Stratégia strelenia gólu pozostáva zo 4 častí:

1. Detekcia lopty a cesta k lopte
2. Detekcia súperovej bránky
3. Príprava na streľbu / Navigácia robota do pozície na jednej osi s loptou aj bránkou
4. Samotná streľba

Každá jedna z nich je kapitola sama o sebe, teda sa im budeme venovať postupne.

3.1.1 Detekcia lopty a cesta k lopte

Detekcia lopty a podobných objektov je podrobne rozpísaná v 2. kapitole. Treba len

poznamenať, že je možné, že lopta bude umiestnená za robotom, teda okrem skenovania je pravdepodobné, že robot sa bude musieť aj otáčať.

Už zaujímavejším problémom je samotná cesta k lopte. Nielen, že je potrebné mať rýchlu a stabilnú chôdzu, ale dôležiá je aj presnosť otáčania k lopte. Je najvhodnejšie, ak robot smeruje k lopte po čo najpriamejšej ceste. Celé toto divadlo sa dá docieliť pomocou troch pohybov – otočenie vľavo, vpravo a chôdza vpred.

Potrebnú presnosť otáčania sme docielili „usekávaním“ otáčacích pohybov presne vo chvíli, keď je robot otočený priamo k lopte. Niekedy to síce spôsobuje cukanie, ale zväčša sa robot dokáže veľmi presne nasmerovať, tam, kam treba.

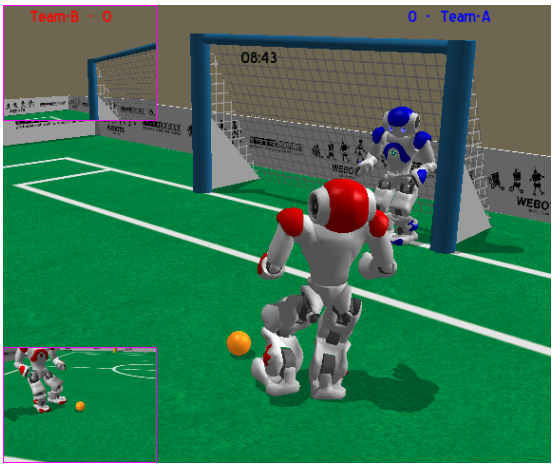
Cesta k lopte však nemôže byť úplne priamočiara, pretože ani senzorické informácie z kamery nie sú pri knísavom pohybe vpred úplne presné. Je preto potrebné implementovať určitú toleranciu, v rámci ktorej robot ide stále rovno, hoci sa oproti lopte o pár desatiniek radiánu vychyľuje.

Keďže robot má počas cesty k lopte relatívne veľké množstvo času, pokúsili sme sa ho nejako zmysluplne využiť a teda robot si každých 5 sekúnd cesty snaží aktualizovať pozíciu brány tak, že jej venuje na zlomok sekundy pohľad. Tento systém funguje úplne skvele a znižuje množstvo času, ktoré robot strávi hľadaním bránky, keď dorazí k lopte.

3.1.2 Detekcia súperovej bránky

Teoreticky je detekcia bránky podrobne rozpísaná v 2. kapitole. Použité sú dve metódy, kde každá má trochu iný prístup. V praxi však často nastávajú prípady, kedy je ťažké určiť, ktorá metóda je výhodnejšia (rýchlejšia) a to hlavne ak robot stoji na hranici pôsobnosti oboch metód. Kdežto pri metóde detekcie vzdialenej bránky sa robot potrebuje pozerať na celú bránku, pri metóde detekcie blízkej bránky je pre neho výhodné pozerať sa iba na pravý, alebo ľavý horný roh. Niekedy to spôsobuje „nerozhodnosť“ robota, kedy nevie, ktorú metódu skôr použiť.

Riešenie tohto problému je striktne oddeliť jednu metódu od druhej. Teda to znamená dve samostatné detekcie, dve samostatné skeny. Keďže hľadanie by to touto formou zabralo všeobecne viac času, rozhodli sme sa ponechať simultánne používanie oboch metód aj napriek občasným menším problémom.

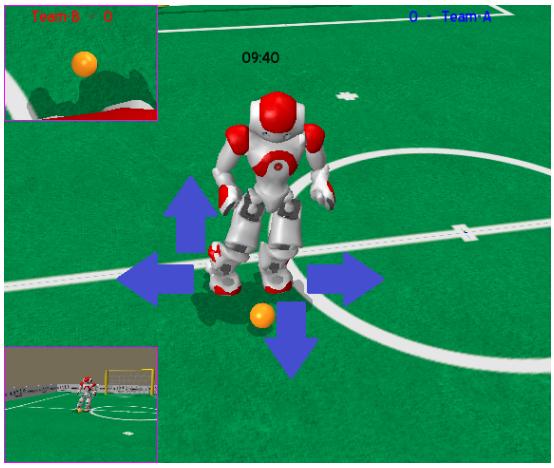


Obrázok 10: Robot sleduje ľavý horný roh pri otačaní sa smerom k bráne použitím metódy Blízkej brány

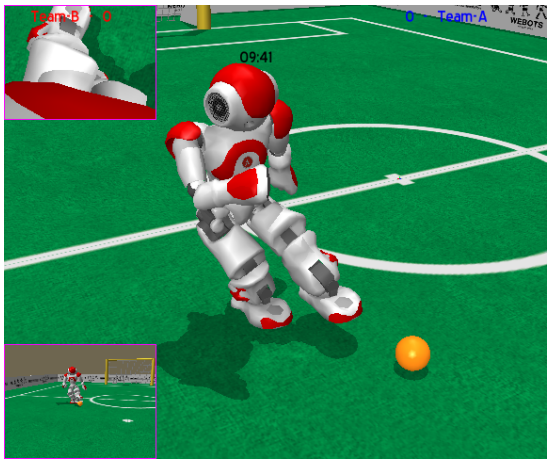
3.1.3 Príprava na streľbu a streľba

Príprava na streľbu je asi najdôležitejšia fáza na ceste k vsieteniu gólu. Na začiatku sa robot otočí smerom k bráne a túto orientáciu sa snaží si udržať. Ak už vidí, že je voči bránke správne otočený, upriami svoju pozornosť na loptu a snaží sa dobre sa postaviť pred streľbou.

Pozorujúc loptu a vnímajúc informácie o jej smere a vzdialenosti robot „stepuje“ okolo lopty pomocou pohybov úkrok doprava, doľava, krok vzad a krok vpred až kým sa mu nepodarí dostať loptu tesne pred ľavú nohu. V prípade, že všetky podmienky sú splnené, robot pristúpi k streľbe.



Obrázok 11: Príprava na streľbu

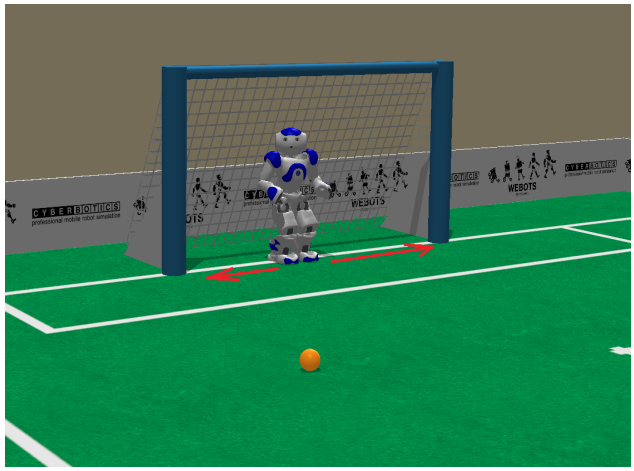


Obrázok 12: Streľba

3.2 Stratégia brankára

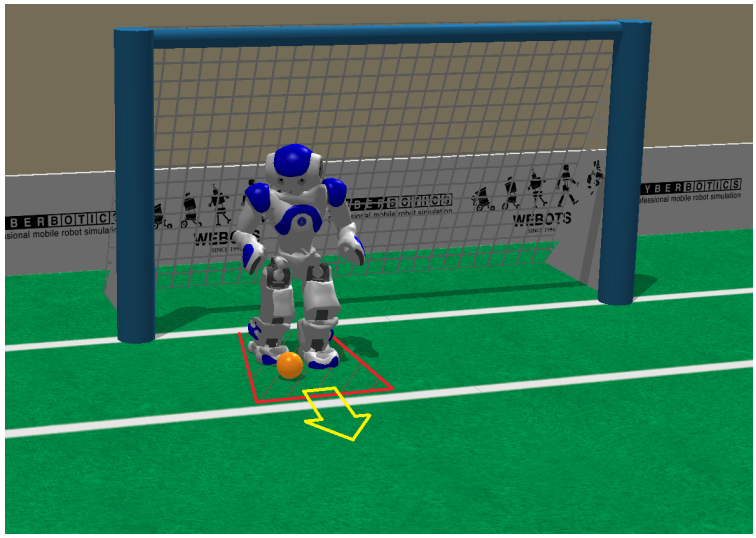
Rýchlosť pohybov akými sa robot dokáže stabilne pohybovať nám neumožňuje implementovať zákrok proti strele smerujúcej do odkrytej časti brány. Našou snahou bolo vypracovať jednoduché správanie sa brankára v rôznych situáciach. Na základe zistených hodnôt smeru lopty a jej vzdialenosti sa robot pohybuje úkrokmi vpravo a vľavo. Tieto úkony vykonáva len v oblasti bránkovej čiary. Cieľom toho je zväčšiť šancu, že pri strele na bránu lopta trafila nášho brankára, čím by sme nedostali gól.

Základom tejto stratégie je vyhýbanie sa používaniu pohybov otáčania okolo vlastnej osi. Pri otáčaní stratí robot svoj základný smer a aj opakovaním otáčaním sa už robot nedostane do pôvodnej polohy, t.j. státie v strede brány a smer rovnobežný s postrannou čiarou ihriska. Avšak problém nastáva pri páde. Pri pokuse útočiacich protihráčov sa často stáva, že pri snahe streliť gól sa dostanú do kontaktu s brankárom, ktorý môže spadnúť. Vtedy sme nutení použiť aj pohyby otáčania, čo nám spôsobí, že pri akejkoľvek ďalšej akcii sa robot nebude správať tak, ako sme si to predstavovali.



Obrázok 13: Pohyb na bránkovej čiare

Ďalším rozšírením činnosti brankára je schopnosť dostať loptu preč z oblasti pred bránou. Ak sa lopta nachádza priamo pred robotom a v malej vzdialenosti, tak sa ju pokúsi odkopnúť preč z priestoru pred bránou. Po tomto pokuse sa vždy snaží vrátiť späť na čiaru. Nevýhodou pri tomto prístupe je to, že brankár koná len na základe informácií o polohe lopty (smer, vzdialenosť) a svojho relatívneho postavenia vzhľadom k bráne. Neberú sa do úvahy umiestnenia ostatných robotov, či už spoluhráčov alebo protihráčov.



Obrázok 14: Pokus o odkopnutie lopty

4 Komunikácia medzi robotmi

Využitím zariadení „emitter“ a „receiver“ je možné zabezpečiť obojstrannú komunikáciu medzi robotmi, ako aj s koordinačným programom, „superviserom“.

Komunikácia medzi robotmi prebieha prostredníctvom paketov. Jeden paket tvorí zvyčajne štruktúra s hlavičkou, ktorá určuje adresáta paketu, a telom, kde sú ďalšie informácie:

```
typedef struct  {
  char header[sizeof(INFO_MESSAGE_STRUCT_HEADER) - 1]; //hlavička
  double ball_dist;  // informácia o vzdialenosti k lopte
  int attacking;  // informácia o stave
} InfoMessage;
```

V súčasnosti využívame tieto zariadenia na zdieľanie informácie o vzdialenosti od lopty. Pri lopte je z dôvodu vylúčenia kolízie dovolené byť len jednému hráčovi, druhý vždy čaká v istej vzdialenosti od lopty.

V budúcnosti je možné využiť daľšie preposielané informácie k prospechu prepracovanejších tímových stratégií a taktík.

Záver

Na predmete Aplikácie umelej inteligencie sme mali možnosť zoznámiť sa a pracovať s programom Webots, ktorý nám umožňuje simuláciu Nao robotického futbalu. Získali sme základný pohľad na to, s čím všetkým treba počítať už len pri samotnej simulácii – samotné pohyby robota, spracovanie obrazu, stratégia, komunikácia medzi robotmi. Tieto úlohy a algoritmy sme síce riešili klasickými metódami, ale zároveň nás nútili premýšľať aj nad využitím rôznych prostriedkov umelej inteligencie.

Použité zdroje

<http://www.aldebaran-robotics.com/>

<http://www.cyberbotics.com/>

<http://www.robotstadium.org/>

<http://birg.epfl.ch/webdav/site/birg/users/175760/private/Nao_Programming_for_the_Robotstadium_On-line_Contest.pdf>

<http://birg.epfl.ch/webdav/site/birg/users/170218/private/report.pdf>