دانشگاه بوعلی سینا

دانشکده‌ی مهندسی کامپیوتر

رساله‌ی ارائه شده به عنوان بخشی از ملزومات
برای دریافت درجه‌ی

کارشناسی

در

گرایش مهندسی نرم‌افزار

عنوان

روبوسینا از صفر

استاد راهنما

دکتر میرحسین دزفولیان

نگارش

نیماکاویانی، مصطفی رفائی جوکندان

اردیبهشت ۱۳۸۴

تقدیم به پدر و مادر عزیزمان

که همواره و در تمامی مراحل حامی و همراه ما بوده‌اند ...

قدردانی

شروع، استمرار و موفقیت این پروژه بی شک مرهون زحمت عزیزانی است که خالصانه و بدون چشم‌داشت ما را یاری کردند، لذا پیش از شروع کلام لازم میدانیم از کلیه این عزیزان که نقش بسزایی در پیشبرد طرح ایفا نمودند تشکر نمائیم.

دوست و استاد ارجمند مهندس حمید ضرابی زاده بنیان‌گذار سنگ بنای تیم روبوسینا بودند و حضور سبز و همکاری صمیمانه‌ی ایشان برای فراهم آوردن امکانات لازم، قدم نخست در ایجاد تیم بود. اعضاء تیم روبوسینا در مدت قریب به یک سال سرپرستی این استاد عزیز بدون دغدغه‌ی محیط خارج از فضای کاری به فعالیت پرداخت و راهنمایی‌های ارزشمند ایشان درهای حرکت به سمت اهداف متعالی را برای تیم گشود. اعضای تیم مفتخرند که شاگرد و دوست این استاد گرامی باشند.

استاد عزیز دکتر میرحسین دزفولیان سرپرست گروه کامپیوتر دانشگاه و همراه ما در بحران‌های سخت کاری بودند. در شرایطی که بی‌توجهی برخی از مسئولین امکان ادامه‌ی طرح را سخت می‌نمود، دلگرمی و حمایت ایشان ما را به ادامه‌ی راه امیدوار می‌کرد. از این رو وظیفه‌ی خود می‌دانیم صمیمانه‌ترین سپاس خود را از ایشان ابراز نمائیم.

دوست و استاد مهربان مهندس حسن بشیری، سرپرست ثانوی تیم پس از عزیمت مهندس ضرابی‌زاده به خارج از کشور برای ادامه‌ی تحصیلات، یادآور موفقیت‌های بسیار برای تیم می‌باشند و افراد گروه روبوسینا به دوستی ایشان مباهات می‌کنند.

دوستان عزیز مهندس رضا نیامنش، مهندس سیّد محمّد مهدی جوادی، مهندس سیّد وهاب میرکنی و مهندس سیّد رضا میرفتّاح با حمایت‌ها و همکاری خود کمک شایانی در شناخت، تداوم و موفقیت پروژه ایفا نمودند و رفع بسیاری از مشکلات از سوی تک تک این دوستان همیشگی همواره با نام روبوسینا همراه می‌باشد و در خاطره‌ی تیم باقی است. در مقابل لطف تمامی این دوستان سر تعظیم فرود می آوریم.

همکاران همیشگی و عزیزمان در تیم شبیه‌سازی روبوسینا، خانم‌ها فاطمه امیری و سمیّه کافی نجف‌آبادی، افرادی بودند که حضور مستمرشان زمینه‌ساز شکل‌گیری و تداوم حرکت در تیم روبوسینا شد. دوستانی که از حمله‌ی سگ‌ها در نیمه‌های شب تا اشک‌ها و لبخندهای گروه در شکست‌ها و پیروزی‌ها کنار تیم بودند و بی‌شک روبوسینا بدون حضور این دوستان هرگز روبوسینا نبود.

در نهایت لطف و محبت بسیار جناب آقای علی‌اکبر پورمسلمی مربی کلاس A کنفدراسیون فوتبال آسیا، AFC، که ما را در درک مفاهیم فنی فوتبال، ایجاد یک سیستم تدافعی منسجم و فراگیری آرایش تیمی مهم و مؤثر در محیط دو بعدی همراهی کردند را ارج می‌نهیم و از این استاد عزیز بی‌نهایت سپاس‌گذاریم.

روبوسینا از صفر

چکیده

در این پایان‌نامه روند افزایشی تحلیل، طراحی و پیاده‌سازی یک تیم از عوامل هوشمند با عنوان روبوسینا به همراه خصوصیات اصلی این تیم بررسی شده است. روبوسینا یک تیم شبیه‌سازی شده فوتبال[۱] مشتمل بر ۱۱ بازیکن خودکار نرم افزاری می‌باشد که در یک محیط فیزیکی شبیه‌سازی شده فوتبال با عنوان کارگزار سرور دوبعدی[۲] عمل می‌کنند. محیط مذکور دو تیم از عوامل فوتبالیست خودکار را قادر می‌سازد تا یک بازی فوتبال را در مقابل هم و با استفاده از قوانین تعریف شده در سیستم انجام دهند. کارگزار سرور یک محیط کاملاً توزیع شده و زمان حقیقی[۳] را به گونه‌ای فراهم می آورد که در آن هریک از اعضا دو تیم باید در جهت دستیابی به هدف نهایی تیمی که همان پیروزی در بازیست با یکدیگر همکاری نمایند. این سیستم شبیه‌ساز بسیاری از پیچیدگیهای محیط واقعی از جمله نویز[۴] در حرکت اشیاً، خطا در حسگرها و عملگرها، قابلیت عملکرد محدود و محدودیت توانایی در برقراری ارتباط بین عوامل را مدل می‌کند. محوراصلی بحث در این پایان‌نامه شامل معماری یک عامل در قالب چند لایه، روش هماهنگ سازی یک بازیکن با سیستم زمان حقیقی کارگزار، روش ابداعی مکان‌یابی برای یک بازیکن، تحلیل و توسعه دنیای اطراف از دید یک بازیکن، الگوها و استراتژیهای مورد استفاده در جلوگیری از نفوذ تیم مقابل و در عین حال حمله به خطوط دفاعی حریف و در نهایت الگوریتم‌های مورد استفاده در شکستن خطوط تدافعی تیم مقابل و بثمر رسانیدن گل می‌باشد. تیم روبوسینا در جریان فعالیت خود در مسابقات متعددی شرکت نمود که نتایج این حضور به اختصار عبارتند از: قهرمانی سومین دوره‌ی مسابقات آزاد روبوکاپ آمریکا در اواخر اردیبهشت ۱۳۸۴، نایب قهرمانی سومین دوره از مسابقات آزاد روبوتیک ایران(حلی کاپ) در اوایل اردیبهشت ۱۳۸۴، مقام پنجم هشتمین دوره از مسابقات جهانی روبوکاپ در تیرماه ۱۳۸۳ در کشور پرتغال، قهرمانی دومین دوره از مسابقات آزاد روبوکاپ آمریکا در اردیبهشت ۱۳۸۳، و قهرمانی دومین دوره از مسابقات آزاد روبوتیک ایران (حلی‌کاپ) در اسفندماه ۱۳۸۲.

واژه‌های کلیدی: ۱)روبات‌های فوتبالیست، ۲)سیستم شبیه‌سازی، ۳)کارگزار شبیه‌سازی، ۴)شبکه‌های عصبی مصنوعی، ۵)درخت تصمیم، ۶)یادگیری تقویتی، ۷)هوش مصنوعی توزیع شده

---

[۱] Simulated Soccer Team

[۲] 2D Soccer Server

[۳] Real Time

[۴] Noise

به نام اهورا مزدای پاک

فهرست

۳ روبوسینا: جنبه‌های بارز

۴ گذری بر الگوریتم‌های یادگیری

فهرست شکل‌ها

فهرست جدول‌ها

۱

پیشگفتار

امروزه فوتبال توجه قشر کثیری از افراد هر جامعه را به خود معطوف کرده است به نحوی که میتوان آن را پاره‌ای از زندگی این مردم نامید. هرچند این طرز تفکر می‌تواند مخالفت‌هایی را به همراه داشته باشد اما با این حال انکار آن به عنوان یک محبوبیت فراگیر ممکن نیست. با این همه وجود تعدادی روبات با قابلیت بازی فوتبال تا حدودی دور از ذهن به نظر می‌رسد و تصور تیمی از موجودات آدم‌نما که بتوانند توپ گرد را در زمین بازی حرکت داده و برای بثمر رسانیدن گل تلاش کنند برای کمتر کسی ممکن است. اما فوتبال روبات‌ها مدتی است که در جوامع علمی جایگاه قابل قبولی برای خود یافته است و میزان گرایش بدان ظرف چند سال گذشته بطور چشمگیری افزایش داشته است و در ایران و در بین دانشجویان مقاطع مختلف و حتی دانش آموزان تحقیق در زمینه روبات‌های فوتبالیست به یکی از مباحث پر طرفدار مبدل شده است که این روند در خصوص اعضای تیم روبوسینا نیز صادق است. در این فصل به فوتبال روبات‌ها از زاویه یک مسئله با مفهوم چند عامله می‌نگریم. مباحث مطرح شده در این فصل عبارتند از : مفاهیم اولیه، محرک‌ها و هدف نهایی روبوکاپ که در بخش ۱—۱ بدان خواهیم پرداخت، بررسی دقیق فوتبال روبات‌ها از منظر چندعامله بودن که در بخش ۱—۲ مورد بررسی قرار خواهد گرفت، اهداف و انگیزه‌های شکل‌گیری روبوسینا و مفاهیمی که محور اصلی تحقیقات را ساختند در بخش ۱—۳ مورد بحث قرار می‌گیرند و در نهایت نتیجه‌گیری به همراه مروری بر قسمت‌های مختلف این پایان‌نامه را در بخش ۱—۴ مرور می‌کنیم.

۱—۱ روبوکاپ. مفاهیم اولیه، هدف نهایی

مسابقات جهانی روبات‌ها[۱] تلاشی در جهت ارتقا الگوریتم‌های هوش مصنوعی و تحقیقات مرتبط با روبات‌های هوشمند می‌باشد که در قالب یک مسئله استاندارد با قابلیت آزمایش و انعطاف بسیار بالا مطرح شده است. هدف نهایی و بلند مدت روبوکاپ عموماً به صورت زیر بیان می‌شود:

> "در نیمه قرن ۲۱، سال ۲۰۵۰ میلادی، یک تیم از روبات‌های هوشمند یک مسابقه رسمی فوتبال را، با رعایت کلیه قوانین فیفا، در مقابل فاتح آخرین قهرمان جام جهانی فوتبال انسانی

[۱] RoboCup

پیروز خواهند شد [۳].“

هدف فوق یکی از بزرگترین مبارزات علمی است که در تاریخ بشری مطرح شده و با توجه به دستاوردهای کنونی از دیدگاه بسیاری از مردم به دیده شک نگریسته می‌شود و چندان محتمل به نظر نمی‌رسد. اما تاریخ مبین این مطلب است که قدرت پیشگویی بشر هیچگاه فراتر از مرز دهه‌ها نرفته است و همواره امکان بروز یک تحول عظیم وجود دارد. علاوه بر هدف اصلی و دوردست روبوکاپ باید اهداف کوتاه مدتی را که در این جریان در حال ایجاد و تکامل هستند در نظر گرفت. هدف سازمان روبوکاپ استفاده از این رقابت‌ها به عنوان ابزاری در جهت ترفیع روبوتیک و تحقیقات هوش مصنوعی با ایجاد یک مسابقه جذاب است. روبوکاپ مجموعه‌ای از تحقیقات یکپارچه است که دامنه وسیعی از تحقیقات هوش مصنوعی را در بر می‌گیرد و از آن جمله میتوان به طراحی مفاهیم عوامل خودکار، همکاری در محیطهای چند عامله، ارتباطات استراتژیک، تصمیم‌گیری‌های زمان حقیقی، رفتار متقابل عامل‌ها، یادگیری، بینایی، کنترل موتورها، کنترل هوشمند روبات‌ها، و بسیاری از مفاهیم دیگر اشاره کرد [۴]. به بیان بهتر کلیه فعالیت‌هایی که سبب می‌شود تا یک تیم از عوامل هوشمند بتوانند به درستی در تقابل با هم عمل کنند را میتوان به عنوان اهداف ملموس و کوتاه مدت روبوکاپ در نظر گرفت و حتی با این احتمال که هدف نهایی هرگز حاصل نشود اهداف کوتاه مدت بدست آمده به اندازه کافی سودمند خواهند بود. هدف دومی که توسط سازمان جهانی روبوکاپ دنبال می‌شود استفاده از ایده روبوکاپ در آموزش و تحریک علاقه عمومی به روبوتیک و هوش مصنوعی با استفاده از یک موضوع با دامنه بسیار وسیع و پر هیجان است [۵]. در حال حاضر با توجه به تعداد رو به رشد دانشگاه‌هایی که روبوکاپ را به عنوان یک طرح تحقیقاتی مطرح می‌کنند و با توجه به افزایش محسوس گرایش به روبوکاپ از سوی جامعه عمومی چنین بنظر می‌رسد که سازمان جهانی در دستیابی به این هدف موفق بوده است.

مفهوم دیگری که در روبوکاپ به سهولت قابل دسترسی است امکان استفاده از نظریه‌ها و الگوریتم‌های متفاوت در برخورد با یک مسئله استاندارد است به شکلی که با استفاده از این محیط روش‌های مختلف می‌توانند به سادگی با هم مقایسه شوند و میزان پیشرفت به سادگی قابل بررسی و اندازه‌گیری است [۵]. نمونه‌ای از تجربیات از این قسم که پیشتر مورد استفاده قرار گرفته است شطرنج بود که به منظور بررسی میزان دقت و درستی الگوریتم‌های جستجو مطرح شد و پیشرفت این روند را می‌توان از طریق مشاهده بازیهایی که بین کامپیوتر و انسان انجام می‌شود اندازه‌گیری نمود. بدلیل قابلیت‌ها و ویژگیهایی که روبوکاپ از آنها برخوردار است این محیط در جهت تولید فن‌آوری‌های مورد نیاز در نسل آینده صنعت مفید به نظر می‌رسد.

برای دستیابی به هدف بلند مدت روبوکاپ، سازمان جهانی روبوکاپ شاخه‌های مختلفی را معرفی نموده است که هر یک از آنها روی بخش خاصی از مفاهیم مطرح شده تمرکز دارند. در حال حاضر مهمترین بخش‌ها عبارتند از:

- روبات‌های آدم‌نما (Humanoid Robots) : در این بخش از مسابقات روبوکاپ توانایی‌های پایه‌ای یک بازیکن فوتبال نظیر توانایی در شوت به سمت دروازه و نیز دفاع از خط دروازه بررسی می‌شود. در این حالت تمایز اشیاء از طریق تفاوت در رنگ آن‌ها صورت می‌گیرد و دستکاری و کنترل از سوی انسان و در خارج از سیستم مجاز است. در این بخش تمرکز اصلی بر ایجاد سیستمی است که در آن یک روبات بتواند در حرکت، حفظ تعادل و شناخت اشیاء عملکردی مشابه با انسان داشته باشد.
- روبات‌های امدادگر (Rescue Robots) : از آن‌جا که امداد یکی از عوامل مهم در زندگی بشر بوده و همواره بخش عظیمی از توجه را به خود معطوف نموده است پروژه‌ی روبات‌های امدادگر با هدف ترفیع دامنه تحقیق و توسعه‌ی روبات‌هایی که توانایی کمک‌رسانی را دارا باشند تعریف شد. این بخش از روبوکاپ در دو قسمت شبیه‌سازی و مفاهیم بنیادیِ روبوتیک تعریف شده است که در آن‌ها بر

مراحل مختلفی از پروژه مشتمل بر سازمان‌دهی همکاری چندعامله، بررسی عوامل روبات فیزیکی در جستجو و نجات، امدادگران دیجیتال شخصی، زیرساخت‌های اطلاعاتی، و گام‌های ارزیابی در سیستم‌های امدادگر و روباتیک، بررسی می‌شوند.

- روبات‌های اندازه متوسط (Middle Size Robots) : در این بخش هر تیم حداکثر دارای ۴ روبات بوده که هر یک در حدود ۷۵cm عرض و ۵۰cm قطر دارند. حدود زمین بازی ۹m در ۵m بوده و روبات فاقد اطلاعات کامل از محیط می‌باشد. در این بخش مهمترین زمینه‌های تحقیق را مکان‌یابی، بینایی، کنترل موتورها و تقابل با سخت‌افزار در بر می‌گیرند.
- روبات‌های اندازه کوچک (Small Size Robots) : در این بخش هر تیم شامل ۵ روبات بوده که هر یک در حدود ۵۰cm ارتفاع و ۱۵cm قطر دارند. زمین بازی به ابعاد زمین تنیس روی میز بوده و دوربین‌هایی که در بالای زمین نصب شده‌اند امکان دید جامع از زمین بازی را برای هر روبات فراهم می‌آورند. زمینه‌های مهم تحقیق در این بخش عبارتند از: کنترل هوشمند روبات، پردازش تصویر و ارتباطات استراتژیک.
- سگهای چهارپای سونی (Sony 4-Legged Robots) : در این بخش هر تیم ۴ سگ سونی را که با نام AIBO شناخته می‌شوند در اختیار دارد. زمین بازی مشابه زمین روبات‌های اندازه متوسط است. روبات‌ها دید کاملی نسبت به زمین بازی ندارد اما از علائم موجود در اطراف زمین به منظور مکان‌یابی بهره می‌گیرد. مهمترین شاخه‌های تحقیق در این بخش را کنترل هوشمند روبات و تفسیر اطلاعات حسی شامل می‌شوند.
- شبیه‌سازی (Simulation) : در لیگ شبیه‌سازی هر تیم ۱۱ بازیکن نرم‌افزاری را در اختیار دارد که در یک محیط شبیه‌سازی شده عمل می‌کنند. این لیگ مشتمل بر دو بخش شبیه‌سازی دوبعدی (2D Environment) و سه‌بعدی (3D Environment) می‌باشد که در محیط دوم تلاش بیشتری در جهت تطبیق محیط شبیه‌سازی شده با محیط واقعی انجام شده است. در محیط دوبعدی عملکرد هر شیئ از محیط به یک فضای دوبعدی نگاشته می‌شود حال آنکه در مدل سه‌بعدی این نگاشت به یک فضای سه‌بعدی صورت می‌گیرد. در این محیط عمده فعالیت‌ها متمرکز بر همکاری عوامل هم‌تیم، یادگیری ماشین، و تحلیل مدل عملکرد حریف می‌باشد. عدم نیاز این بخش به امکانات سخت‌افزاری پیچیده و بعضاً پر هزینه آن را تبدیل به متداول‌ترین دسته از رقابت‌های روبوکاپ نموده است. بعلاوه امکان آزمایش یک تیم در مقابل حریف در بخش شبیه‌سازی به مراتب سهل‌تر از انجام عمل مشابه در سایر لیگ‌ها می‌باشد.

تمرکز اصلی در این رساله بر شاخه شبیه‌سازی قرار گرفته است. این بخش بر مبنای استفاده از یک سیستم شبیه‌سازی شده بنا نهاده شده است و کلیه اعمال اصلی توسط یک محیط زمان حقیقی و خطادار با عنوان کارگزار شبیه‌سازی صورت می‌گیرد به نحوی که دو تیم از روبات‌های نرم‌افزاری را قادر می‌سازد تا در برابر هم به رقابت بپردازند. کارگزار شبیه‌سازی به عنوان یک برداشت از محیط حقیقی و با حذف پیچیدگی‌هایی نظیر حرکت روبات و تشخیص اشیا مختلف در زمین بازی امکان پیاده‌سازی کاراتر و تمرکز بیشتر روی بخش‌های یادگیری را فراهم آورده است.

۱ — ۲ فوتبال روبات‌ها از منظر چندعامله بودن

هوش مصنوعی توزیع شده زیرگروهی از هوش مصنوعی است که در آن بحث اصلی را کنترل سیستمی از موجودیت‌های مستقل و چندگانه که همواره با هم در تقابل هستند تشکیل می‌دهد. این بخش به طور کلی به دو زیر بخش شکسته شده که حل مسئله توزیع شده [۲] و محیط‌های چند عامله[۳] نامیده می‌شوند[٦]. در حل مسئله توزیع شده محور اصلی تحقیقات را مفاهیم مدیریت اطلاعات نظیر تجزیه وظیفه‌مندی‌ها و روش‌های حل تشکیل می‌دهند به نحوی که تعدادی از عوامل در جهت حل یک موضوع خاص با هم تقابل و همکاری دارند، حال آنکه در محیط‌های چند عامله تحقیقات بر ایجاد یک سیستم پیچیده که در آن عوامل به طور مستقل فعالیت می‌کنند و هدف کنترل نحوه عملکرد این عوامل است تمرکز دارد.

یک عامل را می‌توان هر موجودی در نظر گرفت که که توانایی درک محیط پیرامون خود از طریق حسگرها و فعالیت در این محیط از طریق عملگرها را داراست[۷]. در چنین محیطی یک عامل می‌تواند نسبت به محیط اطراف خود پیشینه ذهنی داشته باشد و یا در جریان فعالیت در این محیط اطلاعات لازم را کسب نماید. همچنین یک عامل می‌تواند هدف خاصی را در این محیط دنبال نماید و در جهت استحصال آن با سایر عوامل همکاری داشته باشد که در چنین شرایطی این عامل به عنوان یک عضو از محیط چند عامله در نظر گرفته می‌شود و چنانچه کلیه اعضا در چنین محیطی برای دستیابی به یک هدف با هم همکاری نمایند می‌توان آنها را یک تیم نامید. در چنین محیطی امکان ارتباط بین اعضا از طریق گفتگوهای رادیویی (در محیط واقعی) و یا تبادلات پیش‌فرض (در محیط شبیه‌سازی) میسّر می‌شود و هر عامل باید به شکل کاملاً خودکار و در عین حال در جهت نیل به هدف تیمی تلاش کند. در صورتی که مجموعه‌ای دیگر از عوامل در جهت رسیدن به هدفی بر خلاف عوامل همکار در یک محیط فعالیت نمایند می‌توان آنها را دشمن نامید.

استفاده از یک سیستم چندعامله در بسیاری از موارد مفید و مؤثر می‌باشد. در شرایطی که یک سیستم از عوامل هوشمند با اهداف متفاوت و حتی در شرایطی متعارض با هم فعالیت می‌کنند استفاده از الگوهای چندعامله تنها راه حل می‌باشد[۸]. اما در شرایطی که این اضطرار در استفاده از مدل چندعامله وجود ندارد نیز تمرکز بر روی یک سیستم چندعامله می‌تواند مزایای متعددی را به همراه داشته باشد که به اختصار می‌توانند در قالب زیر بیان شوند[۵]:

- استفاده از عوامل متعدد می‌تواند به عنوان ابزاری در جهت محاسبه موازی مورد استفاده قرار بگیرد و سبب افزایش عملکرد در سیستم شود. این روش خصوصاً در مواردی که امکان شکستن مسئله به تعدادی مسئله قابل حل توسط عوامل مختلف می‌باشد بسیار سودمند خواهد بود.
- استحکام در سیستم‌های چندعامله بسیار بالاتر از سیستم‌های تک‌عامله می‌باشد چرا که در سیستم تک‌عامله هر اختلال در نحوه عملکرد عامل سبب ازکارافتادگی سیستم می‌شود در صورتیکه در سیستم‌های چندعامله سیستم با وجود نقص در عملکرد یک یا چند عامل نیز امکان ادامه فعالیت را داراست.
- شکل پیمانه‌ای [۴] در سیستم‌های چندعامله ملموس‌تر بوده و امکان فعالیت در چنین سیستمی ساده‌تر است چرا که یک برنامه‌نویس بخش‌های مختلف سیستم را استخراج نموده و آن را در اختیار عوامل

---

[۲] Distributed Problem Solving(DPS)

[۳] Multi-Agent Systems (MAS)

[۴] Modular

مختلف قرار می‌دهد. در چنین حالتی هر عامل به سادگی وظیفه خود را انجام می‌دهد که نسبت به واگذاری کل فرایند به یک عامل هزینه و زمان کمتری را به سیستم تحمیل می‌کند.

- قابلیت مقیاس‌پذیری از دیگر ویژگی‌های موجود در سیستم‌های چند عامله است. در این سیستم افزودن یک عامل به مجموعه در مواقع مورد نیاز به سادگی امکان‌پذیر است حال آنکه در یک سیستم یکپارچه دستیابی به این امکان مستلزم صرف هزینه زیادی می‌باشد.
- گستردگی جغرافیایی[۵] مزیت دیگری است که در آن یک سیستم چندعامله می‌تواند محیط فعالیت خود را همزمان در چند نقطه مورد بررسی قرار داده و در نقاط مختلفی از آن عمل کند. یک سیستم تک‌عامله این امکان را دارا نمی‌باشد [۸].
- نسبت هزینه به کارایی در یک سیستم تک‌عامله نسبت به مدل مشابه چندعامله بیشتر می‌باشد چرا که عمدتاً جمع آوری کلیه امکانات روی یک روبات نسبت به تقسیم آن روی مجموعه‌ای از روبات‌ها نیازمند صرف هزینه بیشتری می‌باشد.

از دیدگاه هوش مصنوعی توزیع شده مسابقه فوتبال روبات‌ها نمونه‌ای از مسائل چندعامله است که در مباحث تحقیقاتی متعددی را می‌توان در آن یافت[۹]. هر تیم مجموعه‌ای از ربات‌ها است که باید به منظور رسیدن به هدف نهایی و مشترک که همان پیروزی در بازی است با یکدیگر همکاری داشته باشند. به ثمر رسانیدن گل و جلوگیری از گل زدن توسط حریف را می‌توان به عنوان زیرمجموعه‌ای از هدف اصلی پیروزی در نظر گرفت. عوامل در شرایطی که درک درستی از محیط ندارند و این دید از محیط اطراف محلی است، باید برای بهینه کردن میزان کارایی خود در محیط تلاش کنند. علاوه بر آن باید در یک محیط پویا بصورت زمان حقیقی عمل کنند چرا که موفقیت به سرعت و انعطاف در عملکرد وابسته است. از آنجا که هر دو تیم با هدف پیروزی فعالیت می‌کنند لذا همواره می‌توان تیم مقابل را به عنوان تهدیدی در رسیدن به هدف نهایی در نظر گرفت.

همهٔ آنچه در بالا بدان اشاره شد در مورد شبیه‌سازی فوتبال نیز قابل تعمیم است. کارگزار شبیه‌سازی تمام ویژگی‌های مطرح شده در بالا ازجمله اعمال خطا در سیستم، محدودیت در بینایی، خطا در حسگرها و عملگرها، محدودیت در ارتباط و توانایی‌های فیزیکی را شبیه‌سازی می‌کند. عوامل باید به محرکها و حوادث پاسخ دهند و تصمیم مناسب را بگیرند. زمان درک محیط و عمل در آن متفاوت است و لذا عمل در محیط را وابسته به تصمیم‌گیری‌های مبتنی بر پیش‌بینی از وقایع می‌کنند. از آنجا که فضای حالات ممکن برای بازی بسیار بزرگ است استفاده از یک روش یکنواخت ممکن نیست و لذا طراحی استراتژی‌های قابل تغییر رکن ضروری در دستیابی به موفقیت می‌باشند.

۱—۳ اهداف و انگیزه‌های شکل‌گیری روبوسینا

این پروژه به عنوان یک طرح تحقیقاتی گسترده و تجربه آموزشی بزرگ مطرح است.

رشد علمی و یکپارچگی تیمی تنها در جریان یک طرح کاری درست و منظم حاصل می‌شود. به عنوان دانشجویان دوره کامپیوتر همواره بر این اعتقادیم که انجام پروژه‌هایی از این قسم در کنار تحصیل تاثیر

[۵] geographical distribution

بسزایی در روند پیشرفت علمی خواهد داشت. طراحی تیمی که توانایی کسب عنوان شایسته در مسابقات روبوکاپ را داشته باشد در واقع تعریف و نتیجه‌ای است بر علایق و اهدافی که می‌توان در جریان یک فعالیت علمی جستجو کرد و این مسابقات فرصت مناسبی است برای تبدیل دانشگاه به بستری واقعی برای تحقیق علاقمندان، چنان که شایسته نام آن است.

تیم روبوکاپ دانشگاه بوعلی سینای همدان، روبوسینا، متشکل از ۵ دانشجوی مقطع کارشناسی این دانشگاه در تاریخ ۴ تیرماه سال ۱۳۸۱ فعالیت خود را در دانشکده‌ی کامپیوتر دانشگاه بوعلی سینا شروع کرد. با توجه به نحوه‌ی کار تیمی و فعالیت‌هایی که تا آن زمان در داخل کشور انجام شده بود تصمیم بر آن شد تا پیاده‌سازی سیستم به شکل کاملاً پایه‌ای و با شروع از جزئی‌ترین موارد انجام شود و اهداف کاری پروژه تحلیل، طراحی و پیاده‌سازی یک تیم از روبات‌های فوتبالیست تعریف شد.

روبرو شدن با محیط ناشناخته و در نوع خود عجیب سیستم شبیه‌ساز و محدودیت‌های زمانی برای آماده نمودن تیمی به منظور شرکت در مسابقات جهانی، هزینه سنگین مطالعاتی و زمانی را به تیم تحمیل نمود. اما سرانجام گروه کاری توانست از شناخت و آشنایی خارج شده و وارد مرحله فعالیت‌های علمی و مطالعاتی شود. استفاده از سیستم عامل لینوکس به عنوان یک محیط با توابع سیستمی قدرتمند و قابلیت اطمینان بالا در مدیریت حافظه و پایداری عمل، نخستین گام در ایجاد تیم بود.

پیاده‌سازی یک سیستم بزرگ مانند یک تیم از روبات‌های هوشمند فعالیتی است که در جریان آن بروز خطاهای مختلف امری اجتناب ناپذیر است و لذا بهترین روش در توسعه چنین سیستمی روش توسعه افزایشی است که از روش‌های شناخته شده و متداول در عرصه تولید نرم‌افزار بوده و پیش از این توسط تیم‌های مطرح دیگری در مسابقات جهانی روبوکاپ نیز مورد استفاده قرار گرفته است [۵]. روش توسعه افزایشی بر این اصل استوار است که سیستم باید به نحوی طراحی شود که قابلیت اجرا داشته باشد هرچند که این اجرا بسیار ضعیف و دورتر از آن چیزی باشد که در نهایت هدف اصلی سیستم را تشکیل می‌دهد. در ادامه این سیستم می‌تواند به منظور دستیابی به هدف نهایی اصلاح شود، در عین حال که در هر لحظه یک سیستم قابل اجرا را در اختیار سازنده قرار می‌دهد. بعلاوه مدیریت کدی که بطور موازی توسط اعضای مختلف تیم در حال تغییر و تکامل بود نیاز به استفاده از روش‌های نرم‌افزاری خصوصا در مدیریت موازی نسخه‌های تولید شده را ضروری می‌نمود که در نهایت استفاده از سیستم موازی نسخه‌گذاری [٦] را موجب شد. روند افزایشی در توسعه کد و گسترش بخش‌های مختلف سیستم در نهایت ساختاری را بوجود آورد که توانایی شرکت در مسابقات رسمی را دارا بود.

۱ — ۴ راهنمای بخش‌های رساله

در این رساله سعی می‌کنیم تا شکل‌گیری تیم روبوسینا را از تحلیل و طراحی تا پیاده‌سازی و نتایج، مورد بررسی قرار دهیم. توضیح جامع روی الگوهای تصمیم‌گیری و روش‌های همکاری در سطوح بالا و عملکردهای فردی در سطوح پایین با توجه به شباهتی که بین کارگزار دوبعدی و کارگزار سه‌بعدی وجود دارد می‌تواند برای توسعه و اعمال الگوریتم‌ها در محیط سه‌بعدی نیز مؤثر و قابل استفاده باشد. این رساله در بخش‌های زیر گردآوری شده است که مطالب هر بخش به اختصار به شرح زیر می‌باشند:

---

[٦] Concurrent Versioning System (CVS)

- فصل ۲ (سیستم شبیه‌سازی) : در این فصل مفاهیم مربوط به کارگزار شبیه‌ساز روبوکاپ مورد مطالعه و بررسی قرار می‌گیرد و مفاهیمی از قبیل توصیف فرایند مسابقه، قوانین کارگزار، کاربر مربی، و اطلاعات مبادله شده نظیر اطلاعات حسگرها و نحوه کنترل زمین بازی مورد معرفی و تحلیل قرار می‌گیرد.
- فصل ۳ (جنبه‌های بارز تیم روبوسینا) : از آنجا که کد تیم روبوسینا از پایه شکل گرفته است لذا بسیاری از مفاهیم مطرح شده در این تیم جدید بوده و پیش از آن توسط هیچ تیم دیگری پیاده‌سازی یا طراحی نشده است. در این فصل تمرکز اصلی بر روی بخش‌هایی از تیم است که با ایده‌ها و الگوریتم‌های جدید ایجاد شده‌اند. از این مجموعه می‌توان به معماری سیستم، روش همزمان شدن با کارگزار، الگوریتم مکان‌یابی، و اَعمال سطح بالای یک بازیکن در سطح عملکرد فردی یا عملکرد تیمی اشاره کرد.
- فصل ۴ (گذری بر الگوریتم‌های یادگیر) : در فصل ۴ از این رساله مفاهیم اولیه در جهت آشنایی با الگوریتم‌های مهم هوش‌مصنوعی را مورد مطالعه قرار می‌دهیم. این الگوریتم‌ها اسلوب‌ها و تکنیک‌های استفاده شده برای آموزش یک عامل روبوسینا را معرفی می‌کنند به طوری که در فصل‌های بعد بتوان به کمک آن‌ها به روش‌های آموزش یک بازیکن روبوسینا پرداخت.
- فصل ۵ (عامل روبوسینا و یادگیری) : با استفاده از مطالب فصل قبل در این بخش ۲ مهارت آموزش داده شده به یک عامل روبوسینا را که عبارتند از شوت به سمت دروازه و پاس به سایر بازیکنان را مورد بررسی و توضیح قرار خواهیم داد. این دو عمل به ترتیب با استفاده از الگوریتم‌های شبکه عصبی و درخت‌های تصمیم و با توجه به شرایط و ویژگی‌های موجود در محیط کارگزار به روبات آموزش داده شده‌اند.
- فصل ٦ (پیشنهادات و نتیجه‌گیری) : این بخش قسمت پایانی رساله می‌باشد و در آن استفاده از یک روش یادگیری تقویتی برای استفاده در لایه‌ی تصمیم‌گیری و بر پایه‌ی مفاهیم پاداش و جزا ارائه شده است که این روش در تیم روبوسینا در حال پیاده‌سازی می‌باشد. در نهایت یک جمع‌بندی کلی روی مفاهیم مطرح شده در این رساله پایان‌بخش مطالب خواهد بود.

۲

سیستم شبیه‌سازی

سیستم شبیه‌ساز از سه بخش اصلی تشکیل شده است که عبارتند از :

- کارگزار فوتبال (the Soccer Server)
- نمایشگر فوتبال (the Soccer Monitor)
- نمایشگر مجدد بازی (the LogPlayer)

سیستم کارگزار نرم‌افزار اصلی در طراحی و مدلسازی محیط فوتبال می‌باشد. تمام اطلاعات مربوط به حسگرها، عملگرها، الگوریتم‌های ایجاد خطا و الگوهای مورد استفاده در برقراری ارتباط میان عامل‌ها توسط این سیستم تولید، پردازش و بین عوامل منتقل می‌شوند. مجموعه شبیه‌ساز دو نمایشگر بازی را نیز به همراه دارد که بطور مستقیم با کارگزار تقابل می‌کنند و اطلاعات ایجاد شده از سوی کارگزار را بصورت online یا offline نمایش می‌دهند که روش نمایش online در هنگام انجام یک مسابقه واقعی و روش نمایش offline در زمان تست و یا مطالعه روی نحوه عملکرد یک تیم در یک بازی انجام شده مورد استفاده قرار می‌گیرد. علاوه بر نمایشگرها این مجموعه شامل یک نمایشگر مجدد بازی است که اطلاعات مربوط به یک بازی انجام شده را مجدداً با اتصال به نمایشگر به تصویر می‌کشد و ارتباط آن با نمایشگر سبب نمایش بازی به صورت offline می‌گردد. همانطور که اشاره شد این روش به منظور مطالعه و بررسی عملکرد یک تیم کاربرد دارد. نسخه در حال استفاده سیستم شبیه‌ساز در حال حاضر ۱۰.x می‌باشد. در این فصل به بررسی و مطالعه اجزا تشکیل دهنده سیستم شبیه‌ساز می‌پردازیم. در بخش ۲—۱ کارگزار فوتبال را بطور جزئی مورد مطالعه قرار می‌دهیم. در این بخش فرایند انجام یک مسابقه به همراه قوانین مسابقه، و ویژگی‌های زمین بازی مطالعه می‌شوند، در بخش ۲—۲ مدلهای حسی، حرکتی و نحوه عملکرد بازیکن‌ها بطور خلاصه بررسی می‌شود. در بخش ۲—۳ به بررسی مُدهای بازی در سیستم کارگزار خواهیم پرداخت، در بخش ۲—۴ بازیکنان نامتشابه، در بخش ۲—۵ مربی، و در نهایت در بخش ۲—۶ مطالب این عنوان شده در این فصل را مورد بررسی قرار می‌دهیم.

۲—۱ کارگزار فوتبال

کارگزار فوتبال (Soccer Server) سیستمی است که عامل‌های مختلف یک تیم را، که برنامه‌های نرم‌افزاری نوشته شده در زبان‌های مختلف هستند، قادر می‌سازد تا در مقابل یکدیگر یک بازی فوتبال انجام دهند. ارتباط به صورت اتصال کاربر به کارگزار (Client-Server) انجام می‌شود بدین صورت که کارگزار فوتبال یک زمین بازی فوتبال را به صورت مجازی فراهم آورده و حرکت بازیکنان و توپ را در آن شبیه‌سازی می‌کند. ارتباط میان هر کاربر با کارگزار از طریق پروتکل UDP/IP برقرار می‌شود و بنابراین ارتباط از طریق هر زبانی که UDP/IP را پشتیبانی کند میسّر می‌باشد[?]. هر بازیکن می‌تواند از طریق پورت UDP[1] به کارگزار متصل شده و اطلاعات را مبادله کند. هر برنامه نرم‌افزاری تنها قادر است در نقش مغز تصمیم‌گیر برای ۱ بازیکن عمل نماید و لذا برای اجرای ۱۱ بازیکن باید ۱۱ برنامه مختلف اجرا شوند که باید بر حسب نحوه عملکرد هر بازیکن سازمان‌دهی شوند.

کارگزار به منظور شبیه‌سازی زمین بازی مجموعه‌ای از داده‌های محیطی را مورد استفاده قرار می‌دهد که آشنایی با این اطلاعات در جهت ایجاد درک مناسب از زمین بازی برای روبات و عملکرد در این زمین بسیار سودمند خواهد بود. جدول ۲—۱ لسیت تعدادی از پارامترهای ضروری کارگزار را نمایش می‌دهد.

۲—۱—۱ قوانین مسابقه

قوانین مسابقه شامل دو مجموعه از قوانین هستند : قوانینی که به شکل خودکار و از سوی کارگزار اعمال می‌شوند و قوانینی که توسط سیستم خودکار قابل تشخیص نبوده و لذا از طرف داور انسانی بر بازی اعمال می‌شوند.

- قوانین کارگزار : اعمال قوانینی که از طریق محاسباتی و یا منطقی قابل تشخیص می‌باشند از سوی کارگزار انجام می‌شود. از قوانین محاسباتی می‌توان به offside، حفظ فاصله ۹/۱۵m از سوی بازیکنان هر تیم در هنگام ضربه آزاد تیم مقابل، و یا تنظیم زمان ۵ دقیقه‌ای یا ۳۰۰۰ سیکلی در هر نیمه از بازی اشاره نمود و از مجموعه قوانین منطقی نیز می‌توان کنترل وضعیت در هر یک از مدهای مختلف بازی نظیر corner_kick، kick_in، goal_kick، و یا play_on را نام برد. در واقع می‌توان کارگزار را به عنوان یک ماشین قطعی و متناهی (Deterministic Finite Automata) در برخورد با خطاهایی دانست که امکان تشخیص آنها از سوی کارگزار وجود دارد.
- قوانین انسانی : این دسته از قوانین از نظر عملکرد منطقی ظاهر درستی دارند اما روند اجرای سالم بازی را دچار اختلال می‌نمایند و ار آنجا که از سوی کارگزار قابل تشخیص نمی‌باشند داور انسانی در چنین مواردی اعمال نظر می‌کند و از این رو در این رساله آنها را قوانین انسانی می‌نامیم. احاطه کردن توپ، پر کردن دروازه با تعداد زیادی از بازیکنان، سد کردن راه عبور سایر بازیکنان از روی عمد، استفاده مکرر از دستورات catch و kick، و اشباع شبکه با استفاده از پیام‌های زیاد[?] از جمله خطاهایی هستند که توسط داور انسانی داوری می‌شوند.

[1] در محیط شبیه‌سازی شدهٔ سه‌بعدی برقراری ارتباط بین کارگزار و کاربر از طریق پورت TCP ایجاد می‌شود.

جدول ۲—۱: پارامترهای کارگزار

نام پارامتر	مقدار پیش‌فرض	مقدار فعلی در سرور	توضیح
goal_width	۷٫۲۳	۱۴٫۰۲	عرض دروازه
player_size		۰٫۳	اندازه بازیکن
player_decay		۰٫۴	اُفت بازیکن
player_rand		۰٫۱	
player_weight		۶۰٫۰	وزن بازیکن
player_speed_max		۱٫۰	بیشترین سرعت بازیکن
player_accel_max		۱٫۰	بیشترین شتاب بازیکن
stamina_max		۴۰۰۰٫۰	بیشترین استقامت بازیکن
stamina_inc_max		۴۵٫۰	بیشترین افزایش استقامت بازیکن
recover_dec_thr		۰٫۳	حد کاهش بهبود انرژی
recover_min		۰٫۵	کمترین حد بهبود انرژی
recover_dec		۰٫۰۰۲	کاهش بهبود انرژی
effort_dec_thr		۰٫۳	حد کاهش تلاش
effort_min		۰٫۶	کمترین حد تلاش
effort_dec		۰٫۰۰۵	کاهش تلاش
effort_inc_thr		۰٫۶	حد افزایش تلاش
effort_inc		۰٫۰۱	افزایش تلاش
ball_size		۰٫۰۸۵	اندازه توپ
ball_decay		۰٫۹۴	اُفت توپ
ball_rand		۰٫۰۵	
ball_weight		۰٫۲	وزن توپ
ball_speed_max		۲٫۷	بیشترین سرعت توپ
ball_accel_max		۲٫۷	بیشترین شتاب توپ
dash_power_rate		۰٫۰۰۶	نرخ نیروی دویدن
kick_power_rate		۰٫۰۲۷	نرخ نیروی ضربه به توپ
kickable_margin		۰٫۷	آستانه فضای قابل شوت
max_power		۱۰۰	بیشترین انرژی
min_power		−۱۰۰	کمترین انرژی
maxmoment		۱۸۰	بیشترین گشتاور
minmoment		−۱۸۰	کمترین گشتاور
maxneckmoment		۱۸۰	بیشترین گشتاور گردن
minneckmoment		−۱۸۰	کمترین گشتاور گردن
maxneckang		۹۰	بیشترین زاویه گردن
minneckang		−۹۰	کمترین زاویه گردن
visible_angle		۹۰٫۰	زاویه بینایی پیش‌فرض
visible_distance		۳٫۰	فضای محسوس برای بازیکن
quantize_step		۰٫۱	پارامتر پله‌ای کردن فاصله برای اشیا متحرک
quantize_step_l		۰٫۰۱	پارامتر پله‌ای کردن فاصله برای اشیا ثابت
port		۶۰۰۰	پورت اتصال بازیکن
coach_port		۶۰۰۲	پورت اتصال آموزشگر (trainer)
olcoach_port		۶۰۰۱	پورت اتصال مربی online
simulator_step		۱۰۰	بازه زمانی شبیه‌ساز [واحد:msec]
send_step		۱۵۰	بازه زمانی اطلاعات بینایی [واحد:msec]
recv_step		۱۰	بازه زمانی پذیرش دستورات [واحد:msec]
sense_body_step		۱۰۰	بازه زمانی اطلاعات فردی [واحد:msec]
say_msg_size		۱۰	اندازه رشته پیغام say [واحد:byte]
hear_max		۲	بیشترین تعداد پیام قابل شنیدن در هر سیکل
hear_inc		۱	افزایش شنوایی

نام پارامتر	مقدار پیش‌فرض	مقدار فعلی در سرور	توضیح
hear_decay		۲	اُفت شنوایی
catch_ban_cycle		۵	سیکل‌های ممنوعیت دروازبان بین دو catch متوالی
inertia_moment		۵٫۰	گشتاور اینرسی برای turn
half_time		۳۰۰	اندازه یک نیمه از بازی بر حسب ثانیه
drop_ball_time		۲۰۰	تعداد سیکل‌های انتظار در بازی پیش از drop ball
wind_dir	۰٫۰	۰٫۰	جهت باد
wind_force	۱۰٫۰	۰٫۰	نیروی باد
wind_rand	۰٫۳	۰٫۰	بازه ضریب تصادفی قابل افزایش به باد
wind_random	false		فاکتور باد تصادفی است

۲—۱—۲ ویژگی‌های زمین بازی

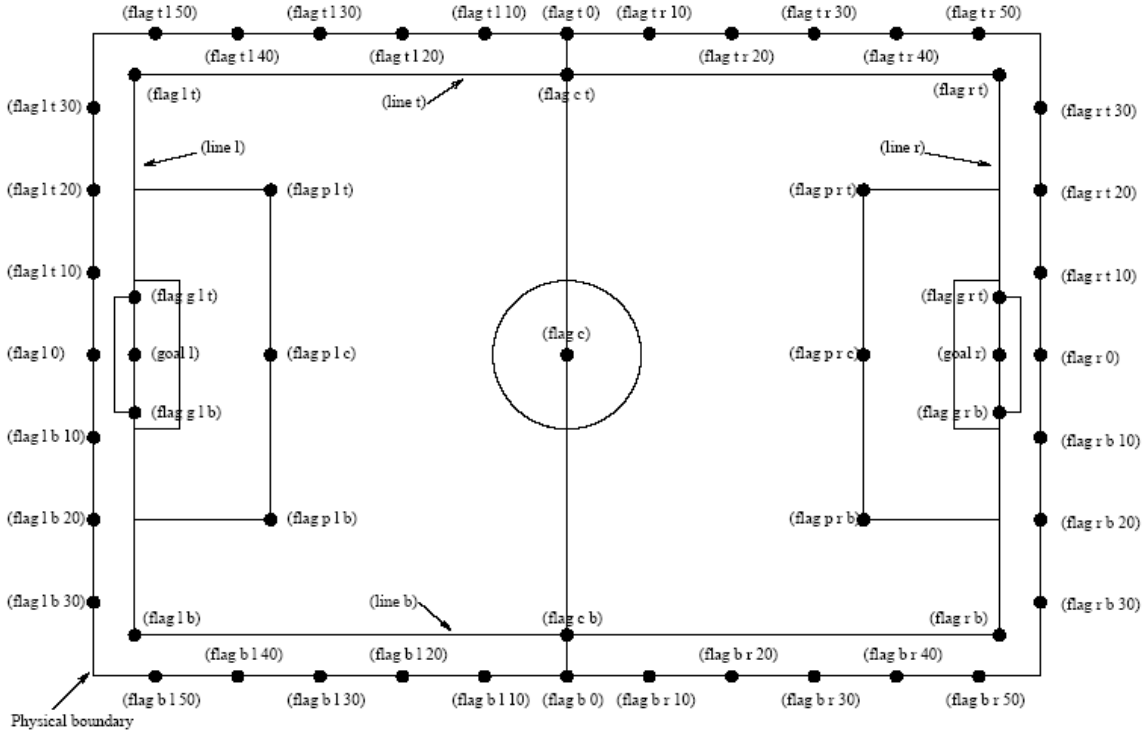
زمین بازی یک فضای شبیه‌سازی شده دوبعدی است که دارای ابعاد ۱۰۵×۶۸ بوده و عرض هر دروازه در آن ۱۴٫۰۲m یعنی حدوداً دو برابر عرض دروازه واقعی می‌باشد. در این زمین توپ و بازیکنان به شکل دایره مدل شده‌اند و تمامی زوایا و فواصل نسبت به مرکز این دایره‌ها سنجیده می‌شود. توپ و بازیکنان اشیا متحرک زمین را می‌سازند. مرکز محور مختصات بر مرکز زمین منطبق است و محور x ها از ۵۲٫۵— تا ۵۲٫۵ و محور y ها از ۳۴— تا ۳۴ فضای زمین بازی را می‌پوشانند. در اطراف زمین بازی ۵۹ پرچم با موقعیت‌های مشخص و از پیش تعریف شده وجود دارد که بازیکن را در تعیین موقعیت خود توانا می‌سازند (شکل ۲—۱).

۲—۲ مدل‌های حسی، حرکتی، عملی

برای اینکه یک عامل بتواند در محیط شبیه‌سازی عملکردی مشابه با محیط واقعی داشته باشد باید بتواند در این محیط با کاگزار شبیه‌سازی تعامل دوسویه داشته باشد. به بیان بهتر باید توانایی دریافت اطلاعات سودمند، که در نقش محرک برای انجام عملیات تصمیم‌گیری می‌باشند، و انجام عمل مناسب در حالت فعلی، که پاسخی بر محرک محیطی است، را دارا باشد. ایجاد چنین تقابل دوطرفه‌ای بین کارگزار با کاربر و بالعکس باید از سوی سیستم کارگزار پشتیبانی شود. در ادامه روشهای تعریف شده‌ی برقراری این ارتباط را بررسی می‌کنیم.

۲—۲—۱ مدل‌های حسی در کارگزار

کارگزار شبیه‌سازی به منظور انتقال اطلاعات محیط به بازیکنان سه مدل حسی را مورد استفاده قرار می‌دهد که یک عامل را در درک محیط اطراف، وضعیت خود، و شرایط حاکم بر بازی توانا می‌کنند. این سه مدل



شکل ۲—۱: پرچم‌ها در زمین بازی

حسی را حسگرهای بینایی[۲]، حسگرهای شنوایی[۳]، و حسگرهای جسمی[۴] تشکیل می‌دهند. در ادامه اطلاعات مبادله شده در هریک از این حسگرها را مورد بررسی قرار می‌دهیم.

حسگرهای بینایی

اطلاعات قابل رؤیت در زمین بازی از طریق حسگرهای بینایی به بازیکن منتقل می‌شوند. این اطلاعات شامل تمام اشیأ ثابت و متحرک قابل مشاهده در زمین هستند که بازیکنان، توپ و پرچم‌های موجود در زمین بازی را شامل می‌شوند. تمام اطلاعات بصورت قطبی و نسبت به مکان بازیکن دریافت می‌شوند و از این رو بازیکن قادر به درک مکان مطلق خود در زمین بازی نمی‌باشد. برای بدست آوردن تقریب درست از مکان خود، بازیکن باید از پرچم‌های موجود در زمین استفاده کند (شکل ۲—۱) که در بخش ۳—۳ بدان خواهیم پرداخت. هر بازیکن علاوه بر فضایی که با توجه به زاویه گردن و زاویه دیداری می‌تواند مشاهده کند، قادر است دایره‌ای به شعاع `visible_distance` را در اطراف خود و خارج از فضای قابل رؤیت درک کند. اطلاعات دیداری در فواصل زمانی `send_step` به بازیکن منتقل می‌شوند که این

[۲] Visual Sensros
[۳] Aural Sensors
[۴] Body Sensors

بازهٔ زمانی با توجه به پارامترهای مربوط به کیفیت، و فضای دیداری قابل تغییر است. هر بازیکن می‌تواند کیفیت، فضای دیداری، و تکرّر زمان دریافت اطلاعات را با استفاده از امکاناتی که در اختیار دارد و با توجه به شرایط بازی تنظیم نماید. بازه زمانی دریافت اطلاعات (ViewQuality) بر حسب میلی ثانیه تنظیم می‌شود و می‌تواند یکی از چهار مقدار ۳۷٫۵ms، ۷۵ms، ۱۵۰ms، و یا ۳۰۰ms را دارا باشد که بطور پیش فرض از سوی کارگزار به مقدار ۱۵۰ms تنظیم شده است. کیفیت اطلاعات دریافتی میتواند بالا (high) و یا پائین (low) باشد و مقدار پیش فرض آن high می‌باشد. بازیکن می‌تواند کیفیت اطلاعات دریافتی را در هر سیکل از بازی به نوع دلخواه تغییر دهد. فضای دیداری (ViewWidth) قطاعی از یک دایره با شعاع در حدود ۶۰m است که زاویه آن توسط بازیکن بین سه مقدار narrow(۴۵درجه)، normal(۹۰درجه) ، و wide(۱۸۰درجه) قابل تنظیم بوده و مقدار اولیه آن از سوی سرور برابر با normal تنظیم شده است. جهت گردن بازیکن جهت نیمساز این زاویه را مشخص می‌کند و با دورشدن از مرکز این قطاع که همان محل بازیکن است، به تدریج از کیفیت اطلاعات دریافتی کاسته می‌شود.

همانطور که پیشتر اشاره شد فرکانس دریافت اطلاعات بینایی از طریق دو پارامتر کیفیت و بازهٔ دیداری قابل تنظیم می‌باشند. فرکانس دریافت اطلاعات از رابطه زیر تعیین می‌شود:

$$\boldsymbol{view\_frequency} = send\_step * \boldsymbol{view\_width\_factor} * \boldsymbol{view\_quality\_facotr}$$

پارامتر *view_width_factor* با توجه به مقدار narrow، normal، یا wide که از سوی بازیکن تعیین می‌شود به ترتیب مقادیر ۰٫۵، ۱٫۰، یا ۲٫۰ را می‌گیرد. به همین ترتیب *view_quality_factor* هم می‌تواند در ازای high یا low به ترتیب مقادیر ۱٫۰ یا ۰٫۵ را دارا باشد. با توجه به این پارامترها کوتاه‌ترین فاصله زمانی از رابطه ۳۷٫۵ = ۰٫۵ × ۰٫۵ × ۱۵۰ با تنظیم *view_width_factor* به narrow و *view_quality_factor* به low و بیشترین بازه از رابطه ۳۰۰ = ۲٫۰ × ۱٫۰ × ۱۵۰ با تنظیم *view_width_factor* به wide و *view_quality_factor* به high حاصل می‌شوند.

اطلاعات بینایی در قالب زیر به بازیکن منتقل می‌شوند:

(see زمان اطلاعات‌شیی$^{+}$)

که در آن

زمان = شماره سیکل بازی در هنگام انتقال داده

اطلاعات‌شیی = ([[زاویه‌سر زاویه‌بدن] سرعت‌زاویه‌ای تغییرفاصله] زاویه فاصله نام‌شیی)

= (زاویه نام‌شیی)

نام‌شیی = (p [“نام‌تیم” [شماره‌بازیکن] [goalie]]]) وجود goali نشان می‌دهد که بازیکن دروازبان است

= (b)

= (g[l|r])

= (f c)

= (f [l|c|r] [t|b])

= (f p [l|r] [t|c|b])

= (f g [l|r] [t|b])

= (f [l|r|t|b] 0)

= (f [t|b] [l|r] [10|20|30|40|50])

= (f [l|r] [t|b] [10|20|30])

= (l [l|r|t|b])

= (B)

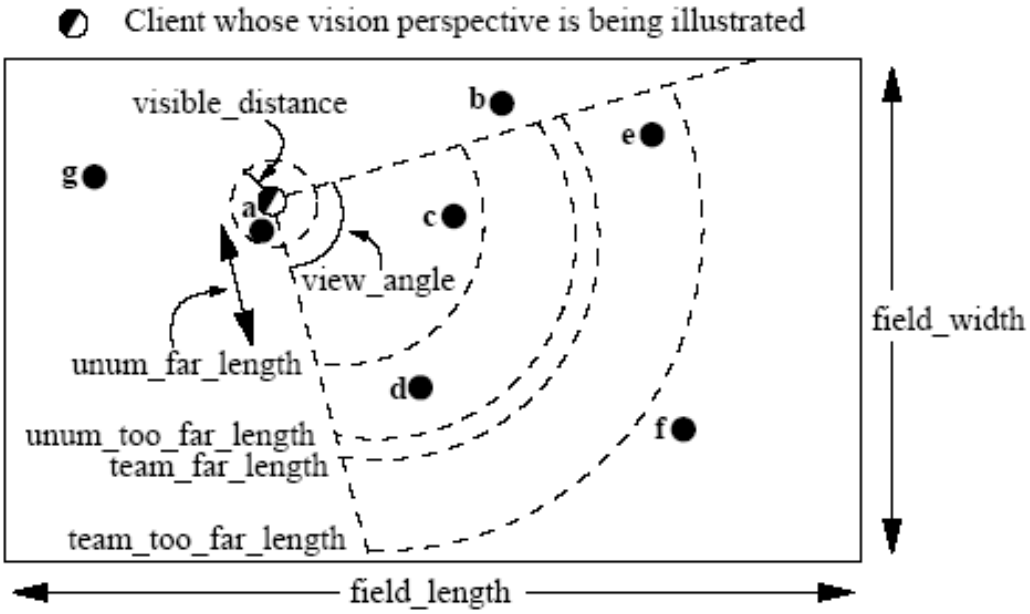
= (F)

= (G)
= (P)
عدد حقیقی مثبت = فاصله
بین ۱۸۰— و ۱۸۰ درجه = زاویه
عدد حقیقی = تغییرزاویه
عدد حقیقی = سرعت‌زاویه
بین ۱۸۰— و ۱۸۰ درجه = زاویه‌بدن
بین ۱۸۰— و ۱۸۰ درجه = زاویه‌سر
رشته = نام‌تیم
بین ۱ تا ۱۱ = شماره‌بازیکن

نام شیئ همواره کاراکتری را به همراه دارد که نوع شیئ را مشخص می‌کند: p برای بازیکن، b برای توپ، f برای پرچم‌ها، g برای گل‌ها و l برای خطوط استفاده می‌شوند. کاراکترهای P، B، F، و G در شرایطی استفاده می‌شوند که اشیا در فاصله visible_distance و بیرون از فضای دیداری بازیکن قرار دارند و توسط بازیکن دیده نمی‌شوند اما درک می‌شوند.

همانطور که اشاره شد با دور شدن از مرکز قطاع دیداری به تدریج از کیفیت اطلاعاتی که توسط بازیکن دریافت می‌شود کاسته می‌شود ضمن اینکه در شرایطی که کیفیت دیداری از سوی بازیکن به low تنظیم می‌شود فقط نام شیئ و زاویه نسبی آن به بازیکن ارسال می‌شود اما در شرایطی که این پارامتر به high تنظیم شود با توجه به نوع شیئ و فاصله با آن اطلاعات جامع‌تری در اختیار بازیکن قرار می‌گیرد. روشی که کارگزار برای اعمال این تغییر کیفیت بکار می‌برد به این صورت است که اگر فاصله بازیکن تا شیئ مورد نظر را dist در نظر بگیریم :

- برای پرچم‌ها و خطوط زمین اطلاعات همواره شامل نام، زاویه و فاصله با این اشیا می‌باشد. در مورد خطوط اطراف زمین این فاصله و زاویه از تقاطع نیمساز زاویه دیداری با خط مربوطه ایجاد می‌شوند.
- اگر شیئ دیده شده p (بازیکن) باشد کیفیت اطلاعات از مقایسه dist با ۴ پارامتر داخلی سرور تعیین می‌شوند که team_far_length, unum_too_far_length, unum_far_length و $team\_too\_far\_length$ می‌باشند. نوع اطلاعات ارسالی از روی روابط زیر استخراج می‌شوند:
  - ⋆ اگر $dist \leq unum\_far\_length$: نام تیم به همراه شماره بازیکن و فاصله، زاویه، تغییرفاصله، سرعت‌زاویه‌ای، زاویه بدن، و زاویه سر در پیغام بینایی وجود دارند.
  - ⋆ اگر $unum\_far\_length < dist \leq unum\_too\_far\_length$: نام تیم به همراه فاصله، و زاویه وجود دارند اما احتمال دریافت اطلاعات از شماره بازیکن، تغییرفاصله، سرعت‌زاویه‌ای، زاویه بدن، و زاویه سر بازیکن دیده شده با نزدیک شدن dist به unum_too_far_length از ۱ به ۰ میل می‌کند.
  - ⋆ اگر $dist \leq unum\_too\_far\_length = team\_far\_length$: تغییرفاصله، سرعت‌زاویه‌ای، زاویه بدن، و زاویه سر بازیکن دیده شده در پیغام بینایی وجود ندارند اما نام تیم به همراه فاصله، و زاویه وجود دارند.
  - ⋆ اگر $unum\_too\_far\_length = team\_far\_length < dist < team\_too\_far\_length$: شماره بازیکن در پیغام دیداری وجود ندارد و احتمال دیده شدن نام تیم بازیکن نیز با نزدیک شدن dist به team_too_far_length به طور خطی به ۰ میل می‌کند.
  - ⋆ اگر $dist > team\_too\_far\_length$: نام تیم قابل تشخیص نیست و بازیکن دیده شده دارای هویت مجهول است.



شکل ۲—۲: زاویه دیداری یک بازیکن

- در شرایطی که اطلاعات مربوط به b (توپ) باشد نیز روند ارسال اطلاعات با توجه به فاصله، مشابه قبل است.
  - ⋆ اگر $dist \leq unum\_far\_length$: فاصله، زاویه، تغییرفاصله، و سرعت‌زاویه‌ای توپ از طریق پیغام بینایی منتقل می‌شود.
  - ⋆ اگر $unum\_far\_length < dist \leq unum\_too\_far\_length$: فاصله و زاویه همواره در پیغام وجود دارند اما احتمال وجود پارامترهای تغییرفاصله و سرعت‌زاویه‌ای به شکل خطی به سمت ۰ میل می‌کند.
  - ⋆ اگر $dist \leq unum\_too\_far\_length = team\_far\_length$: فاصله و زاویه همواره در پیغام وجود دارند اما پارامترهای تغییرفاصله و سرعت‌زاویه‌ای از لیست اطلاعات منتقل شده حذف می‌شوند.

مقادیر فعلی مورد استفاده برای پارامترهای $unum\_too\_far\_length$, $team\_far\_length$ $team\_too\_far\_length$ $unum\_far\_length$, در جدول ۲—۲ آمده‌اند.

شکل ۲—۲ نمایی از یک بازیکن را ارائه می‌دهد که وسعت دید آن با توجه به پارامترهای مورد بحث مشخص شده است. زاویه سر بازیکن در جهت نیمساز این زاویه دیداری است[۱۰].

مقادیر فاصله، زاویه، تغییرفاصله، سرعت‌زاویه‌ای، زاویه‌بدن، و زاویه‌گردن با استفاده از فرمول‌های زیر از طرف سرور محاسبه می‌شوند.

$$
\begin{aligned}
p_{rx} &= p_{xt} - p_{xo} \\
p_{ry} &= p_{yt} - p_y o \\
v_{rx} &= v_{xt} - v_{xo} \\
v_{ry} &= v_{yt} - v_{yo} \\
\text{فاصله} &= \sqrt{p_{rx}^{۲} + p_{ry}^{۲}} \\
\text{زاویه} &= \arctan(p_{ry}/p_{rx}) - a_{۰} \\
e_{rx} &= p_{rx}/\text{فاصله} \\
e_{ry} &= p_{ry}/\text{فاصله} \\
\text{تغییرفاصله} &= (v_{rx} \cdot e_{rx}) + (v_{ry} \cdot e_{ry}) \\
\text{تغییرزاویه} &= [(-(v_{rx} \cdot e_{ry}) + (v_{ry} \cdot erx))/Distance].(۱۸۰/\pi) \\
\text{زاویه‌بدن} &= body\_dir\_abs - a_{۰} \\
\text{زاویه‌گردن} &= neck\_dir\_abs - a_{۰}
\end{aligned}
$$

که در روابط بالا (p_{xt}, p_{yt}) و (v_{xt}, v_{yt}) به ترتیب مکان و سرعت شیئ هدف و (p_{xo}, p_{yo}) و (v_{xo}, v_{yo}) به ترتیب مکان و سرعت بازیکن بیننده را نشان می‌دهند. $a_{۰}$ جهت مطلق سر بازیکن بیننده را نشان می‌دهد. (p_{rx}, p_{ry}) و (v_{rx}, v_{ry}) به ترتیب مکان و سرعت نسبی شیئ هدف و (e_{rx}, e_{ry}) بردار یکه در جهت مکان نسبی می‌باشند. زاویه‌سر و زاویه‌گردن تنها در شرایطی اضافه می‌شوند که شیئ رؤیت شده بازیکن باشد.

با هدف تطبیق مدل شبیه‌سازی به مدل واقعی خطا به سیستم افزوده شده است که این خطا با استفاده از پله‌ای کردن[۵] داده‌های ارسالی از سوی سرور حاصل می‌شود. به عنوان نمونه برای افزودن خطا به فاصله یک از رابط زیر استفاده می‌شود:

$$Q\_Distance = Quantize(exp(Quantize(ln(Distance), StepValue)), ۰/۱)$$

که در آن $StepValue$ از پارامترهای داخلی سرور بوده و می‌تواند دو مقدار $quantize\_step$، برای اشیا متحرک، و $l\_quantize\_step$، برای اشیا ثابت، را دارا باشد. مقادیر مربوط به $quantize\_step$ و $l\_quantize\_step$ در جدول ۲—۲ آمده‌اند. $Distance$ فاصله حقیقی و $Q\_Distance$ فاصله با اعمال نویز می‌باشند. رابطه $Quantize$ از روی فرمول زیر محاسبه می‌شود:

$$Quantize(V, Q) = rint(V/Q) \cdot Q$$

تابع $rint$ مقدار خروجی را به نزدیک‌ترین عدد صحیح گرد می‌کند. با توجه به روابط بالا می‌توان دریافت که حداکثر میزان خطای اعمال شده برای فاصله یک بازیکن برابر با ۰/۱ این فاصله خواهد بود یعنی "**فاصله** × ۰/۱ ± **فاصله**" و در مورد اشیا ثابت "**فاصله** × ۰/۰۱ ± **فاصله**". برای اعمال نویز در سرعت، زاویه و سرعت‌زاویه‌ای از مجموعه روابط زیر استفاده می‌شود:

$$
\begin{aligned}
Q\_DistChange &= Q\_Distance \cdot Quantize(DistChange/Distance, ۰/۰۲) \\
Q\_Direction &= Quantize(Direction, ۱/۰) \\
Q\_DirChange &= Quantize(DirChange, ۰/۱)
\end{aligned}
$$

که $DistChange$، $Direction$ و $DirChange$ به ترتیب سرعت، زاویه و سرعت زاویه‌ای و $Q\_DistChange$، $Q\_Direction$ و $Q\_DirChange$ همین مقادیر پس از اعمال خطا می‌باشند.

---

[۵] Quantize

جدول ۲—۲: پارامترهای کارگزار برای استفاده در حسگرهای بینایی

پارامتر	مقدار
send_step	۱۵۰
team_far_length	۴۰/۰
visible_angle	۹۰/۰
team_too_far_length	۶۰/۰
visible_distance	۳/۰
quantize_step	۰/۱
unum_far_length	۲۰/۰
quantize_step_l	۰/۰۱
unum_too_far_length	۴۰/۰

حسگرهای شنوایی

سیستم کارگزار سرور یک محیط شلوغ با پهنای باند کم را شبیه‌سازی می‌کند که در آن تمام عامل‌ها می‌توانند از طریق یک خط ارتباطی مشترک با قابلیت اطمینان پائین با هم ارتباط برقرار نمایند[۱۱]. در چنین محیطی بازیکنان قادر هستند تا اطلاعاتی را که از سوی سایر عامل‌ها در شبکه ارسال می‌شود دریافت نمایند و به تحلیل اطلاعات دریافتی بپردازند. اطلاعات دریافتی به سرعت و بدون هیچ تأخیری در سیستم منتشر می‌شود و همه بازیکنان توانایی شنیدن آن را دارا می‌باشند. قالب پیام دریافتی به شکل زیر است:

("پیام" فرستنده زمان **hear**)

که در آن

زمان	=	شماره سیکل بازی در هنگام انتقال داده
فرستنده	=	online_coach_l | online_coach_r
	=	self
	=	referee
	=	
	=	پیام

مربی سمت چپ یا سمت راست
زمانی که فرستنده خود بازیکن است
زمانی که فرستنده داور است
زاویه دریافت پیغام در صورتی که ارسال کننده بازیکن باشد
محتویات پیام ارسالی از سوی فرستنده‌ها

حجم پیغام قابل انتقال به پارامتر say_msg_size در کارگزار وابسته است و در نسخه فعلی برابر با ۱۰ بایت می‌باشد. پارامتر $hear\_max$ بیشترین ظرفیت اطلاعات قابل شنیدن توسط بازیکن را مشخص می‌کند و با هر بار شنیدن از این ظرفیت به اندازه $hear\_decay$ کم می‌شود.در شروع هر سیکل از بازی ظرفیت شنیداری به اندازه $hear\_inc$ افزوده می‌شود تا به بیشینه مقدار $hear\_max$ می‌رسد. ظرفیت شنیداری بازیکنان تیم خودی از تیم حریف مستقل است تا از اشغال ظرفیت یک تیم توسط تیم دیگر جلوگیری شود. یک بازیکن می‌تواند تنها اطلاعات مربوط به آن دسته از بازیکن‌ها را دریافت کند که در محدوده‌ای نزدیک‌تر از $audio\_cut\_dist$ باشند. مقادیر مربوط به این پارامترها در جدول ۲—۳ آمده است.

جدول ۲—۳: پارامترهای کارگزار برای استفاده در حسگرهای شنیداری

پارامتر	مقدار	پارامتر	مقدار
say_msg_size	۵۱۲	hear_max	۲
hear_inc	۱	hear_decay	۲
audiot_cut_dist	۵۰/۰		

حسگرهای جسمی

پیام‌های مربوط به حسگرهای جسمی حاوی تمام اطلاعات فیزیکی از بازیکن نظیر سرعت، توان، زاویه‌سر و تعداد اعمال انجام شده توسط بازیکن می‌باشند و در فواصل زمانی *sense_body_step* به بازیکن ارسال می‌شوند. قالب پیام ارسالی به شکل زیر می‌باشد:

(**sense_body** زمان
(**view_mode** کیفیت‌اطلاعات فضای‌دیداری)
(**stamina** توان تلاش)
(**speed** مقدارسرعت جهت‌سرعت)
(**neck_angle** زاویه‌گردن)
(**kick** *Kick*تعداد)
(**dash** *Dash*تعداد)
(**turn** *Turn*تعداد)
(**say** *Say*تعداد)
(**turn_neck** *TurnNeck*تعداد)
(**catch** *Catch*تعداد)
(**move** *Move*تعداد)
(**change_view** *ChangeView*تعداد))

که در آن

زمان = شماره سیکل بازی در هنگام انتقال داده
کیفیت‌اطلاعات = کیفیت اطلاعات دیداری که می‌تواند *high* یا *low* باشد
فضای‌دیداری = فضای دیداری که می‌تواند *narrow*، *normal* یا *wide* باشد
توان = عدد حقیقی مثبت و معرف میزان انرژی بازیکن
تلاش = عدد حقیقی مثبت و معرف ظرفیت تلاش بازیکن
مقدارسرعت = مقدار خطادار اندازه بردار سرعت فعلی بازیکن
جهت‌سرعت = مقدار خطادار جهت بردار سرعت فعلی بازیکن
زاویه‌گردن = جهت گردن بازیکن نسبت به بدن
شمارنده‌ها = بیانگر تعداد هریک از اعمال انجام شده

مقادیر پارامترهای مورد نیاز برای استفاده در حسگر جسمی در جدول ۲—۴ آمده‌اند.

جدول ۲—۴: پارامترهای کارگزار برای استفاده در حسگرهای جسمی

پارامتر	مقدار
sense_body_step	۱۰۰

۲—۲—۲ مدل‌های حرکتی در کارگزار

مدل حرکتی استفاده شده در سیستم کارگزار از روش گام به گام استفاده می‌کند بدین شکل که در هر سیکل از بازی سرعت بازیکن با مکان پیشین آن جمع می‌شود و مکان جدید را ایجاد می‌کند. سرعت نیز دچار یک **اُفت** مشخص می‌شود. برای محاسبه این مجموعه از تغییرات از روابط زیر استفاده می‌شود:

$$
\begin{aligned}
(u_x^{t+1}, u_y^{t+1}) &= (v_x^t, v_y^t) + (a_x^t, a_y^t) + (\tilde{r}_1, \tilde{r}_2) + (w_1, w_2) : \text{شتاب} \\
(p_x^{t+1}, p_y^{t+1}) &= (p_x^t, p_y^t) + (u_x^{t+1}, u_y^{t+1}) : \text{جابجایی} \\
(v_x^{t+1}, v_y^{t+1}) &= \text{افت} \times (u_x^{t+1}, u_y^{t+1}) : \text{افت سرعت} \\
(a_x^{t+1}, a_y^{t+1}) &= (0, 0) : \text{شتاب صفر می‌شود}
\end{aligned}
$$

در روابط بالا (p_x^t, p_y^t)، (v_x^t, v_y^t) و (a_x^t, a_y^t) به ترتیب مکان، سرعت و شتاب بازیکن در سیکل t را نمایش می‌دهند و $(\tilde{r}_1, \tilde{r}_2)$ و (w_1, w_2) به ترتیب بردار مقادیر تصادفی و باد هستند که به حرکت شیئ افزوده می‌شوند. اُفت یک پارامتر داخلی سرور می‌باشد که برای بازیکن و توپ به ترتیب مقادیر $player\_decay$ و $ball\_decay$ را می‌گیرد.شتاب اشیا نتیجه مستقیم اعمالی است که بازیکن انجام می‌دهد. شتاب خود بازیکن با انجام عمل دویدن (dash) و شتاب توپ با ضربه زدن به آن (kick) ایجاد می‌شوند. در شرایطی که دو شیئ با یکدیگر برخورد می‌کنند در جهت زاویه‌ای که هریک از آن آمده‌اند تا زمانی که با هم برخورد نداشته باشند به عقب بازگردانده می‌شوند و پس از آن سرعت هریک از آنها به ۰/۱ مقدار اولیه تقلیل می‌یابد.

برای شبیه‌سازی عدم قطعیت موجود در دنیای واقعی و برای ایجاد خطا در حرکت اشیا سیستم کارگزار بطور یکسانی اعداد تصادفی را در حرکت تمامی اشیا وارد می‌کند و این خطا در قالب یک بردار از مقادیر تصادفی به شکل $\tilde{r}_i$ به مدل حرکتی اشیا افزوده می‌شوند که محدوده هر مؤلفه از آن در بازه $[-r_{max}, r_{max}]$ قرار دارد. مقدار r_{max} وابسته به سرعت شیئ بوده و از رابطه زیر محاسبه می‌شود:

$$r_{max} = Rand \cdot ||(v_x^t, v_y^t) + (a_x^t, a_y^t)||$$

که در رابطه بالا $Rand$ از پارامترهای داخلی سرور می‌باشد و برای بازیکن مقدار $player\_rand$ و برای توپ مقدار $ball\_rand$ را می‌گیرد.

علاوه بر روش ایجاد خطا در حرکت اشیا، کارگزار توانایی افزودن باد به سیستم حرکتی را نیز داراست. سرعت باد با توجه به روابط زیر استنباط می‌شود و در نهایت بردار (w_1, w_2) به مدل حرکتی شیئ اضافه می‌شود:

$$(w_x, w_y) = \pi(\boldsymbol{wind\_force}, \boldsymbol{wind\_dir})$$

$$(w_1, w_2) = ||(v_x^t, v_y^t) + (a_x^t, a_y^t) + (\tilde{r}_1, \tilde{r}_2)|| \cdot \frac{(w_x + \tilde{e}_1, w_y + \tilde{e}_2)}{Weight \times 10000}$$

در این رابطه $wind\_force$ و $wind\_dir$ از پارامترهای داخلی کارگزار می‌باشند و π روالی برای تبدیل مدل قطبی به مدل معادل برداری است. $\tilde{e}_i$ عددی تصادفی از بازه $[-wind\_rand, wind\_rand]$ می‌باشد. **Weight** با توجه به وزن شیئ مورد نظر انتخاب می‌شود که در مورد بازیکن مقدار $player\_weight$ و در مورد توپ مقدار $ball\_weight$ را می‌گیرد. در حال حاضر پارامتر باد مورد استفاده قرار نمی‌گیرد و این مقدار برابر با $(0, 0)$ در نظر گرفته می‌شود. جدول ۲—۵ مقادیر مربوط به پارامترهای فوق را نمایش می‌دهد.

جدول ۲—۵: پارامترهای کارگزار برای استفاده در مدل حرکتی

پارامتر	مقدار	پارامتر	مقدار
ball_decay	۰/۹۴	player_decay	۰/۴
ball_rand	۰/۰۵	player_rand	۰/۱
ball_weight	۰/۲	player_weight	۶۰/۰
wind_force	۰/۰	wind_dir	۰/۰
wind_rand	۰/۰		

۲–۲–۳ مدل‌های عملی در کارگزار

مدل‌های عملی شامل کلیه اعمالی است که بازیکن می‌تواند در زمین بازی انجام دهد. بطور کلی ۱۰ عمل برای هر بازیکن که به سیستم کارگزار متصل می‌شود قابل انجام می‌باشند. این دستورات عبارتند از :

۱) catch که مختص دروازبان بوده و امکان دریافت توپ را بوجود می‌آورد

۲) dash که امکان حرکت با سرعت دلخواه را به بازیکن می‌دهد

۳) kick که به بازیکن امکان ضربه به توپ را می‌دهد

۴) move که برای حرکت به موقعیت خاص و در شرایط ویژه‌ای از بازی استفاده می‌شود

۵) say که امکان برقراری ارتباط با سایر عوامل را فراهم می‌آورد

۶) turn که امکان چرخش را ایجاد می‌کند

۷) turn_neck که بواسطه آن یک بازیکن توانایی حرکت گردن را پیدا می‌کند

۸) change_view که برای تعیین قالب دیداری مطلوب کاربرد دارد

۹) sense_body که در آن بازیکن اطلاعات فیزیکی لازم را از کارگزار درخواست می‌کند.[۶]

۱۰) score که برای اطلاع از تعداد گل‌های به ثمر رسیده توسط تیم خودی و تیم حریف می‌تواند مورد استفاده قرار بگیرد.

از مجموعهٔ دستورات بالا، دستورات turn، move، kick، dash، و catch **دستوراتی اصلی**[۷] هستند و در هر سیکل از بازی فقط باید یکی از آنها به کارگزار ارسال شوند، اما سایر دستورات، **دستورات همزمان**[۸] می‌باشند و می‌توانند به همراه یک دستور اصلی به کارگزار فرستاده شوند. چنانچه در یک سیکل از بازی بیش از یک دستورِ اصلی به کارگزار فرستاده شود یکی از آنها به شکل تصادفی توسط کارگزار انتخاب و اجرا می‌شود که معمولاً دستوری است که زودتر ارسال شده است. بازیکن برای ارتباط با سیستم کارگزار باید این دستورات را در قالب تعریف شده‌ای بکار برد تا امکان درک آن از سوی کارگزار باشد. در ادامه مفاهیم و قالب هر یک از این دستورات را بررسی می‌کنیم.

عمل Catch

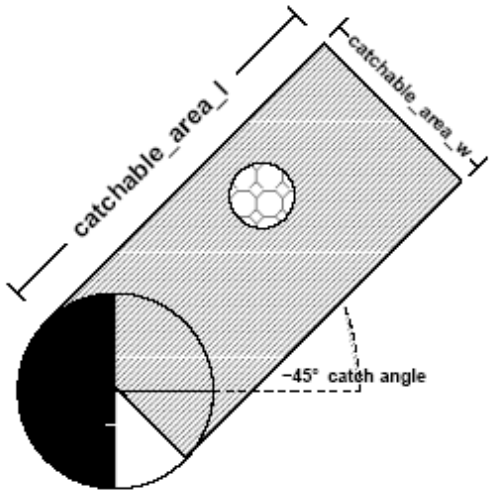
دروازبان تنها بازیکنی است که می‌تواند عمل catch را انجام دهد. این عمل تنها در شرایطی قابل اجراست که بازی در مُد "play_on" در حال انجام باشد. بازیکن می‌تواند توپ را هنگامی که در محوطه catchable قرار دارد در اختیار بگیرد. این فضا در زاویه ϕ از جهت بدن بازیکن و به شکل مستطیلی با ابعاد $catchable\_area\_l$ و $catchable\_area\_w$ در جهت این زاویه ایجاد می‌شود(شکل ۲–۳). حداقل و حداکثر

---

[۶] این دستور با توجه به ویژگی فعلی کارگزار که اطلاعات جسمی را به طور خودکار و در شروع هر سیکل از بازی ارسال می‌کند کاربرد چندانی ندارد اما در نسخه‌های پیشین که سیستم کارگزار فاقد چنین امکانی بود مورد استفاده قرار می‌گرفت. اطلاعات دریافتی حاصل از این دستور در بخش **حسگرهای جسمی**(۲–۲–۱) قابل مطالعه است.

[۷] primary command

[۸] concurrent command



شکل ۲—۳: فضای catchable توسط دروازبان ($\phi = -۴۵°$)

جدول ۲—۶: پارامترهای کارگزار برای استفاده در عمل catch

پارامتر	مقدار	پارامتر	مقدار
catchable_area_l	۲٫۰	goalie_max_moves	۲
catchable_area_w	۱٫۰	minmoment	−۱۸۰
catch_probability	۱٫۰	maxmoment	۱۸۰
catch_ban_cycles	۵	drop_ball_time	۲۰۰

زاویه ϕ به ترتیب توسط پارامترهای *minmoment* و *maxmoment* محدود می‌شوند. احتمال تصاحب توپ در این فضا *catch_probability* می‌باشد و پس از انجام هر عمل catch بازیکن تعداد *catch_ban_cycles* از انجام عمل catch ناتوان است.

قالب دستور catch به شکل زیر می‌باشد:

(**catch** dir)

که dir همان زاویه ϕ و به بیان بهتر زاویه‌ایست که عمل catch در آن جهت انجام می‌گیرد.

با دریافت توپ توسط دروازبان سمت چپ یا سمت راست مُد بازی به goalie_catch_ball_[l|r] و سپس بلافاصله در همان سیکل از بازی به free_kick_[l|r] تغییر می‌کند. پس از آن دروازبان قادر است به تعداد *goalie_max_moves* در محوطه جریمه خود عمل move انجام دهد و پس از آن ملزم است که توپ را پرتاب کند. چنانچه توپ به اندازه *drop_ball_time* در اختیار دروازبان باشد و پرتاب نشود، عمل پرتاب از سوی داور خودکار انجام می‌شود. مقادیر مربوط به پارامترهای بالا در جدول ۲—۶ آمده اند.

جدول ۲—۷: پارامترهای کارگزار برای استفاده در عمل dash

پارامتر	مقدار	پارامتر	مقدار
min_power	−۱۰۰	max_power	۱۰۰
stamin_max	۴۰۰۰	extra_stamina	۰/۰
stamin_inc_max	۴۵	dash_power_rate	۰/۰۰۶
effort_max	۱/۰	effort_min	۰/۶
effort_dec	۰/۰۰۵	effort_dec_thr	۰/۳
effort_inc	۰/۰۰۱	effort_inc_thr	۰/۶
recover_dec	۰/۰۰۲	recover_dec_thr	۰/۳
recover_min	۰/۵	player_speed_max	۱/۰
player_accel_max	۱/۰		

عمل Dash **و مدل** Stamina

dash دستوری است که امکان سرعت گرفتن بازیکن در جهت بدنش را به او می‌دهد. این دستور با یک پارامتر که میزان شتاب را مشخص می‌کند کار می‌کند و قالب زیر را داراست :

(**dash** power)

که در آن power انرژی حرکت و یک عدد حقیقی و در بازه $minpower$ و $maxpower$ می‌باشد. این انرژی رابطه مستقیم با **توان**[۹] بازیکن دارد. این توان در شروع بازی به مقدار $stamina\_max$ مقداردهی می‌شود و با هر بار عمل dash از آن کاسته می‌شود. چنانچه میزان انرژی مثبت باشد عمل dash در جهت بدن بازیکن و میزان کاهش توان برابر با میزان انرژی مصرفی است اما اگر انرژی ارسال شده با دستور dash منفی باشد حرکت در خلاف جهت بدن بوده و کاهش توان به اندازه دو برابر انرژی مصرفی است. چنانچه میزان توان کمتر از انرژی ارسال شده باشد مقدار انرژی به اندازه توان تغییر می‌کند و اجرای دستور نیاز به توان بیشتری ندارد. با کاسته شدن از میزان توان محاسبه انرژی دویدن مؤثر[۱۰] (edp) از سوی کارگزار آغاز می‌شود. پارامتر $dash\_power\_rate$ و نیز میزان فعلی تلاش[۱۱] به منظور محاسبه edp استفاده می‌شوند که تلاش می‌تواند در بازه $effort\_min$ و $effort\_max$ باشد. edp از رابطه زیر محاسبه می‌شود:

$$edp = effort \cdot \boldsymbol{dash\_power\_rate} \cdot Power$$

در رابطه فوق $Power$ همان مقدار انرژی ارسال شده به کارگزار از سوی بازیکن و $dash\_power\_rate$ از پارامترهای داخلی کارگزار است. با محاسبه edp، این مقدار به همراه جهت بدن بازیکن برای محاسبه شتاب حرکتی مورد استفاده قرار می‌گیرند. پس از محاسبه (a_x^t, a_y^t) و با داشتن سرعت و مکان قبلی بازیکن می‌توان مدل حرکتی را بدست آورد (رجوع شود به بخش ۲—۲—۲).

برای محاسبه شتاب باید روند تأثیر توان روی آن بررسی شود. سه متغیر اصلی در تعیین توان بازیکن مؤثر می‌باشند:

۱) Stamina : توان ذخیره شده در بازیکن را نشان می‌دهد، با انجام هر عمل dash از مقدار آن به اندازه انرژی مصرف شده در دستور dash کاسته می‌شود و با شروع هر سیکل از بازی به آرامی بازیابی می‌شود. مقدار Stamina بین ۰ و $stamina\_max$ می‌باشد.

---

[۹] Stamina

[۱۰] effective dash power

[۱۱] effort

۲) Recovery : معرف میزان بازگشت Stamina در شروع هر سیکل از بازی می‌باشد. به بیان بهتر برگشت توان را در بازیکن دستکاری می‌کند. مقدار Recovery همواره در بازه $[effort\_min, effort\_max]$ قرار دارد.

۳) Effort : بیانگر میزان تأثیر یک عمل dash در جابجایی بازیکن است و همواره بین $recover\_min$ و ۱ قرار دارد.

مدل Stamina مورد استفاده توسط کارگزار در زیر آمده است.

{if stamina is below recovery decrement threshold. recovery is reduced}
if $stamina \leq recover\_dec\_thr \cdot stamina\_max$ **then**
  if $recovery > recover\_min$ **then**
    $recovery \leftarrow recovery - recover\_dec$
  end if
end if

{if stamina is below effort decrement threshold, effort is reduced}
if $stamina \leq effort\_dec\_thr \cdot stamina\_max$ **then**
  if $effort > effort\_min$ **then**
    $effort \leftarrow effort - effort\_dec$
  end if
  $effort \leftarrow \max(effort, effort\_min)$
end if

{if stamina is above effort increment threshold, effort is increased}
if $stamina \geq effort\_inc\_thr \cdot stamina\_max$ **then**
  if $effort < effort\_max$ **then**
    $effort \leftarrow effort + effort\_inc$
    $effort \leftarrow \min(effort, effort\_max)$
  end if
end if

{recover the stamina a bit}
$stamina \leftarrow stamina + reovery \cdot stamina\_inc\_max$
$stamina \leftarrow \min(stamina, stamina\_max)$

با توجه به اینکه شتاب لازم برای حرکت بازیکن تابع مستقیمی از انرژی مصرفی توسط اوست و با در نظر گرفتن این مسئله که انرژی فرستاده شده از سوی بازیکن در یک عمل dash لزوماً همان مقداری نیست که در نهایت توسط کارگزار بر سیستم تأثیر می‌گذارد، رابطه زیر را محاسبه انرژی شتاب نهایی مورد استفاده قرار می‌دهیم:

$$(a_x^t, a_y^t) = edp \times (\cos(\theta^t), \sin(\theta^t))$$

در رابطه بالا θ جهت بدن بازیکن را نشان می‌دهد. توجه به این نکته ضروری است که عمل dash تنها در یک سیکل باعث ایجاد شتاب می‌شود و در شروع سیکل بعدی مجدداً شتاب بازیکن صفر می‌شود. مقدار بردار شتاب $\vec{a_t}$ و مقدار بردار سرعت $\vec{v_t}$ پس از محاسبه به ترتیب با حد بالای $player\_accel\_max$ و $player\_speed\_max$ نرمال می‌شوند و مدل حرکتی بازیکن را ایجاد می‌کنند. جدول ۲—۷ لیست پارامترهای مورد استفاده از سوی کارگزار برای محاسبه یک فرایند dash را نشان می‌دهد.

عمل kick

kick دستوری است که به بازیکن امکان ضربه به توپ در جهت خاصی را می‌دهد. این دستور باید به شکل زیر به کارگزار ارسال شود:

(**kick** power dir)

که در آن power انرژی صرف شده برای ضربه به توپ و در بازه $[minpower, maxpower]$ و dir جهتی است که توپ بدان شلیک می‌شود. dir می‌تواند در بازه $[minmoment, maxmoment]$ انتخاب شود. برای ضربه به توپ، توپ باید حتماً در فاصله‌ای کمتر از $kickable\_margin + player\_size + ball\_size$ نسبت به بازیکن قرار بگیرد که $ball\_size$، $player\_size$ و $kickable\_margin$ از پارامترهای داخلی کارگزار می‌باشند.

جدول ۲—۸: پارامترهای کارگزار برای استفاده در عمل kick

پارامتر	مقدار	پارامتر	مقدار
minpower	−۱۰۰	kick_rand	۰/۰
maxpower	۱۰۰	ball_accel_max	۲/۷
minmoment	−۱۸۰	ball_speed_max	۲/۷
maxmoment	۱۸۰	ball_size	۰/۰۸۵
kickable_margin	۰/۷	player_size	۰/۳
kick_power_rate	۰/۰۲۷		

انرژی صرف شده برای ضربه زدن به توپ نیز همانند انرژی لازم برای دویدن لزوماً به همان اندازه که به کارگزار فرستاده می‌شود برای جابجایی توپ مؤثر نیست. محاسبه انرژی مؤثر برای شیلک توپ[۱۲] (ekp) با توجه به رابطه زیر صورت می‌گیرد:

$$ekp = Power \cdot \boldsymbol{kick\_power\_rate}$$

$$ekp = ekp \cdot \left(۱ - ۰/۲۵ \times \frac{dir\_diff}{۱۸۰^{\circ}} - ۰/۲۵ \times \frac{dist\_diff}{\boldsymbol{kickable\_margin}} \right)$$

در رابطه فوق $Power$ همان انرژی صرف شده برای ضربه به توپ و $kick\_power\_rate$ از پارامترهای داخلی کارگزار است. در نهایت شتاب ایجاد شده برای توپ با توجه به ضربه وارد شده به آن از رابطه زیر محاسبه می‌شود:

$$(a_x^t, a_y^t) = ekp \times (\cos(\theta^t), \sin(\theta^t)) + (\tilde{k}_۱, \tilde{k}_۲)$$

[۱۲] effective kick power

در رابطه بالا θ زاویه‌ای است که توپ بدان سمت خواهد رفت و برابر با مجموع زاویه بدن بازیکن و زاویه dir انتخابی توسط بازیکن در هنگام عمل kick است. بردار $(\tilde{k}_1, \tilde{k}_2)$ برای افزودن خطا به مدل حرکتی توپ مورد استفاده قرار می‌گیرد و به شکل تصادفی در بازه $[-k_{max}, k_{max}]$ انتخاب می‌شود. محاسبه این خطا با استفاده از $kick\_rand$ که از پارامترهای داخلی کارگزار است، به طریق زیر انجام می‌شود :

$$k_{max} = \boldsymbol{kick\_rand} \cdot \frac{Power}{\boldsymbol{max\_power}}$$

که $kick\_rand$ و max_power از پارامترهای داخلی کارگزار می‌باشند. برای بازیکنان معمولی مقدار $kick\_rand$ برابر با صفر در نظر گرفته می‌شود. با محاسبه $\vec{a}_t$ و $\vec{v}_t$ این دو مقدار با حدود بالای $ball\_accel\_max$ و $ball\_speed\_max$ نرمال می‌شوند.

با استفاده از روش‌های عنوان شده در بخش **مدل حرکتی** (۲—۲—۲) می‌توان الگوی حرکت توپ را استنباط کرد. لیست پارامترهایی که توسط کارگزار برای محاسبه عمل kick مورد استفاده قرار می‌گیرند و مقادیر آنها در جدول ۲—۸ آمده‌اند.

عمل move

عمل move برای قرار دادن بازیکن در یک مکان خاص از زمین مورد استفاده قرار می‌گیرد و تنها می‌تواند در مُدهای مشخصی از بازی استفاده شود. این عمل در هنگامیکه بازی در مُد $before\_kick\_off$ یا $goal\_l$ یا $goal\_r$ قرار دارد به بازیکنان هر دو تیم این امکان را می‌دهد که برای شکل دهی ساختار تیمی ابتدایی، به مکان مشخصی از زمین بازی move کنند. این دستور بدون اعمال خطا اجرا می‌شود و بازیکن را دقیقاً در محل مطلوب قرار می‌دهد. دروازبان علاوه بر امکان انجام move در شرایط بالا همانطور که در **مدل** catch اشاره شد، می‌تواند این عمل را در هنگام گرفتن توپ نیز انجام دهد. قالب این دستور به شکل زیر است:

(**move** x y)

در شرایطی که مُد بازی یکی از سه مقدار $before\_kick\_off$ یا $goal\_l$ یا $goal\_r$ می‌باشد بازیکن می‌تواند x و y را به ترتیب در بازه $[-pitch\_length/2, 0.0]$ و $[-pitch\_width/2, pitch\_width/2]$ انتخاب کند، اما در شرایطی که دروازبان توپ را می‌گیرد x و y باید در محوطه جریمه تیم خودی انتخاب شوند. تعداد اعمال move مجاز برای بازیکن پس از تصاحب توپ تعداد $goalie\_max\_moves$ می‌باشد. اگر دروازبان بیشتر از این تعداد عمل move را به کارگزار ارسال کند پیغامی به شکل (error too_many_moves) دریافت می‌کند و اعمال اضافی انجام نخواهند شد. پارامترهای مورد نیاز برای انجام یک عمل move در جدول ۲—۹ آمده‌اند.

جدول ۲—۹: پارامترهای کارگزار برای استفاده در عمل move

پارامتر	مقدار	پارامتر	مقدار
pitch_length	۵۲٫۵	goalie_max_moves	۲
pitch_width	۳۴٫۰		

جدول ۲—۱۰: پارامترهای کارگزار برای استفاده در عمل say

پارامتر	مقدار	پارامتر	مقدار
say_message_size	۱۰	audio_cut_dist	۵۰
hear_max	۲	hear_inc	۱
hear_decay	۲		

عمل say

say روشی است که بازیکن را قادر می‌سازد تا یک پیام را در زمین بازی پخش کند. این عمل به بازیکن امکان برقراری ارتباط با سایر بازیکنان را می‌دهد. طول داده قابل انتقال در یک پیغام say برابر با $say\_message\_size$ می‌باشد و بازیکن محدود به استفاده از مجموعه کارکترهای [0-9a-zA-Z()<>.?+-*/] (بدون علامت‌های براکت) است. پیام ارسالی از سوی یک بازیکن بدون هیچ وقفه زمانی به سایر بازیکنان ارسال می‌شود و توسط حسگرهای شنوایی سایر بازیکنان (۲—۲—۱)، در فاصله $audio\_cut\_dist$ قابل درک است. محدودیت ارسال یک پیام از طریق say با محدود شدن سایر بازیکنان در شنیدن پیام‌های say حاصل می‌شود.این پیام باید در قالب زیر به کارگزار ارسال شود:

(**say** message)

که message پیام ۱۰ بایتی پخش شده در محیط است. جدول ۲—۱۰ لیست پارامترهای مورد استفاده توسط کارگزار در ایجاد یک پیغام say را نشان می‌دهد.

عمل turn

چرخش در زمین بازی عملی است که با استفاده از دستور turn برای بازیکن ممکن می‌شود. این عمل در قالب زیر به کارگزار ارسال می‌شود:

(**turn** Moment)

با استفاده از دستوری مشابه بالا بازیکن می‌تواند به زاویه دلخواه Moment در زمین بازی بچرخد و Moment می‌تواند در بازه $[minmoment, maxmoment]$ انتخاب شود. این عمل نیز مشابه اعمال kick و dash تحت تأثیر خطا می‌باشد و زاویه ارسال شده به کارگزار لزوماً همان زاویه‌ای نیست که بازیکن تحت آن می‌چرخد. زاویه مؤثر برای چرخش[۱۳] (etm) از اینرسی حرکتی بازیکن تأثیر می‌گیرد و با توجه به فرمول زیر محاسبه می‌شود:

$$etm = \frac{(۱ + \tilde{r}) \cdot Moment}{۱ + \boldsymbol{inertia\_moment} \cdot player\_speed}$$

در رابطه بالا $inertia\_moment$ برابر با اینرسی بازیکن بوده و جزو پارامترهای کارگزار است. $\tilde{r}$ عدد تصادفی و در بازه $[-player\_rand, player\_rand]$ می‌باشد و Moment میزان چرخشی است که توسط بازیکن به کارگزار ارسال شده است. $player\_speed$ سرعت بازیکن در سیکل جاری است و چنانچه بازیکن با حداکثر سرعت در حال حرکت باشد این مقدار برابر با $player\_speed\_max$ می‌باشد و نتیجتاً بیشترین و کمترین مقدار ممکن برای چرخش ± ۳۰ است اما از آنجا که یک عمل dash نمی‌تواند با یک عمل turn همزمان باشد بیشترین سرعت یک بازیکن پیش از انجام عمل turn می‌تواند $player\_speed\_max \cdot player\_decay$

[۱۳] effective turn moment

باشد و با در نظر گرفتن این مقادیر حداکثر زاویه‌ای که بازیکن می‌تواند تحت آن بچرخد ۶۰± می‌باشد. جدول ۲—۱۱ لیست پارامترهایی است که کارگزار برای مدل کردن عمل turn مورد استفاده قرار می‌دهد.

جدول ۲—۱۱: پارامترهای کارگزار برای استفاده در عمل turn

پارامتر	مقدار	پارامتر	مقدار
minmoment	−۱۸۰	maxmoment	۱۸۰
inertia_moment	۵/۰	player_rand	۰/۱

عمل turn_neck

با استفاده از دستور turn_neck بازیکن می‌تواند گردن خود را به شکل مستقل از سایر اعمال بچرخاند. این دستور از قالب زیر پیروی می‌کند:

(**turn_neck** NeckMoment)

در رابطه بالا NeckMoment زاویه‌ایست که گردن می‌تواند تحت آن بچرخد. زاویه گردن می‌تواند در بازه $[minmoment, maxmoment]$ قرار بگیرد اما مقدار زاویه گردن بازیکن نسبت به بدن او در بازه $[minneckmoment, maxneckmoment]$ قرار می‌گیرد و از جانب کارگزار اعلام می‌شود. از آنجا که turn_neck از دسته اعمال همزمان است، می‌تواند به شکل همزمان با turn یا سایر اعمال اصلی انجام شود، اما چرخش گردن مستقل از اینرسی حرکتی بازیکن می‌باشد. پارامترهایی که کارگزار برای شبیه‌سازی چرخش گردن مورد استفاده قرار می‌دهد در جدول ۲—۱۲ آمده‌اند.

جدول ۲—۱۲: پارامترهای کارگزار برای استفاده در عمل turn_neck

پارامتر	مقدار	پارامتر	مقدار
minneckmoment	−۱۸۰	maxneckmoment	۱۸۰
minneckang	−۹۰	maxneckang	۹۰

عمل change_view

این دستور به بازیکن امکان می‌دهد تا وضعیت دیداری خود را بر حسب نیاز و در شرایط مختلف تغییر دهد. این دستور با الگوی زیر به کارگزار ارسال می‌شود:

(**change_view** viewWidth viewQuality)

در دستور ارسالی فوق viewWidth می‌تواند یکی از سه مقدار low، normal و یا high را بگیرد و viewQuality نیز می‌تواند یکی از مقادیر low، یا high را داشته باشد. با توجه به این مقادیر ارسالی و با تطبیق این مقادیر با ضرایب متناسب، کارگزار بازه زمانی مورد نظر برای ارسال پیام‌های دیداری را تعیین می‌کند (بخش ۲—۲—۱). این دستور از مجموعه دستورات همزمان بوده و می‌تواند به تعداد متعددی در یک سیکل از بازی به کارگزار ارسال شود.

جدول ۲—۱۳: تقسیم بندی مُدهای بازی از سوی کارگزار

مُدبازی	t_c	مُد متعاقب	توضیح
before_kick_off	۰	kick_off_*Side*	قبل از آغاز هر نیمه از بازی
play_on			در طول روال عادی بازی
time_over			در شرایطی که بازی به پایان رسیده باشد
kick_off_*Side			شروع بازی را اعلام می‌کند (پس از فشردن دکمه $Kick\_off$ یا شروع خودکار)
kick_in_*Side		play_on	هنگامی که یک تیم باید پرتاب out انجام دهد
free_kick_*Side		play_on	یک ضربه خطا به سود تیم سمت *Side*
corner_kick_*Side		play_on	یک ضربه کرنر به سود تیم سمت *Side*
goal_kick_*Side*		play_on	یک ضربه دروازه به سود تیم سمت *Side*
goal_*Side*			گل توسط تیم سمت *Side*
drop_ball	۰	play_on	پرتاب دستی توپ از سوی داور انسانی
offside_*Side*	۳۰	free_kick_*OSide*	خطای افساید توسط تیم سمت *Side*
goal_*Side*_n	۵۰	kick_off_*OSide*	به ثمر رسیدن *n*امین گل تیم سمت *Side*
foul_*Side*	۰	free_kick_*OSide*	خطای تیم سمت *Side*
goalie_catch_ball_*Side*	۰	free_kick_*OSide*	دروازبان تیم سمت *Side* توپ را catch کند
time_up_without_a_team	۰	time_over	تا پایان نیمه دوم حریفی در زمین نباشد
time_up	۰	time_over	اگر در پایان ۶۰۰۰ سیکل دو تیم با هم مساوی نباشند ارسال می‌شود
half_time	۰	before_kick_off	با رسیدن به نیمه ارسال می‌شود
time_extended	۰	before_kick_off	اگر در پایان ۶۰۰۰ سیکل دو تیم مساوی باشند ارسال می‌شود

در جدول فوق *Side* می‌تواند مقدار l یا r را، که به ترتیب معرف تیم سمت چپ و سمت راست می‌باشند، بپذیرد. *OSide* معرف جهت مقابل است.

عمل score

این عمل نیز از اعمال همزمان قابل ارسال به کارگزار است که به بازیکن امکان فهمیدن نتیجه بازی را می‌دهد. قالب دستور به شکل ساده (**score**) می‌باشد. از آنجا که نتیجه بازی توسط بازیکن قابل ذخیره سازی است این دستور چندان در استفاده متداول نیست.

۲—۳ مُدهای بازی

همانطور که در ابتدای فصل اشاره شد سیستم کارگزار برای تشخیص خطاهای منطقی و محاسباتی از یک داور خودکار استفاده می‌کند. این داور بسته به شرایط بازی عملکرد کارگزار را از طریق انتقال وضعیت به یک مجموعه از حالت‌های از پیش تعریف شده کنترل می‌کند. این مجموعه حالت‌ها مُدهای بازی هستند که بازیکن در هریک از آنها مجبور است سیاست خاصی را در عملکرد خود اتخاذ کند. لیست این مُدها و توضیح در مورد هر یک از آنها در جدول ۲—۱۳ آمده است.

چنانچه یکی از مُدهایی که در جدول ۲—۱۳ با علامت * مشخص شده‌اند رخ دهد، داور خودکار

همه بازیکنان تیم حریف را وادار می‌کند تا خارج از دایره‌ای به شعاع $offside\_kick\_margin$ و مرکزیت توپ قرار بگیرند، که در حال حاضر مقدار این شعاع ۹/۱۵m می‌باشد.

۲—۴ بازیکنان نامتشابه

یکی از ویژگی‌هایی که از نسخه ۷ کارگزار به سیستم اضافه شده است امکان تعویض بازیکنان داخل زمین است. بطور پیش‌فرض ۱۱ بازیکن در زمین بازی وجود دارند که همه آنها توسط کارگزار از نوع اولیه انتخاب شده‌اند. در شروع یک مسابقه و هنگامی که بازی در مُد before_kick_off قرار دارد مربی online می‌تواند بازیکنان دلخواه خود را در زمین بازی وارد کند. انواع بازیکنانی که مربی می‌تواند انتخاب کند از یک مجموعه به تعداد $player\_types$ انتخاب می‌شوند که نوع صفر در این مجموعه پیش‌فرض می‌باشد. بازیکنان با انواع مختلف دارای توانایی‌های مختلف هستند و از نظر سرعت و انرژی و سایر پارامترها با هم متفاوت هستند. در جریان بازی هم مربی قادر است به تعداد $subs\_max$ در زمین بازی بازیکنان جدیدی وارد کند. تعداد بازیکنانی از یک نوع خاص که می‌توانند همزمان در زمین بازی حضور داشته باشند نباید بیشتر از $subs\_max$ باشد و با هر تعویض در زمان بازی کلیه ویژگی‌های بازیکن جدید به مقادیر اولیه مقداردهی می‌شوند و خصوصیاتی نظیر انرژی و تلاش به بیشترین مقادیر خود بر می‌گردند. خصوصیاتی مانند سرعت و انرژی که برای بازیکنان با انواع مختلف متفاوت است با توجه به نوع بازیکن جدید مقادیر مناسب را می‌گیرند.

با شروع بازی، لیست تمامی بازیکنان با انواع مختلف به همه بازیکنان و مربی‌های online منتقل می‌شود. هر دو تیم موظف هستند که از یک مجموعه یکسان و مشخص بازیکنان خود را انتخاب کنند. با بدست آوردن اطلاعات مربوط به بازیکنان مختلف مربی قادر است بازیکنان مطلوب خود را با توجه به نیازی که در مناطق مختلف بازی حس می‌کند انتخاب و جایگزین کند. قالب دستوری که از سوی کارگزار به عوامل درون زمین ارسال می‌شود به شکل زیر می‌باشد:

(**player_type** id player_speed_ max stamina_inc_max player_decay inertia_moment dash_power_rate player_size kickable_margin kick_rand extra_stamina effort_max effort_min)

پارامترهای که در پیام بالا آمده‌اند ویژگی‌هایی هستند که در بین بازیکنان نامتشابه متفاوت هستند و می‌توانند خصوصیات خاصی را برای بازیکنان ایجاد نمایند. در مجموعه ۱۱ نفری یک تیم همه بازیکنان به استثناء دروازبان می‌توانند با سایر انواع تعویض شوند، اما دروازبان در طول بازی باید لزوماً از نوع صفر باشد. در پیام فوق id بیانگر نوع بازیکن و در بازه [۷—۰] است. جدول ۲—۱۵ مقایسه‌ای بین پارامترهای موجود در کارگزار برای بازیکنان نوع صفر و معادل آنها برای بازیکنان نامتشابه است. جدول ۲—۱۴ لیست پارامترهای مورد استفاده توسط کارگزار برای مدیریت ورود و خروج بازیکنان نامتشابه را نشان می‌دهد.

جدول ۲—۱۴: پارامترهای کارگزار برای استفاده در تعویض بازیکنان

پارامتر	مقدار	پارامتر	مقدار
player_types	۷	subs_max	۳

جدول ۲—۱۵: مقایسه پارامترهای کارگزار بین بازیکنان پیش‌فرض و نامتشابه

پارامترهای کارگزار برای بازیکن پیش‌فرض		پارامترهای معادل برای بازیکن نامتشابه		
نام	مقدار	نام	مقدار	بازه
stamin_inc_max	۴۵	stamina_inc_max_delta_factor	−۱۰۰	۲۵—۴۵
		player_speed_max_delta_min	۰/۰	
		player_speed_max_delta_max	۰/۲	
extra_stamina	۰/۰	extra_stamina_delta_min	۰/۰	۰/۰—۱۰۰
		extra_stamina_delta_max	۱۰۰/۰	
dash_power_rate	۰/۰۰۶	dash_power_rate_delta_min	۰/۰	۰/۰۰۶—۰/۰۰۸
		dash_power_rate_delta_max	۰/۰۰۲	
effort_min	۰/۶	effort_min_delta_factor	−۰/۰۰۲	۰/۴—۰/۶
		effort_stamina_delta_min	۰/۰	
		effort_stamina_delta_max	۱۰۰/۰	
effort_max	۱/۰	effort_max_delta_factor	−۰/۰۰۲	۰/۸—۱/۰
		extra_stamina_delta_min	۰/۰	
		extra_stamina_delta_max	۱۰۰/۰	
player_speed_max	۱/۰	player_speed_max_delta_min	۰/۰	۱/۰—۱/۲
		player_speed_max_delta_max	۰/۲	
player_decay	۰/۴	player_decay_delta_min	۰/۰	۰/۴
		player_decay_delta_max	۰/۲	
kickable_margin	۰/۷	kickable_margin_delta_min	۰/۰	۰/۷—۰/۹
		kickable_margin_delta_max	۰/۲	
kick_rand	۰/۰	kick_rand_delta_factor	۰/۵	۰/۰—۰/۱
		kickable_margin_delta_min	۰/۰	
		kickable_margin_delta_max	۰/۲	
inertia_moment	۵/۰	player_decay_delta_min	۰/۰	۵/۰—۱۰/۰
		player_decay_delta_max	۰/۲	
		inertia_moment_delta_factor	۲۵/۰	
player_size	۰/۳	player_size_delta_factor	−۱۰۰/۰	۰/۱—۰/۳
		dash_power_rate_delta_min	۰/۰	
		dash_power_rate_delta_max	۰/۰۰۲	

۲—۵ مربی

در سیستم شبیه‌سازی مربی می‌تواند در دو وضعیت online coach و trainer فعالیت کند که هر دو نوع داده‌های مطلق و بدون خطا را از کارگزار دریافت می‌کنند.

در حالت اول مربی قادر است بازیکنان خاص را در تیم وارد کند و یا در جریان بازی عملیات تعویض و هدایت یک تیم را در اختیار بگیرد. برای این منظور مربی می‌تواند از طریق پورت ۶۰۰۲ و اتصال UDP/IP به کارگزار متصل شده و در جریان بازی عملکرد تیم خودی و حریف را زیر نظر بگیرد و در صورت نیاز توصیه‌ها و تعویض‌های لازم را در تیم خودی انجام دهد. این مُد از مربی عموماً در بازی‌های رسمی مورد استفاده قرار می‌گیرد و با توجه به درک کامل و بدون خطایی که از محیط دارد قادر است به تحلیل و بررسی عملکرد تیم حریف نیز بپردازد اما توانایی آن در برقراری ارتباط با بازیکنان با توجه به تعداد پیام‌هایی که می‌تواند رد و بدل کند محدود شده است. به منظور توانا نمودن مربی در کار با تیم‌های مختلف، زبان

استانداردی تعریف شده است که مشتمل بر پنج قسمت Info، Advice، Define، Meta، و Freeform می‌باشد.

در حالت trainer مربی قادر خواهد بود با ایجاد شرایط خاصی در زمین بازی امکان یادگیری یا تست یک فعالیت خاص را برای تیم ایجاد کند. در این وضعیت مربی می‌تواند بازیکنان دو تیم و یا توپ را به نقطه خاصی از زمین بازی منتقل کند، بازی را در مُد خاصی قرار دهد، مطلبی را در زمین منتشر کند و یا سایر اعمالی که می‌تواند برای آموزش تیم مفید باشد را در زمین انجام دهد. حالت trainer در بازی‌های رسمی غیر فعال است و تنها در شرایط آزمایشگاهی مورد استفاده قرار می‌گیرد. برای این منظور باید کارگزار به همراه پارامتر "coach-" اجرا شود و در این حالت برنامه trainer می‌تواند از طریق پورت ۶۰۰۱ و UDP/IP به کارگزار متصل شده و پیام‌های لازم را ارسال کند. در این حال مربی می‌تواند مجموعه دستورات زیر را به کارگزار ارسال نماید:

- (change_mode PlayMode) : که مد بازی را به PlayMode تغییر می‌دهد. PlayMode می‌تواند یکی از انواع اشاره شده در بخش ۲—۳ باشد. اگر PlayMode برابر با $before\_kick\_off$ باشد بازیکنان به طور خودکار به زمین تیم خودی فرستاده می‌شوند.
- (move OBJECT X Y [VDIR [VELX VELY]]) : با استفاده از این دستور شیئ OBJECT به مکان (X, Y) از زمین بازی منتقل می‌شود. در صورتیکه VDIR تعریف شود جهت مطلق بدن و سر (فقط برای بازیکن) در این راستا قرار می‌گیرد و اگر (VELX, VELY) نیز تعریف شده باشند به شیئ مورد نظر سرعتی به اندازه بردار $\vec{v}$=(VELX, VELY) داده می‌شود.
- (recover) : به کمک این دستور انرژی صرف شده توسط تمام بازیکنان مجدداً به مقدار اولیه باز گردانده می‌شود.
- (ear MODE) : این دستور انتقال داده‌های شنیداری به trainer را کنترل می‌کند. MODE می‌تواند on یا off باشد. در حالت on اطلاعات شنیداری به trainer منتقل می‌شود و در حالت off این داده‌ها به trainer منتقل نمی‌شوند.
- (start) : این دستور بازی را شروع می‌کند، در واقع معادل فشردن دکمه kick_off می‌باشد.
- (team_names) : با استفاده از این دستور نام تیم سمت چپ و راست به trainer گزارش می‌شود.
- (checkball) موقعیت توپ در زمین بازی را درخواست می‌کند. توپ می‌تواند در یکی از مکان‌های in_field، $goal\_l$، $goal\_r$ و یا out_of_field باشد. جواب از کارگزار به شکل (ok check_ball TIME BALLPOS) دریافت می‌شود که در آن TIME شماره سیکل بازی را مشخص می‌کند و BALLPOS یکی از چهار مکانی است که توپ می‌تواند داشته باشد.

هریک از مجموعه دستورات بالا در صورتی که به درستی اجرا شوند یک پیام تأیید از سوی کارگزار ارسال می‌شود و در صورتی که نادرست انجام شوند پیام خطا در پاسخ فرستاده می‌شود[۱۱].

۲—۶ بازنگری فصل

در این بخش اصول عملکرد کارگزار در طراحی و مدل سازی بازی و نیز روش‌هایی که برای ارتباط با عامل‌ها مورد استفاده قرار می‌دهد بررسی شد. مجموعه اطلاعات دریافتی در قالب See، Hear، و

Sense_Body و ساختار این پیام‌ها بررسی شد. مجموعه اعمالی که یک بازیکن مجاز است در جریان یک بازی انجام دهد و همچنین ساختاری که باید در ارسال این دستورات به سیستم کارگزار رعایت کند مطالعه شد و روش‌های مربوط برای دستیابی به یک عملکرد بهینه توضیح داده شد. انواع مُدهایی که به عنوان حالت‌های مختلف عملکرد در یک بازی از سوی کارگزار تعریف شده‌اند و شرایط رخداد آنها مورد بررسی قرار گرفت. و در نهایت استفاده از مربی و بازیکنان نامتشابه به عنوان مجموعه ابزاری که می‌توانند امکان عملکرد بهتر برای یک تیم را فراهم آورند و همچنین مجموعه مزایایی که هر یک از آنها می‌توانند داشته باشند و خصوصیات آنها بررسی شد.

با درک اطلاعات کافی در خصوص محیطی که مجموعه عوامل باید در آن فعالیت کنند قادر خواهیم بود تا سیستم مناسبی را که بهترین سازگاری را با محیط خود دارد ایجاد کنیم چرا که درک بهتر دامنه عملکرد شرط لازم برای ایجاد یک مجموعه مؤثر از عوامل هوشمند است.

۳

روبوسینا: جنبه‌های بارز

هر تیم شبیه‌سازی شده به عنوان یک سیستم نرم‌افزاری بزرگ و پیچیده شامل مجموعه‌ای از مفاهیم مؤثر در عملکرد بهینه آن می‌باشد. این مجموعه از ویژگی‌ها باعث می‌شوند تا درک یک تیم از جایگاه خود و پاسخ تک تک عوامل به محرک‌های محیطی به گونه‌ای باشند که در نهایت دستیابی به هدف نهایی برای تیم تسهیل گردد. هر یک از عوامل سعی می‌کنند تا با استفاده از مجموعه امکاناتی که محیط در اختیار آنان قرار می‌دهد مناسب‌ترین عملکرد را در جهت نیل به هدف نهایی تیم داشته باشند. عمل مطابق شرایط محیطی شامل مواردی نظیر: همزمان شدن بازیکن با یک محیط زمان حقیقی، درک محیط اطراف با توجه به وضعیت اشیاء موجود در آن و به خاطر سپردن موقعیت اشیاء، استنتاج از روی وضعیت جاری و در نهایت عمل با استفاده از توانایی‌های تعریف شده برای یک عامل می‌باشد. بازیکنان ملزم هستند با توجه به امکاناتی که محیط برای آنان تعریف می‌کند عمل کنند و عملکرد بهتر نتیجه استفاده بهتر از منابع محیطی و زمینه‌ساز رسیدن به هدف نهایی است.

شکل‌گیری تیم روبوسینا به شکل کاملاً پایه‌ای انجام شد و در ساختار داخلی این تیم به جز موارد اندکی، تمام کدها از پایه نوشته شد. این عمل هر چند در ابتدا هزینه‌ی زمانی و مطالعاتی زیادی را تحمیل کرد اما در نهایت سبب ایجاد شناخت درست از محیط کارگزار و قابلیت بالا در کار با سیستم شد. سیاست کدنویسی از پایه باعث شد تا این تیم در جریان فعالیت خود مدل‌های جدیدی را برای مسائل مختلف مطرح شده در یک عامل شبیه‌سازی شده ارائه دهد و روش‌هایی را مورد استفاده قرار دهد که پیش از این طراحی و پیاده‌سازی نشده بودند.

در این فصل مهم‌ترین ویژگی‌های تیم روبوسینا را مورد بررسی قرار می‌دهیم. این ویژگی‌ها شامل بررسی معماری یک بازیکن و مراحل ایجاد یک بازیکن کامل است که در بخش ۳—۱ بررسی می‌شود، همزمان‌سازی بازیکن با محیط کارگزار برای برقرار ارتباط درست و بدون تأخیر در بخش ۳—۲ مطالعه می‌شود، بخش ۳—۳ به بحث مکان‌یابی یک عامل در محیط عملکرد خود می‌پردازد که اولین قدم در ایجاد دنیای اطراف است. بخش ۳—۴ مجموعه اعمال سطح بالا را که یک بازیکن می‌تواند به عنوان یک عامل مستقل انجام دهد بررسی می‌کند که این مجموعه اعمال شامل توانایی در حرکت با توپ، قطع توپ، شوت کردن توپ و پاس دادن می‌باشند، در بخش ۳—۵ الگوی عملکرد تیمی روبوسینا در جاگیری بازیکنان بررسی می‌شود، در بخش ؟؟ روشهای تدافعی و در نهایت در بخش ؟؟ روشهای تهاجمی به عنوان اصلی‌ترین و برجسته‌ترین مفاهیم در محیط چند عامله بررسی می‌شوند. بخش ۳—۶ نیز جمع‌بندی و

نتیجه‌گیری بر مفاهیم مطرح شده در این فصل می‌باشد.

۳—۱ معماری سیستم

معماری سیستم در یک عامل روبوسینا مشابه با مدل مورد استفاده در سایر تیم‌ها می‌باشد. در چنین مدلی یک روش پائین به بالا و افزایشی [۱] مورد استفاده قرار می‌گیرد که در آن سیستم از نگاه **رفتاری**[۲]، **کاربردی**[۳] و **پیاده‌سازی**[۴] بررسی می‌شود[۲۱ ،۵]. در نگاه رفتاری آن‌چه که باید سیستم در جریان عملکرد از خود بروز دهد بررسی می‌شود، در نگاه کاربردی الگویی که به واسطه‌ی آن سیستم درک می‌شود، ضرورت‌های آن سنجیده می‌شود و نیز انطباق هر تعریف منطقی به پیمانه‌ی معادلِ قابل درک برای ماشین صورت می‌گیرد، شناسایی می‌شود و در نهایت در نگاه پیاده‌سازی پیمانه‌های بدست آمده از مرحله‌ی قبل برای پیاده‌سازی و عمل در مدل‌های سخت‌افزاری و نرم‌افزاری آماده می‌شوند. در این بخش نگاه اجمالی به سیستم را از زاویه‌ی رفتاری در دستور کار قرار می‌دهیم. در این چنین روندی مجموعه ضرورت‌هایی که در ایجاد یک عامل فوتبالیست باید مد نظر قرار بگیرند را به اختصار شرح می‌دهیم.

همان‌طور که در ابتدا اشاره شد مدل مورد استفاده در تیم روبوسینا از نگاه رفتاری مشابه با سایر تیم‌هاست. مدل مرسوم برای سیستم در این حالات یک مدل لایه‌ای پائین به بالاست که در آن هر لایه اطلاعات را از لایه‌ی زیرین دریافت کرده و پس از پردازش اطلاعات و تعامل با لایه‌ی بالاتر (در صورت وجود)، دستورات را به لایه‌ی پائین ارسال می‌کند. به طور کلی در چنین معماری مجموعه لایه‌های زیر برای سیستم تعریف می‌شوند.

- لایه‌ی ارتباطات (Connection Layer)
- لایه‌ی دنیای اطراف (World Model Layer)
- لایه‌ی مهارت‌ها (Skills Layer)
- لایه‌ی تصمیم‌گیری (Decision Layer)

۳—۱—۱ لایه‌ی ارتباطات

در پائین‌ترین لایه، لایه‌ی ارتباطات قرار دارد. وظیفه‌ی اصلی در این لایه ارتباط شبکه‌ای با کارگزار فوتبال و دریافت و ارسال پیام می‌باشد. در اصل اطلاعات از طریق این لایه به سیستم وارد می‌شوند و پس از عبور از لایه‌های بالاتر و اِعمال پردازش به این لایه بازگشته و پاسخ مناسب برای محیط کارگزار را ارسال می‌کنند. در این لایه اطلاعات دریافتی از سوی کارگزار تجزیه[۵] می‌شوند و برای استفاده از سوی لایه‌ی بالاتر در

[۱] incremental
[۲] functional
[۳] operational
[۴] implementational
[۵] Parse

ساختمان‌های مناسبی از داده ذخیره می‌شوند. پس از انجام عمل پردازش نیز نتیجه‌ی نهایی در این لایه به شکلی قابل فهم برای کارگزار بدل شده و ارسال می‌شود. در این لایه، چنان‌که در بخش ۳—۲ بدان خواهیم پرداخت، کنترل پیغام‌ها به صورت برون‌خطی[۶] [۱] انجام می‌شود، یعنی حلقه‌ای وجود ندارد که دائماً رسیدن پیغام را کنترل کند بلکه بررسی دریافت پیغام با استفاده از وقفه‌های سیستم‌عامل[۷] صورت می‌گیرد. پیغام‌های دریافت شده در ادامه به یک تابع تجزیه‌کننده ارسال می‌شوند که عمل جداسازی داده‌های مفید را انجام می‌دهد.

۳—۱—۲ لایه‌ی دنیای اطراف

در این لایه عملیات ایجاد درک از محیط اطراف برای بازیکن انجام می‌شود که به واسطه‌ی آن بازیکن قادر می‌شود تا یک ذهنیت کلی از فضای اطراف خود را به دست آورد. در این لایه اطلاعات دریافتی توسط حسگرهای مختلف پردازش شده و به شکلی قابل درک برای بازیکن تبدیل می‌شوند. World Model در اصل ساختمانی از موقعیت و سرعت اشیاء مؤثر در زمین به همراه تقریبی از حرکت‌های آتی و رخدادهای آینده می‌باشد. آن‌چه وظیفه‌ی این لایه از سیستم را بسیار سنگین و کلیدی می‌کند وجود خطا در حسگرهاست. در واقع باید لایه‌ی World Model مجموعه‌ای از سیاست‌ها را در جهت کنترل و کاهش خطا و ایجاد درک درست از محیط فراهم کند.

۳—۱—۳ لایه‌ی مهارت‌ها

این لایه شامل مجموعه‌ای از اعمال است که به بازیکن امکان می دهد تا در زمین بازی فعالیت کند. اَعمال این لایه به طور کلی استفاده از داده‌های ذخیره شده در World Model و پردازش آن‌ها و در نهایت ایجاد یک دستور قابل فهم برای کارگزار است. عمل انتخاب بین مجموعه‌ی رفتارهای موجود در این لایه از سوی لایه‌ی بالاتر انجام می‌شود و پس از آن عمل انتخاب شده پردازش و ایجاد نتیجه‌ی خروجی را بر عهده می‌گیرد. این مجموعه از دستورات عمدتاً به ۳ بخش سطح پائین، میانه و بالا تقسیم می‌شوند که در بخش ۳—۴ به تفصیل به توضیح مجموعه‌ی سطح بالا خواهیم پرداخت.

۳—۱—۴ لایه‌ی تصمیم‌گیری

لایه‌ی تصمیم‌گیری بالاترین لایه در مدل رفتاری بازیکن است که در آن سیاست‌های کلی برای عملکرد بازیکن اتخاذ می‌شوند. در این لایه داده‌های دریافتی و ذخیره شده در World Model از یک نگاه کلی و بنا به گزینه‌های ممکن در عملکرد بازیکن بررسی می‌شوند و در نهایت با توجه به تشخیص این لایه یکی از اعمال سطح بالای لایه‌ی مهارت برای انجام تأیید می‌شود. این لایه عمدتاً در یک قالب قانون‌مدار عمل می‌کند و به بیان بهتر از یک ساختار ماشین حالت قطعی و متناهی بهره می‌گیرد که با رخدادِ یک وضعیت خاص ماشین تصمیم‌گیری را به یک حالت مشخص و در واقع یک عمل سطح بالای معین هدایت می‌کند.

---

[۶] offline

[۷] Interrupt

۳—۲ همزمان‌سازی با محیط زمان‌حقیقی

اولین گام در پیاده‌سازی یک سیستم زمان‌حقیقی ایجاد امکان دریافت داده‌های محیطی، پردازش اطلاعات و انجام عمل مناسب در بازه زمانی مجاز است که معمولاً این فرصت زمانی از حد ثانیه تجاوز نمی‌کند. سیستم کارگزار برای شبیه‌سازی یک محیط زمان‌حقیقی از ساختار زمانی خاصی استفاده می‌کند که در آن هر بازیکن یک سیکل یعنی ۱۰۰ میلی ثانیه برای انجام مجموعه اعمال فوق فرصت در اختیار دارد و هماهنگ شدن بازیکن در انجام کل فعالیت‌های بالا با کارگزار **همزمان شدن**[۸] با کارگزار می‌باشد. هرچه همزمان شدن با کارگزار به شکل دقیق‌تری انجام شود یک عامل در انجام فعالیت‌های خود تواناتر می‌شود و توانایی مجموعه‌ی عوامل در انجام عمل مناسب به عملکرد تیمی درست می‌انجامد.

همان طور که اشاره شد سیستم کارگزار فضای بازی را به مجموعه‌ای از بازه‌های زمانی تقسیم می‌کند که هر جزء از این فضا ۱۰۰ms طول می‌کشد. در واقع این الگو زمان بازی را به یک فضای گسسته می‌نگارد و در هر ۱۰۰ms عامل می‌تواند یک عمل اصلی را به همراه مجموعه‌ای از اعمال همزمان به کارگزار ارسال کند. برای اطلاع در مورد اعمال اصلی و همزمان به بخش ۲—۲—۳ رجوع شود. چنانچه در این بازه دستوری به کارگزار ارسال نشود بازیکن یک فرصت عمل را از دست داده است و در این حال با یک **حفره**[۹] در عملکرد بازیکن مواجه هستیم. اگر بازیکن در این بازه زمانی بیش از یک دستور اصلی به سیستم ارسال کند نیز یکی از اعمال به طور تصادفی اجرا می‌شود که در این حال با **تصادم**[۱۰] در داده‌های ارسالی رو به رو می‌شویم[۵]. اهمیت همزمان‌سازی زمانی بیشتر روشن می‌شود که درمی‌یابیم علاوه بر تنظیم عمل در یک سیکل از بازی، استفاده بهینه از داده‌های جدید و به روز با انجام همزمان‌سازی درست صورت می‌گیرد که عملکرد فردی و تیمی را به شکل قابل ملاحظه‌ای تحت تأثیر قرار می‌دهد.

همزمان کردن یک بازیکن با کارگزار مرحله‌ای از ایجاد یک عامل فوتبالیست است که به موجب آن عملکرد عامل به سخت‌افزار سیستم وابسته می‌شود و طبعاً استفاده از ماشین‌هایی با پردازنده‌های تواناتر و حافظه‌ی اصلی بزرگتر چالش‌های کمتری در مواجهه با مدل زمان‌حقیقی در سیستم شبیه‌سازی ایجاد می‌کند[۱۱]. اما طراحی عامل باید به گونه‌ای باشد که امکان عمل روی یک ماشین با سرعت متوسط را نیز داشته باشد.

از دیگر ویژگی‌هایی که مدل زمان‌بندی کارگزار از آن برخوردار است غیرهمزمان بودن دریافت داده‌های دیداری و جسمی است. داده‌های دیداری و اطلاعات مربوط به حسگرهای بینایی که مهم‌ترین عامل در تصمیم‌گیری روبات می‌باشند با توجه به فرکانس دیداری مطرح شده در ۲—۲—۱ از سوی کارگزار به بازیکن ارسال می‌شوند و زمان شروع یک سیکل از بازی لزوماً با زمان دریافت این داده‌ها یکی نیست بنابراین درک این مطلب که آیا امکان استفاده از یک داده‌ی دیداری در یک سیکل از بازی وجود دارد یا خیر برای بازیکن بسیار مهم است. اشتباه در محاسبه این فرصت می‌تواند باعث ایجاد حفره در عمل و یا تصمیم‌گیری بر پایه داده‌های قدیمی و نامطمئن شود. از سوی دیگر همواره و در ابتدای هر سیکل از بازی

---

[۸] Synchronization

[۹] hole

[۱۰] clash

[۱۱] در مسابقات جهانی روبوکاپ ۲۰۰۳ ایده‌ی طراحی تیم‌هایی با قابلیت اجرا روی یک دستگاه LapTop مطرح شد تا چالش در جهت ایجاد تیم‌های با توان همزمان شدن در شرایط دشوار شکل گیرد. این ایده در مسابقات آزاد آمریکا در سال ۲۰۰۴ به شکل عملی اجرا شد.

یک پیام sense_body از طرف کارگزار به عامل ارسال می‌شود و این بدان معنی است که با دریافت هر پیام sense_body، ۱۰۰ میلی ثانیه زمان عامل برای تصمیم‌گیری شروع می‌شود. بررسی‌های انجام شده نشان می‌دهد که فاصله زمانی بین دو داده‌ی از یک نوع به مقدار خیلی زیاد به فاصله زمانی مورد انتظار نزدیک است (به عنوان نمونه خطای کمی در زمان دریافت دو داده یکسان sense_body وجود دارد) و همواره یک پیام sense_body قبل از یک پیام see به عامل می‌رسد [۵]؛ بنابراین هر داده‌ی see مربوط به سیکلی از بازی است که پیام sense_body آن پیشتر آمده است .در ادامه روشی را که روبوسینا برای همزمان‌سازی مورد استفاده قرار داد مطالعه می‌کنیم.

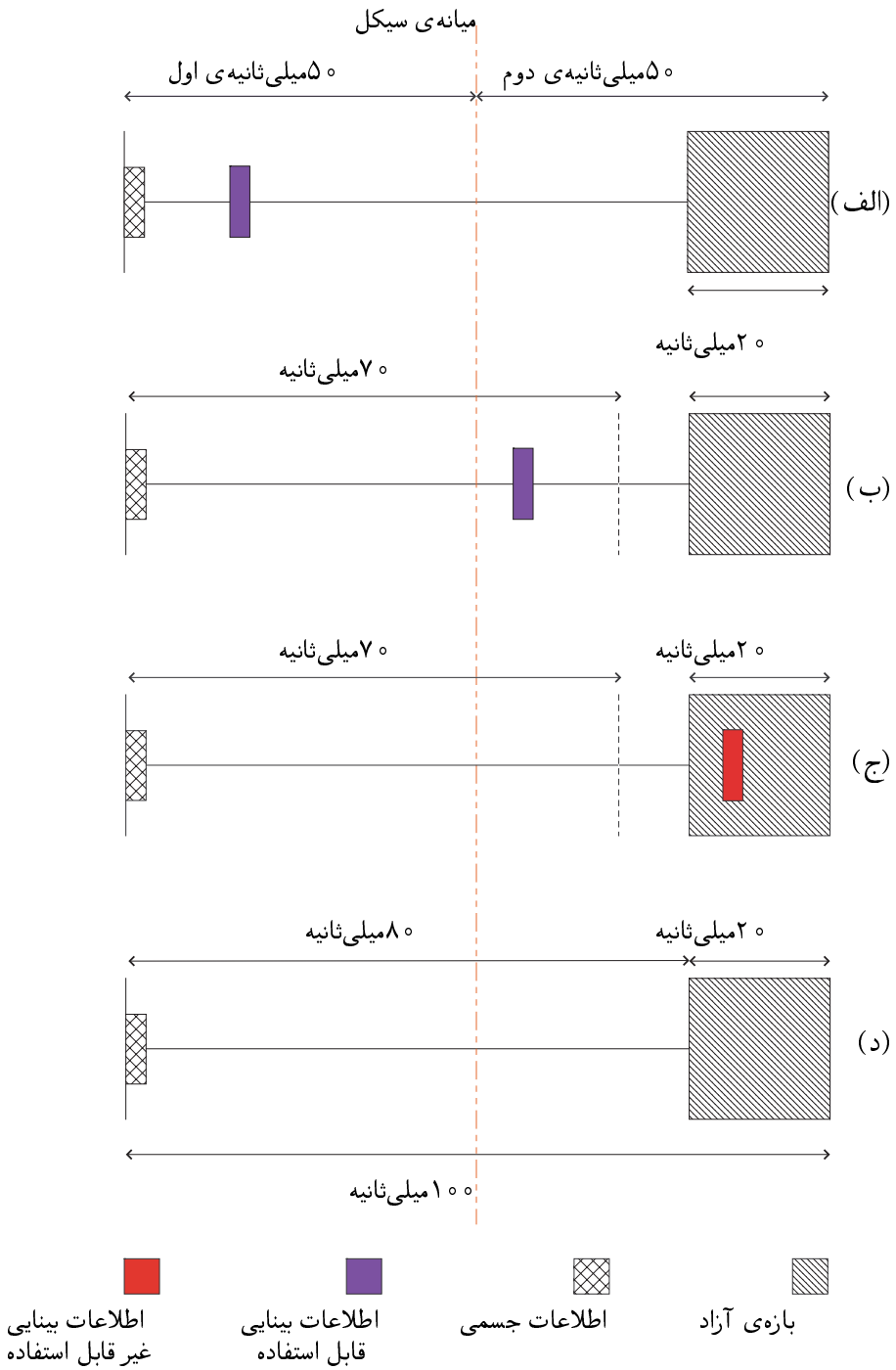
۳—۲—۱ روش همزمان شدن

روش‌های متعددی برای همزمان‌سازی توسط تیم‌های مختلف ارائه شده است که می‌توان تعدادی از آنها را در مراجع [۵، ۱۳] مطالعه کرد. در این بخش روش مورد استفاده در تیم روبوسینا را بررسی می‌کنیم. مزایا و معایب این روش در بخش بعد مورد مطالعه قرار خواهد گرفت.

همان طور که اشاره شد در یک الگوی زمان‌بندی باید سازمان‌دهی برنامه به شکلی باشد که امکان حداکثر استفاده از زمان برای دریافت داده و پردازش آن وجود داشته باشد. برقراری ارتباط با کارگزار از طریق اتصال UDP/IP انجام می‌شود و بازیکن باید همواره پورت ارتباط با کارگزار را مورد بررسی قرار دهد تا داده‌ی جدیدی از دست نرود. از سوی دیگر باید پردازش داده و انجام عمل نیز در این زمان صورت گیرند. با توجه به اینکه امکان عدم دریافت داده وجود دارد بنابراین باید روند حرکت اشیاء زمین با استفاده از داده‌های قبلی پیش‌بینی شود تا در صورت نیاز مورد استفاده قرار بگیرد. برای بررسی شروع یک سیکل از بازی مناسب‌ترین راه استفاده از پیام sense_body است چرا که دلیل بر شروع سیکل جدیدی از بازی است. با ذخیره فرکانس دیداری تعیین شده از سوی بازیکن می‌توان فواصل دریافت پیام see را نیز محاسبه کرد و امکان وجود پیام دیداری در به روز رسانی دنیای اطراف را سنجید. شکل ۳—۱ روند عمل در یک سیکل بازی را نشان می‌دهد.

با توجه به شکل، وجود داده‌ی see در یک سیکل سه وضعیت را داراست. وجود داده‌ی see در نیمه اول یک سیکل (شکل ۳—۱—(الف))، وجود داده در ۵۰میلی‌ثانیه دوم یک سیکل (شکل ۳—۱—(ب) و ۳—۱—(ج)) و یا نبود see در یک سیکل (شکل ۳—۱—(د)). لازم به ذکر است که ممکن است در یک سیکل از بازی بیش از یک داده‌ی see با توجه به فرکانس دیداری تنظیم شده وجود داشته باشد اما از آنجا که داده‌های بینایی دریافتی در یک سیکل کاملاً با هم مشابه هستند ملاک برای اِعمال تقسیم‌بندی فوق زمان اولین داده‌ی دریافتی است. ضمناً با توجه به شکل ۲۰میلی‌ثانیه‌ی انتهای هر سیکل زمان آزاد در ارسال داده است که بازیکن در این فاصله به جز تجزیه‌ی داده‌های دریافتی هیچ عمل پردازشی دیگری انجام نمی‌دهد. دلیل استفاده از این بازه‌ی آزاد تلاش تمام عوامل در ارسال داده به پورت کارگزار است که می‌تواند سبب ایجاد ترافیک در پورت کارگزار شود؛ لذا ارسال زودتر داده می‌تواند از این ترافیک بکاهد و دریافت داده توسط کارگزار را تضمین کند.

در شرایطی که داده‌ی see در ۵۰میلی‌ثانیه اول دریافت شود زمان کافی برای پردازش اطلاعات وجود دارد و بازیکن با تکیه بر داده‌ی جدید عمل تصمیم‌گیری را انجام می‌دهد. در صورتی که داده در نیمه‌ی دوم سیکل دریافت شود با دو حالت روبرو هستیم. حالت اول زمانی است که داده در ۲۰میلی‌ثانیه‌ی



شکل ۳—۱: وضعیت‌های مختلف دریافت داده در یک سیکل از بازی

ابتدای این بازه دریافت شود (شکل ۳—۱—(ب))که در این حالت زمان برای تصمیم‌گیری وجود دارد و می‌توان بر اساس داده‌ی دریافتی جدید تصمیم‌گیری نمود. در این حال ممکن است عمل پردازش داده‌ها وارد بازه‌ی آزاد نیز بشود، اما از آنجا که تعداد کمی از بازیکنان در این حالت از زمان‌بندی قرار می‌گیرند چندان مشکل‌ساز نخواهد بود. حالت دوم زمانی است که داده‌ی see در ۳۰میلی‌ثانیه‌ی انتهای یک سیکل دریافت شود که این وضعیت در شکل ۳—۱—(ج) نشان داده شده است. در این شرایط استفاده از داده‌ی see صرف‌نظر می‌شود و داده ذخیره می‌شود تا در سیکل بعدی با اِعمال پیش‌بینی روی آن مورد استفاده قرار بگیرد. در نهایت این امکان هست که داده‌ی see در یک سیکل از بازی مشابه شکل ۳—۱—(د) وجود نداشته باشد که در این شرایط تصمیم‌گیری باید روی داده‌های قدیمی که روند حرکت آنها یک سیکل به جلو برده شده است انجام شود. توجه به این نکته ضروری است که تقسیم‌بندی زمانی با توجه به فرصت زمانی که یک عامل روبوسینا برای تصمیم‌گیری نیاز دارد انجام شده است و اعداد برای استفاده از الگوریتم‌های تصمیم‌گیری دیگر قابل تغییر می‌باشند.

برای دستیابی به روش زمان‌بندی فوق از سیگنال‌های سیستمی قابل استفاده در سیستم عامل استفاده شد. سیستم عامل لینوکس مجموعه‌ای از سیگنال‌ها را به منظور کنترل عملکرد اجزاء مختلف مورد استفاده قرار می‌دهد که در این میان دو سیگنال I/O و Alarm برای ایجاد امکانات فوق مناسب هستند. برای مطالعه در خصوص سیگنال‌ها می‌توان به [۱۵] رجوع کرد. سیگنال Alarm به این شکل عمل می‌کند که با مشخص کردن بازه‌ی زمانی خاصی برای سیستم توسط برنامه، سیستم می‌تواند به طور متناوب با سپری شدن این بازه زمانی به برنامه Alarm دهد. در پیاده‌سازی مدل بالا بازه‌ی Alarm ۱۰میلی‌ثانیه تنظیم شد و یک شمارنده این فواصل زمانی را ثبت می‌کرد که با دریافت هر پیام sense_body شمارنده صفر می‌شد و سیکل جدیدی آغاز می‌شد. با استفاده از این روش یک سیکل در ۱۰ بازه‌ی ۱۰میلی‌ثانیه‌ای قابل بررسی بود. سیگنال I/O زمانی رخ می‌دهد که روی پورت‌های ورودی سیستم داده قرار بگیرد و از آن می‌توان برای تشخیص دریافت داده از سوی کارگزار استفاده کرد.

مسئله دیگری که در همزمان کردن بازیکن وجود دارد رعایت مفاهیم پایه‌ای مربوط به همزمان شدن است. داده‌های دریافتی از کارگزار پس از تجزیه در ساختمان داده‌ای ذخیره می‌شوند و در شرایطی با انجام عمل پیش‌بینی یک سیکل به جلو برده می‌شوند. این ساختمان داده بخش بحرانی در عمل تحلیل و ذخیره داده است و باید تضمین شود که مشکلات اصلی در همزمان‌سازی یعنی **انحصار متقابل**، **پیشرفت** و **گرسنگی** برای آن مرتفع است. روش‌های ارائه شده توسط سایر تیم‌ها استفاده از **نخ‌کشی** را برای تضمین شرایط بالا پیشنهاد می‌کند (رجوع شود به [۵]) اما روش استفاده شده در عامل روبوسینا بدون نیاز به نخ‌کشی عمل درست فرآیندها را تضمین می‌کند.

این روش انحصار متقابل را پشتیبانی می‌کند چرا که در هر لحظه یک فرآیند اجرا می‌شود و بقیه‌ی سیگنال‌ها قفل می‌شوند که بواسطه آن تغییر در داده‌های ذخیره شده ایجاد نمی‌شود. از سوی دیگر با توجه به پیش‌بینی برای رسیدن یک پیام see از روی فرکانس دیداری احتمال اینکه داده‌ی جدید see در وسط پردازش داده‌های قبلی برسد صفر می‌باشد و بنابراین هیچ‌گاه دو پروسه به شکل همزمان وارد ساختمان داده نمی‌شوند. پیشرفت نیز توسط این روش پشتیبانی می‌شود چرا که با ورود یک داده سایر پروسه‌ها قفل می‌شوند و بنابراین همواره پروسه‌ای که در ناحیه‌ی بحرانی وارد می‌شود برای عمل اولویت پیدا می‌کند. در نهایت مشکل گرسنگی بین دو پروسه وجود نخواهد داشت چرا که محاسبه زمانی مانع از انتظار نامتناهی یکی از پروسه‌ها برای ورود به ناحیه بحرانی می‌شود و از طرف دیگر با وجود سیگنال I/O داده پردازش می‌شود، اما، اِعمال تغییر روی ناحیه‌ی بحرانی بسته به شرایط بازی و زمان باقی‌مانده در ادامه‌ی سیکل دارد.

مزایای اصلی این روش در ایجاد دسترسی دقیق به بخش‌های مختلف یک سیکل از بازی می‌باشد.

همان‌طور که اشاره شد این ویژگی باعث می‌شود تا به سادگی بتوان با پیش‌بینی زمان رسیدن یک داده‌ی see امکان استفاده یا صرف‌نظر از آن در سیکل جاری را بررسی کرد. از سوی دیگر می‌توان زمان دقیق عملکرد پروسه‌های مختلف را مورد مطالعه قرار داد و با تخصیص زمان مورد نیاز به هر پروسه میزان اثر آن در جریان عملکرد عامل را سنجید. اما از سوی دیگر این روش به شدت به ساعت ماشین وابسته است و اگر پردازنده به هر دلیلی زیر فشار قرار بگیرد و نتواند در بازه‌های زمانی درستی پیام Alarm را به شکل یک سیگنال به برنامه گزارش دهد در این صورت زمان‌بندی به طور کامل دچار درهم ریختگی می‌شود و فشار اعمال شده به شکل قابل ملاحظه‌ای تعداد حفره‌ها در بازی را بالا می‌برد. علت اصلی این مشکل عدم به روز شدن به موقع شمارنده‌ی زمانی در یک سیکل است که به دلیل وابستگی شدید همه پروسه‌ها به این شمارنده، که ایجاد زمان‌بندی خاصی را مطابق اشکال ۳—۱ موجب شده است، سبب اشکال در اجرای درست و به موقع هر پروسه می‌شود. با دریافت داده‌ی sense_body چنانچه کل عملکرد پردازنده‌ی ماشین با مشکل مواجه نشده باشد می‌توان با مقداردهی مجدد شمارنده به صفر به پی‌گیری روال عادی امیدوار بود اما چنانچه روند پاسخ برنامه به Alarm ارسالی از پردازنده شکل درست و منظم پیدا نکند همزمان‌سازی با مشکل مواجه می‌شود. جدول ۳—۱ روند افزایش حفره‌ها با زیاد شدن تعداد پروسه‌ی بازیکنان و افزایش بار پردازنده مرکزی را نشان می‌دهد.

جدول ۳—۱: روند افزایش تعداد حفره‌ها و تصادم‌ها در الگوی همزمان شدن یک عامل روبوسینا

n (تعداد پروسه‌ی بازیکنان فعال)	c (تعداد سیکل‌های سپری شده)	تعداد حفره‌ها	تعداد تصادم‌ها
۱	۶۰۰۰	۰	۰
۵	۳۰۰۰	۵	۲
۱۱	۳۰۰۰	۲۷	۷
۲۲	۳۰۰۰	۲۰۶	۱۱

جدول فوق از اجرای بازیکنانی به تعداد n در جدول ۳—۱ روی یک LapTop با پردازنده مرکزی ۲٫۸GHz و حافظه‌ی اصلی ۲۵۶MB حاصل شده است و تعداد حفره‌ها و تصادم‌ها برای یکی از این عوامل در مدت زمان c سیکل بازی بررسی شده است. همان طور که در جدول مشاهده می‌شود با دو برابر شدن تعداد فرآیندها تعداد حفره‌ها به شکل قابل ملاحظه افزایش می‌یابد که بیانگر وابستگی الگوریتم به بار روی CPU است. در شرایطی که تعداد پروسه‌ها از ۵ به ۱۱ افزایش می‌یابد در تعداد حفره‌ها رشد ۴۴۰ درصدی مشاهده می‌شود و زمانی که تعداد از ۱۱ به ۲۲ افزایش می‌یابد این رشد حدود ۶۶۳ درصد می‌باشد که نسبت به مقدار قبلی ۲۲۳ درصد افزایش داشته است. الگوی عملکرد در یک عامل روبوسینا به نحوی طراحی شده است که تصادم در مقایسه با تعداد حفره‌ها چندان زیاد نیست.

۳—۳ تعیین موقعیت روبات

مکان‌یابی از نخستین مفاهیمی است که در شکل‌دهی یک مدل از دنیای اطراف بازیکن مطرح می‌شود. در واقع یک روبات پیش از آن که بتواند تخمینی نسبت به محل سایر اشیاء و عوامل موجود در زمین داشته باشد باید دریافتن وضعیت خود در محیط عمل توانا باشد. محیط شبیه‌سازی با اعمال خطا در داده‌های دیده شده فضایی را ایجاد می‌کند که در آن می‌توان به جستجوی راهکارهایی برای حل مسئله مکان‌یابی نیز پرداخت. در این خصوص نمونه‌هایی از فعالیت‌های ارائه شده در سایر تیم‌ها در مراجع [۵] و [۱] آمده‌اند که به دلیل پیچیدگی‌های عملی، دشواری در پیاده‌سازی و یا عدم توضیح دقیق، چندان قابل استفاده نیستند. در این بخش

به بررسی الگوریتم‌های استفاده شده در تیم روبوسینا به منظور مکان‌یابی یک عامل در زمین بازی می‌پردازیم.

با توجه به آنچه در بخش حسگرهای بینایی توضیح داده شد (۲—۲—۱) کارگزار به شکل خودکار در رؤیت اشیاء موجود در زمین خطایی را وارد می‌کند که عدم قطعیت در درک اشیاء موجود در زمین را موجب می‌شود. برای مکان‌یابی یک عامل در زمین مناسب‌ترین راه استفاده از اشیاء ثابت با موقعیت‌های معلوم است و در سیستم شبیه‌سازی استفاده از پرچم‌ها این امکان را حاصل می‌کند. پرچم‌ها اشیاء ثابت با مکان مشخص هستند که در اطراف زمین بازی نسب شده‌اند و هر داده دریافتی از حسگرهای بینایی شامل فاصله و زاویه با تعدادی از این پرچم‌هاست. بی‌شک بیشتر بودن تعداد پرچم‌ها موجب تخمین دقیق‌تر از مکان بازیکن خواهد بود.

از آنجا که داده‌ی حاوی اطلاعات پرچم‌ها نیز دارای خطا می‌باشد نمی‌توان از این داده به طور مستقیم برای استنباط مکان بازیکن استفاده کرد و به همین منظور باید روشی برای کاهش میزان خطا به کار گرفت. فرمول‌های مورد استفاده برای اعمال خطا در فاصله و زاویه پرچم به شکل زیر توسط کارگزار مورد استفاده قرار می‌گیرند:

$$Quantize(V,Q) = round(\frac{V}{Q}) \times Q$$

$$Q\_Distance = Quantize(e^{(Quantize(ln(Distance),۰/۰۱))}, ۰/۱)$$

$$Q\_Direction = Quantize(Direction, ۱/۰)$$

که در آن تابع $Quantize$ برای اعمال خطا در سیستم به کار می‌رود، $Distance$ و $Direction$ فاصله و زاویه حقیقی بین بازیکن و پرچم و $Q\_Distance$ و $Q\_Direction$ فاصله و زاویه بین بازیکن و پرچم پس از اعمال خطا می‌باشند. برای بدست آوردن فضایی که بازیکن با احتمال ۱/۰ در آن وجود دارد کافی است تا فرمول $Quantize$ را معکوس کنیم. با این عمل خواهیم داشت:

$$Quantize^{-۱}_{min}(V,Q) = (round(\frac{V}{Q}) - ۰/۵) \times Q$$

$$Quantize^{-۱}_{max}(V,Q) = (round(\frac{V}{Q}) + ۰/۵) \times Q$$

با استفاده از دو رابطه فوق و با توجه به این امر که $Q\_Distance$ و $Q\_Direction$ توسط حسگرهای بینایی در اختیار بازیکن قرار می‌گیرند می‌توان d_{min} و d_{max} را به عنوان کمترین و بیشترین فاصله‌ای که بازیکن به طور حتم در آن وجود دارد به شکل زیر محاسبه کرد:

$$d_{min} = Quantize^{-۱}_{min}(\ln(Quantize^{-۱}_{min}(Q\_Distance, ۰/۱)), ۰/۰۱)$$

$$d_{max} = Quantize^{-۱}_{max}(\ln(Quantize^{-۱}_{max}(Q\_Distance, ۰/۱)), ۰/۰۱)$$

و برای $Distance$ خواهیم داشت:

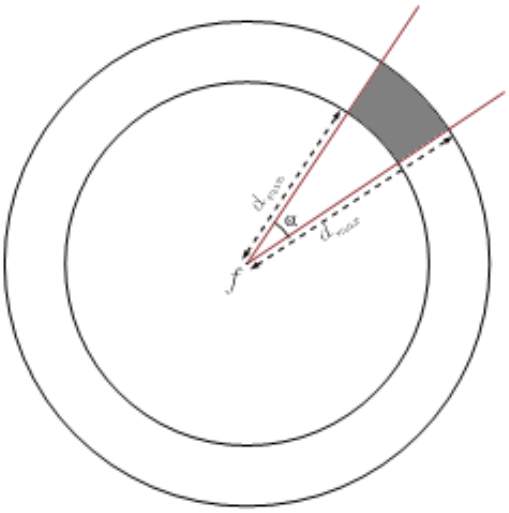
$$d_{min} \leq Distance \leq d_{max}$$

به شکل مشابه برای $Direction$ نیز خواهیم داشت:

$$(\theta_{min} = Q\_Direction - ۱) \leq Direction \leq (Q\_Direction + ۱ = \theta_{max})$$

که در آن θ_{min} و θ_{max} کمترین و بیشترین مقدار ممکن برای زاویه‌ایست که پرچم می‌تواند تحت آن زاویه دیده شود.

d_{min} و d_{max} را می‌توان شعاع دو دایره هم‌مرکز به مرکزیت پرچم در نظر گرفت که بازیکن در فضای بین این دو دایره قرار دارد و θ_{min} و θ_{max} نیز دو شعاع از این دایره هستند که دو درجه نسبت به هم اختلاف زاویه دارند و بازیکن در قطاعی قرار می‌گیرد که این دو شعاع می‌سازند. با این مجموعه اطلاعات می‌توان فضایی را ایجاد کرد که بازیکن با احتمال ۱/۰ در آن وجود دارد(شکل ۳—۲).



شکل ۳—۲: ناحیه حاشور خورده فضای محتمل برای وجود بازیکن را نشان می‌دهد. پرچم در مرکز دایره‌های ایجاد شده قرار دارد و شعاع دایره‌ها d_{min} و d_{max} می‌باشد. زاویه مرکزی $\alpha = \theta_{max} - \theta_{min} = ۲^\circ$

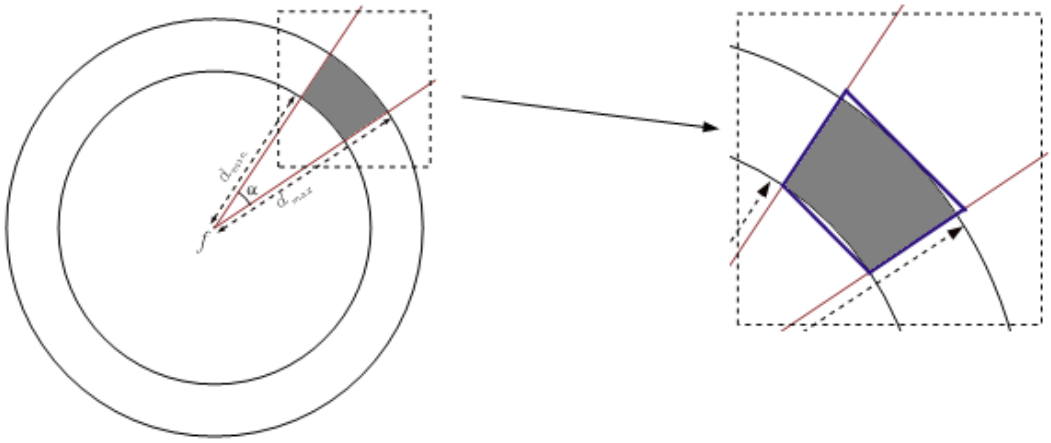
برای هر پرچم درون زمین می‌توانیم یک قطاع مشابه شکل فوق به دست آوریم و با توجه به اینکه بازیکن در تمام این فضاها وجود دارد بدون شک در اشتراک این مجموعه از فضاها نیز خواهد بود. بنابراین مسئله به اشتراک‌گیری بین مجموعه‌ی فضاهایی که از معکوس کردن تابع $Quantize$ ایجاد می‌شوند محدود می‌شود.

عمل اشتراک‌گیری بین مجموعه فضاهایی نظیر شکل ۳—۲ از آنجا که این فضا فاقد شکل هندسی منظم است چندان ساده نیست پس ناگزیر هستیم روشی برای تبدیل این فضا به یک شکل هندسی منظم به کار گیریم. در [۲] دو روش بررسی شده توسط تیم روبوسینا ارائه شده‌اند که در این بخش الگوریتم ذوزنقه‌های محیطی را توضیح می‌دهیم[۱۲].

کوچکترین شکل محدبی که می‌توان در اطراف فضای محتمل برای وجود بازیکن در نظر گرفت

[۱۲] این الگوریتم در نهایت در تیم روبوسینا مورد استفاده قرار گرفت و با توجه به سرعت اجرا و دقت کاملاً مفید و مناسب برای پیاده‌سازی است.

ذوزنقه‌ی محیط کننده این فضا است. فضای حاصل از این ذوزنقه وجود بازیکن را تضمین می‌کند. شکل ۳—۳—ب ذوزنقه‌ی محیطی ایجاد شده روی فضای حاصل از شکل ۳—۳—الف را نشان می‌دهد.

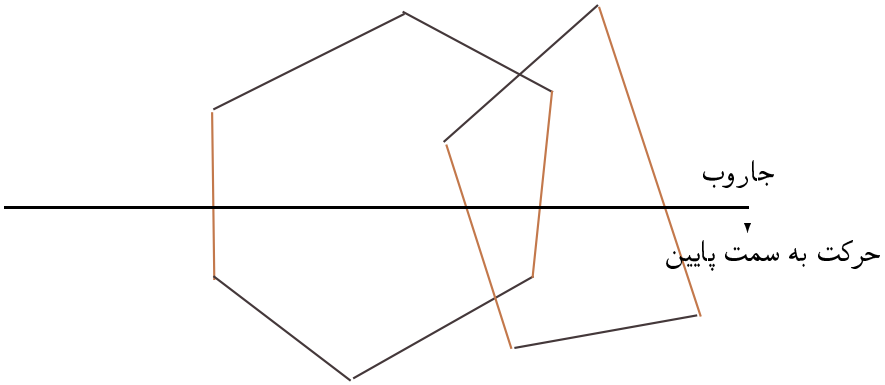


(الف) (ب)

شکل ۳—۳: (الف) تبدیل فضا به یک ذوزنقه محدب (ب) فضای ذوزنقه‌ای محتمل برای وجود بازیکن با خطوط آبی رنگ مشخص شده‌است. به دلیل برخورداری این فضا از ساختار هندسی محدب، الگوریتم‌های هندسه محاسباتی به سادگی روی آن قابل اعمال است.

با تبدیل فضا به یک ذوزنقه‌ی محدب می‌توان از روش‌های هندسه محاسباتی برای اشتراک‌گیری روی این ذوزنقه‌ها استفاده کرد. الگوریتم‌های متعددی برای اشتراک‌گیری روی چندضلعی‌ها وجود دارد که آخرین آنها توسط آقای de Burg و سایرین در [۱۲] ارائه شده است و در آن، زمان محاسبه اشتراک بین دو چندضلعی $O(m+n)$ معرفی شده‌است که m و n به ترتیب تعداد رئوس هریک از دو چندضلعی می‌باشد. ایده اصلی که در پس این طرح وجود دارد استفاده از یک خط جاروب در اشتراک‌گیری روی دو چندضلعی است. این خط از بالا به سمت پائین شروع به حرکت می‌کند و در هر لحظه با ۴ یال از دو چندضلعی که با هم اشتراک دارند برخورد می‌کند که این یال‌ها را **یال‌های فعال** می‌نامیم (شکل ۳—۴). با استفاده از حرکت خط جاروب و تشخیص یال‌های فعال می‌توانیم چندضلعی محدب حاصل از اشتراک را بسازیم. با رسیدن به انتهای هر یال از این ۴ یال فعال به نقطه‌ی جدیدی می‌رسیم که یال قبلی را از مجموعه یال‌های فعال پاک می‌کند و یال جدیدی را به مجموعه می‌افزاید که در این حالت هم دامنه‌ی یال‌های فعال و هم وضعیت خط جاروب بروز رسانی می‌شوند. جزئیات این روش را می‌توان در [۱۲] مطالعه کرد.

روشی که در تشخیص اشتراک چندضلعی‌ها توسط تیم روبوسینا مورد استفاده قرار گرفت مشابه با روش مطرح شده در [۱۲] می‌باشد با این تفاوت که در این روش به جای تشخیص یال‌های فعال چندضلعی، نقاط سازنده‌ی یال‌های فعال شناسایی و به مجموعه اضافه می‌شوند. این روش تشخیص نقاط را ساده می‌کند ضمن این که پیچیدگی مربوط به مقایسه یال‌ها با یکدیگر را نیز ندارد. هر نقطه از یک چندضلعی که بین دو یال فعال چندضلعی دیگر قرار دارد از نقاط سازنده چندضلعی جدید است، ضمن این که نقاط برخورد دو یال از دو چندضلعی نیز به طور حتم در مجموعه نقاط چندضلعی مشترک می‌باشند. از آنجا که خط جاروب فقط یک بار از هر نقطه از چندضلعی می‌گذرد بنابراین مشابه با [۱۲] زمان اجرای

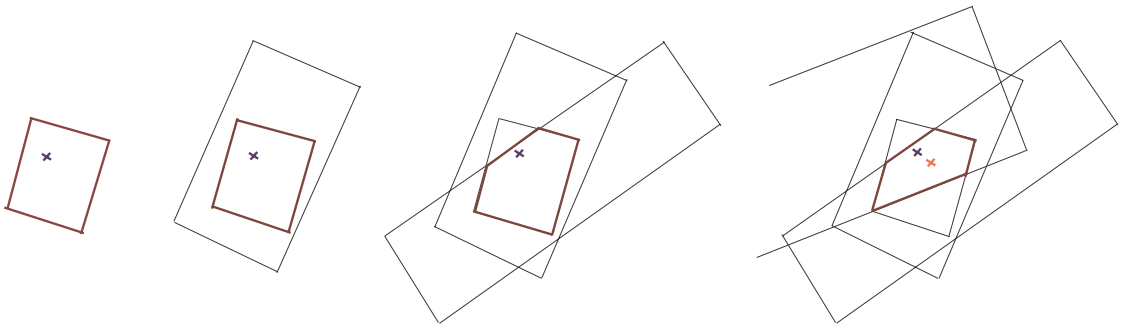


شکل ۳–۴: حرکت خط جاروب به سمت پایین و مشخص شدن یال‌های فعال

الگوریتم متناسب با مجموع تعداد رئوس چندضلعی‌ها و از مرتبه‌ی $O(m+n)$ می‌باشد. با استفاده از روش تقسیم و حل می‌توان اشتراک n چندضلعی محدب را در زمان $O(n \log n)$ بدست آورد. اگر $T(n)$ زمان اجرا برای اشتراک روی n چندضلعی محدب باشد اجرای کل الگوریتم $T(n) = ۲T(n/۲) + O(n)$ زمان خواهد برد که $O(n)$ زمان اشتراک‌گیری بین مجموعه دو چندضلعی بدست آمده است. با محاسبه این رابطه بازگشتی زمان اجرای کل الگوریتم $O(n \log n)$ به دست می‌آید.

الگوریتم اشتراک روی چندضلعی‌های محدب در زیر آورده شده است:

```
method mergeConvexPolygons
    set List to NULL
    set List۱ to Points in convex polygon CP۱
    set List۲ to Points in convex polygon CP۲
    set LeftList۱ and RightList۱ for left-side and right-side Points in List۱
    set LeftList۲ and RightList۲ for left-side and right-side Points in List۲
    while unchecked segments have remained do
        make l۱ and r۱ segments from LeftList۱ and RightList۱ according to SweepLine Position
        make l۲ and r۲ segments from LeftList۲ and RightList۲ according to SweepLine Position
        if start point p۱ for l۱ or r۱ is in between l۲ and r۲ then
            add start point p۱ to List
        end if
        if start point p۲ for l۲ or r۲ is in between l۱ and r۱ then
            add start point p۲ to List
        end if
        if there is any intersection point p۳ between segments then
            add intersected point p۳ to List
        move SweepLine down
    end while
```



شکل ۳—۵: اشتراک‌گیری روی ذوزنقه‌های محیطی حاصل از پرچم‌ها

جدول ۳—۲: میزان دقت و زمان اجرا در الگوریتم ذوزنقه‌های محیطی

الگوریتم مکان‌یابی	میانگین خطا(متر)	انحراف از معیار	زمان اجرا(میلی ثانیه)
الگوریتم ذوزنقه‌های محیطی	۰/۰۶۱	۰/۰۳۸	۰/۳۵۱

make convex polygon ICP from $List$

end method

با استفاده از روش اشتراک روی چندضلعی‌ها می‌توان کوچک‌ترین فضایی که بازیکن با احتمال ۱/۰ در آن قرار دارد را به دست آورد. شکل ۳—۵ نمونه‌ی عملی پیاده‌سازی و اجرای الگوریتم ذوزنقه‌های محیطی در محیط شبیه‌سازی فوتبال را که توسط کتابخانه LEDA[۱۳] مدل شده است نمایش می‌دهد. در این شکل عمل اشتراک‌گیری روی ذوزنقه‌های ایجاد شده توسط ۴ پرچم نمایش داده شده است و فضای ایجاد شده نهایی کوچک‌ترین فضای ایجاد شده از این روش است که بازیکن در آن وجود دارد. برای اینکه مختصات مکانی بازیکن را در این فضا حدس بزنیم روی رئوس چندضلعی نهایی میانگین می‌گیریم و نقطه به دست آمده را موقعیت دقیق بازیکن فرض می‌کنیم. در شکل ۳—۵ نتیجه عمل میانگین‌گیری نقطه‌ایست که با رنگ قرمز نشان داده شده است و نقطه آبی رنگ مکان اصلی بازیکن است که از داده $full\_state$[۱۳] استخراج شده است. جدول ۳—۲ میزان دقت الگوریتم ارائه شده را بررسی می‌کند. عملکرد این الگوریتم روی ماشینی با پردازنده پنتیوم ۱/۳GHz با حافظه اصلی ۲۵۶MB بررسی شده است.

۳—۴ مهارت‌های سطح بالا

سلسل مراتب برای توانایی عملکرد در یک بازیکن را می‌توان به سه مجموعه‌ی مهارت‌های سطح پایین، سطح میانه و سطح بالا تقسیم کرد که قابلیت بازیکن برای انجام مهارت‌های هر لایه منوط به کسب توانایی در لایه‌ی پائین‌تر می‌باشد و هر اندازه میزان دقت در یک عمل لایه پایین‌تر بیشتر باشد عملکرد بهتری در

[۱۳] $full\_state$ شامل مجموعه‌ای از اطلاعات بدون خطا در مورد تمام اشیاء موجود در زمین است که از سوی کارگزار و صرفاً به منظور تست در محیط آزمایشگاه در اختیار بازیکنان قرار می‌گیرد.

لایه بالاتر نتیجه خواهد شد. اَعمال لایه پایین شامل مواردی نظیر چرخش گردن یا بدن به سمت یک نقطه‌ی خاص، حرکت به یک نقطه خاص از زمین، متوقف کردن توپ، هم راستا کردن سر و بدن و ... می‌باشد. اَعمال سطح میانه را می‌توان مواردی از قبیل حرکت به مکان خاص دانست که از دستور لایه پایین حرکت به سمت یک نقطه استفاده می‌کند یا چرخش به سمت یک شیئ خاص که از دستور سطح پایین چرخش به سمت یک نقطه استفاده می‌کند و مواردی نظیر این. در نهایت اعمال سطح بالا در تقسیم‌بندی انجام شده برای یک عامل روبوسینا شامل مجموعه‌ی اعمال دریبل، پاس مستقیم، پاس در عمق، قطع توپ و شوت به سمت دروازه می‌باشد که هر یک از آنها مجموعه‌ای از اعمال در سطح میانه و یا در صورت نیاز سطح پایین را مورد استفاده قرار می‌دهند. در مورد اعمال لایه پایین و میانی مطالب به شکل کامل و جامعی در مرجع [۵] آمده‌اند و عامل روبوسینا نیز این مجموعه از اعمال را به شکل گسترده در پیاده‌سازی اعمال سطح بالای خود به کار می‌گیرد اما در لایه بالا پیاده‌سازی‌ها مدل‌های منحصر به فردی را دنبال می‌کنند لذا در این بخش با صرف‌نظر از توابع لایه‌های اول و دوم به توضیح اَعمال در بالاترین لایه می‌پردازیم.

مجموعه اعمال سطح بالا را می‌توان با استفاده از روش‌های یادگیری به یک عامل آموخت و یا به شکل محاسباتی پیاده‌سازی کرد. در این بخش به معرفی این مجموعه از اعمال و الگوریتم‌های پیاده‌سازی آن‌ها با روش‌های محاسباتی می‌پردازیم و در فصل‌های آینده بعد از معرفی روش‌های یادگیری، مجموعه مهارت‌هایی را که الگوریتم‌های یادگیری روی آنها پیاده‌سازی شده مورد مطالعه قرار می‌دهیم.

۳—۴—۱ قطع توپ (Intercepting the Ball)

این مهارت از مهارت‌های پایه‌ای می‌باشد و عبارت است از سیستمی که هم الویت گرفتن توپ توسط بازیکنی نسبت به هم‌تیمی‌هایش را درک کند و هم عمل گرفتن توپ را انجام دهد. درک درست بازیکنی که باید توپ گیرنده‌ی اصلی باشد باعث می‌شود تا دریافت توپ سریعتر و مطمئن تر انجام شود و آرایش تیمی بازیکنان دچار آشفتگی نشود. این مهارت به طور کلی به دو قسمت تقسیم می‌شود:

۱) مهارت گرفتن توپ

۲) ابزار کمکی استراتژی برای درک تصاحب کننده توپ

مهارت گرفتن توپ

هر بازیکن برای تصمیم‌گیری نیاز دارد که بداند بازیکن‌های موثر بر روی توپ و نیز خود، توپ را در چند سیکل زمانی می‌گیرند. اطلاعات مربوط به خود بازیکن را می‌توان با دقت نسبتاً خوبی با استفاده از داده‌های موجود به‌دست آورد. اطلاعات بازیکن‌های دیگر یا از طریق مفاهیم منتقل شده توسط بازیکن مورد نظر[۱۴] و یا از طریق الگوریتم‌های محاسباتی پیمایشی بدست می‌آیند. از آنجا که به‌دست آوردن تعداد سیکل‌های گرفتن توپ برای بازیکنان دیگر به علت خطا در داده‌ها محیطی غیر ممکن است، در نتیجه، دو الگوریتم مختلف یکی برای فواصل دور و دیگری برای فواصل نزدیک مورد استفاده قرار می‌گیرند که در فواصل نزدیک پارامترهای بیشتری برای محاسبه دقیق‌تر حرکت توپ مورد استفاده قرار می‌گیرند.

---

[۱۴] communication

◇ الگوریتم تخمینی پیدا کردن سیکل‌ها(برای سایر بازیکنان):

سرعت توپ اگر نیروی جدیدی بر آن وارد نشود بصورت تصاعد حسابی شروع به کاهش می‌کند و با بررسی این وضعیت که اگر توپی در n سیکل زمانی به نقطه خاصی برسد بازیکن نیز بتواند در همان بازه‌ی زمانی یا کمتر در آن نقطه‌ی خاص باشد، n را به عنوان جواب این مسئله بر می‌گرداند.در ضمن متغیر $interceptMaxCycles$ در تمام توابع وابسته به قطع توپ برابر با مقدار ۶۰ در نظرگرفته شده است، همچنین تابع predictNrCyclesToPoint تعداد سیکل‌های رسیدن یک شیی‌ء به یک نقطه خاص را برمی‌گرداند.

method getInteseptionInfo
$i = 1$
repeat
$i = i + 1$
q_{t+i} = predictPosistionAfterNrCycles($ball$,i)
n = predictNrCyclesToPoint($Player$,q_{t+i})
until $n < i$ **or** $i > interceptMaxCycles$
end method

◇ الگوریتم دقیق پیدا کردن سیکل‌ها(برای خود بازیکن):

۱) simpleIntercept : این تابع برای مواردی است که توپ به بازیکن نزدیک است و بررسی حالت‌های مختلف گرفتن توپ سریع‌ترین راه حل را برمی‌گرداند.

۲) getInterceptionInfo : مشابه الگوریتم تخمینی پیدا کردن سیکل‌هااست بااین تفاوت که چون بازیکن دادهایی با دقت بالا نسبت به خود دارد جزئیات بیشتری در این تابع بررسی می شود.

همان‌طور که ذکر شد getInterceptionInfo مشابه الگوریتم تخمینی پیدا کردن سیکل‌هاست و بر همین اساس فقط به توضیح simpleIntecept بسنده می‌کنیم.

simpleIntercept :
اگر گرفتن توپ بوسیله‌ی بازیکن حداکثر در ۵ سیکل امکان‌پذیر باشد این تابع می تواند جوابی برای انجام این عمل داشته باشد. این تابع دارای سه زیر تابع اصلی onlyForwardIntercept و onlyBackwardIntercept و turnDashIntercept می‌باشد. تابع onlyForwardIntercept شرایطی را که بازیکن حرکت رو به جلو و بدون چرخش برای گرفتن توپ انجام می دهد را بررسی می کند، تابع onlyBackwardIntercept هم شرایطی حرکت به عقب و بدون چرخش را بررسی می کند، ولی تابع turnDashIntercept این امکان را که بازیکن بتواند بچرخد و بعد از آن حرکت رو به جلو و یا رو به عقب برای گرفتن توپ داشته باشد را بررسی می کند.در این بین این نکته باید ذکر شود که حرکت رو به عقب به علت مصرف بالای stamina چندان مطلوب نیست ولی به علت حساسیت این مهارت بررسی آن ضروری است.

استراتژی، ابزار کمکی برای درک تصاحب کننده‌ی توپ

وضعیت توپ در زمین مشخص می‌کند که یک تیم باید حمله کند و یا دفاع کند و اگر بتوان این وضعیت را درست درک کرد می‌توان استراتژی درست دفاعی یا حمله‌ای که معمولاً با هم خیلی متفاوتند را انتخاب کرد. توپ در زمین به سه حالت **آزاد**، **در مالکیت حریف** و یا **در مالکیت تیم خودی** می‌باشد. در دو وضعیت که توپ مالک دارد، اگر داده‌های کافی وجود نداشته باشد، درک صاحب توپ کار سختی است ولی اما اگر توپ آزاد باشد درک این احتمال که توپ در نهایت در تصاحب چه کسی خواهد بود وضعیت توپ را در آینده مشخص می‌کند. و این درک را می توان یااز طریق اطلاعات گفته شده توسط بازیکنان نزدیک توپ (communication) و یا بررسی وضعیت گرفتن توپ برای تمام بازیکنان مؤثر بر توپ به دست آورد.

برای حل این مسئله حافظه‌ای در نظر گرفته شده که وضعیت گرفتن توپ برای تمام بازیکنان در آن ذخیره می‌شود. داده‌های این حافظه به دو صورت تحلیل اطلاعاتی که بازیکن می‌بیند و یا از طریق اطلاعاتی که خود بازیکنان انتقال می‌دهند (communication) به دست می آید. فرض می‌کنیم در حرکت توپ تحولی ایجاد نشود (بازیکنی تغییری در حرکت توپ ایجاد نکند)، در این شرایط داده‌ی مبین زمان تصاحب توپ که از طریق ارتباط با بازیکنان دیگر به دست می آید در سیکل بعد واحد از تعداد سیکل‌های دریافتیِ آن بازیکن کم شده و به عنوان داده‌ی فعلی ذخیره می‌شود. و اما بازیکنانی که داده‌ای انتقال نداده‌اند، اگر بازیکن دیده شود داده‌های آن‌ها به روز می شود ولی اگر دیده نشود از تعداد سیکل‌های آن بازیکن یک واحد کم می شود و داده‌هایی که از این طریق به روز می‌شوند در شرایطی اعتبار خود را حفظ می‌کنند که از آخرین باری که دیده شده‌اند حداکثر ۵ سیکل زمانی گذشته باشد. چنانچه در حرکت توپ تحولی ایجاد شد، کل اطلاعات فاقد اعتبار است و کل اعمال باید از سر گرفته شود.

۳–۴–۲ دریبل (Dribbling)

دریبل از جمله مهارت‌هایی است که وابسته به تصمیم‌گیری فردی می‌باشد. در این مهارت انتظار می‌رود که بازیکن توپ را به یک نقطه‌ی خاص انتقال دهد یا با توپ در جهت خاصی حرکت کند و چنانچه بازیکن مدافع در برابر آن قرار گیرد از او عبور کند. در ضمن این مهارت باید قابلیت عمل با سرعت‌های مختلف را دارا باشد. در شرایط عادی که بازیکن مدافعی وجود ندارد این مهارت عمل ساده حرکت به نقطه‌ی خاص را انجام می‌دهد. اما در صورت وجود مدافع، عمل اصلی این مهارت یعنی جا گذاشتن بازیکن فعال می‌شود. این مهارت به سه قسمت تقسیم شده:

۱) حرکت با توپ

۲) فریب دادن بازیکن

۳) حفظ توپ

و اگر این مهارت قادر به انجام اعمال فوق نباشد به تابع فراخوانی کننده جواب غیرممکن بودن انجام این مهارت را برمی‌گرداند.

method dribble

```
    if run with ball is possible then
        runWithBall( )
    else if palyer is misleadable then
        mislead( )
    else
        holdBall( )
    end if
end method
```

حرکت با توپ

در حرکت با توپ سعی در ضربه‌زدن به توپ به شکلی می‌شود که بازیکن بتواند زمان بیشتری بدود و در نهایت توپ را در اختیار بگیرد. حرکت با توپ را می‌توان به دو حالت تقسیم بندی کرد.

۱) حالتی که توپ در $kickable\_area$ بازیکن بماند.

۲) حالتی که توپ بعد از چند سیکل زمانی دوباره به $kickable\_area$ بازیکن برگردد. در این صورت تقسیم‌بندی کمتر از ۳، بیشتر از ۳ و مساوی با ۳ سیکل بر اساس اینکه توپ در چه زمانی به $kickable\_area$ بازیکن وارد می‌شود صورت می‌گیرد که سرعت‌های حرکت مختلفی را نتیجه می‌دهد. البته این نکته قابل ذکر است که بازه‌ی انتهایی این زمان‌بندی نسبت به نوع بازیکن و شرایط آن قابل تغییر است.

برای موفقیت در یک ضربه‌ی مناسب این مهارت نیاز به بررسی بازیکنان مؤثر بر توپ دارد که با محاسبه‌ی ضربه‌های مختلف به توپ و احتمال دریافت آن توسط بازیکنان مدافع و مؤثر، تعداد سیکل زمانی مناسب برای حرکت با توپ توسط بازیکن به دست می‌آید. نکته دیگر حفظ یک تصمیم و ایجاد انسجام منطقی برای یک تصمیم و تداوم آن در زمان‌های آینده است به شکلی که با بررسی درست مانع ایجاد خطا و احتمالاً از دست رفتن توپ گردد. برای حل این مسئله حافظه‌ای در نظر گرفته می‌شود که در زمان‌های آینده اگر شرایط تصمیم‌گیری صادق بود این تصمیم تداوم می‌یابد و در غیر این صورت تصمیم جدید گرفته می‌شود.

یکی از مشکلات که معمولاً در شرایط حرکت با توپ ایجاد می‌شود اختلاط مکانی بازیکن و توپ[۱۵] است که باعث ایجاد خطای زیادی در وضعیت توپ و به احتمال زیاد باعث ایجاد یک تصمیم غلط در حرکت می‌شود و باید سعی شود تا این حالت را از سیستم حذف کرد. در این شرایط چاره‌ای جز حذف حالت‌هایی که منجر به تصادم می شوند و نیز بررسی همیشگی برای امکان وجود تصادم نیست. با این حال در بعضی از حالت‌ها گریز از این وضعیت امکان ندارد و باید شرایط را برای یک شروع جدید آماده کرد. قاعدتاً باید بهینه‌ترین حالت ممکن برای شروع یک حرکت پا به توپ جدید انتخاب شود. بعد از بررسی‌های مختلف این نتیجه به دست آمد که زاویه ۱۳۵ درجه نسبت به بدن بازیکن و با انتخاب علامت $\pm$ نسبت به مکان فعلی توپ مناسب‌ترین حالت برای یک شروع خوب حرکت با توپ است.

[۱۵] Collision

```
method runWithBall
    previousDecision = get dribble decision from memory
    if isSafe( previousDecision ) then
        return true
    end if
    cycle = find best cycle for run with ball
    if isCollision( nextBallSituation, myNextPosition ) then
        cycle = bestRunWithBallPosition( )
    else if cycle ≤ 3 and ballSafeFasterNCycle( ) == false then
        return false
    end if
    initializeDribbleMemory( )
    decision = createNewDecisionRunWithBall( cycle )
    setNewDribbleMemory( decision )
end method
```

از آنجا که پیاده‌سازی الگوریتم محاسباتی که بوسیله‌ی آن بتوان بهترین ضربه‌ی ممکن به توپ را طوری به دست آورد که در نهایت هم توپ بدون انجام turn اضافی به $kickable\_area$ بازیکن برگردد، و هم بیشترین زمان ممکن برای دویدن به همراه توپ را به بازیکن بدهد، امری دشوار است؛ استفاده از یک شبکه‌ی عصبی در آموزش این مهارت سودمند و درخور توجه می‌باشد.

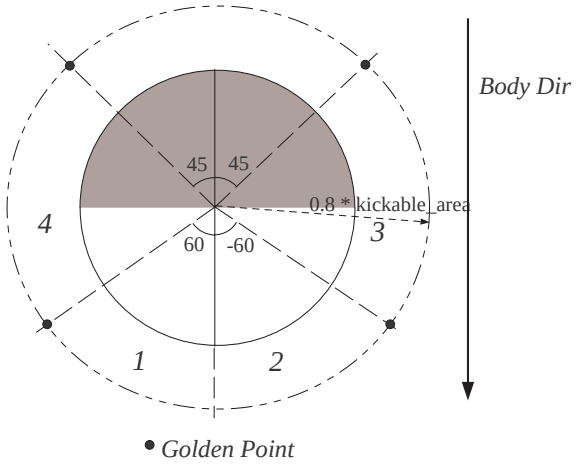
فریب دادن بازیکن

فریفتن بازیکن مهم‌ترین قسمت مهارت دریبل است و دقت در پیاده‌سازی و بررسی جزئیات بیشتر، توانایی این مهارت را بیشتر می‌کند. فریب یک بازیکن مدافع به دو صورت امکان دارد :

۱) **جا گذاشتن بازیکن به وضعیت درگیر و در نهایت فریب بازیکن.** در این حالت بازیکن به مسیر خود ادامه می‌دهد و اگر بازیکن مدافع برای ادامه مسیر مزاحمت ایجاد کرد به وسیله‌ی استراتژی فریب سعی در فریب دادن بازیکن می‌کند.

۲) **جا گذاشتن بازیکن به صورت غیر درگیر.** انتخاب مسیری برای حرکت که بازیکن بدین طریق هم با بازیکن حریف درگیر نشود و هم با ایجاد خطا در عملکرد مدافع باعث جاماندن مدافع شود.

در حالت اول سرعت بالای دریبل و افزایش قدرت پیشروی به سمت دروازه حریف بالاتر است و در ضمن در صورت موفقیت به فریب بازیکن، فاصله‌ی زیادی بین مدافع و بازیکن مهاجم ایجاد می‌شود. در حالی که در حالت دوم گریز از عدم درگیری کار سختی است و انتخاب مسیری که باعث اشتباه مدافع شود برای تیم‌های مختلف به علت روش های مختلف در Block توپ دشوار است. در ضمن از سرعت پایین‌تری در اجرا برخوردار است، ولی در صورت پیاده‌سازی می‌تواند سودمند باشد. از آنجا که تیم روبوسینا یک تیم سرعتی است و همچنین مطالعه‌ی خوبی بر روی رفتار عمومی بازیکنان هنگام درگیری و تصاحب توپ



شکل ۳–۶: نقاط مهم برای Dribbler

انجام شده بود، حالت اول پیاده‌سازی شد.

مهم‌ترین نکاتی که باعث می‌شود بازیکن مدافع دچار اشتباه شود عبارتند از :

- پیاده‌سازی ضعیف مهارت بلاک که معمولاً تیم‌ها در پیاده‌سازی مهارت بلاک به طور اشتباه فقط به مکان توپ حساس هستند و به وضعیت توپ نسبت به بازیکن حریف اهمیتی نمی‌دهند و به طور واضح‌تر با این مهارت مانند گرفتن توپ (Intercept) برخورد می‌کنند.
- از آنجا که به طور معمول بازیکنان برای گرفتن توپ به طور رو در رو با بازیکنان حریف درگیر می‌شوند و معمولاً دارای سرعت بالایی هم هستند اگر اشتباهی در عمل بلاک انجام دهند حداقل ۳ سیکل از بازیکن حامل توپ عقب می‌مانند و این مسئله در بازیکنان نامتشابه شدیدتر هم هست.

برای بررسی بیشتر نیازمندیم که فضای بازیکن را به مرکزیت بدن بازیکن به صورت شکل ۳–۶ تقسیم‌بندی کنیم و نقاط طلایی را در هر یک از این فضاها و به اندازه‌ی $۰/۸ \times kickable\_area$ و با زاویه‌های تعیین شده در شکل تعیین نماییم.

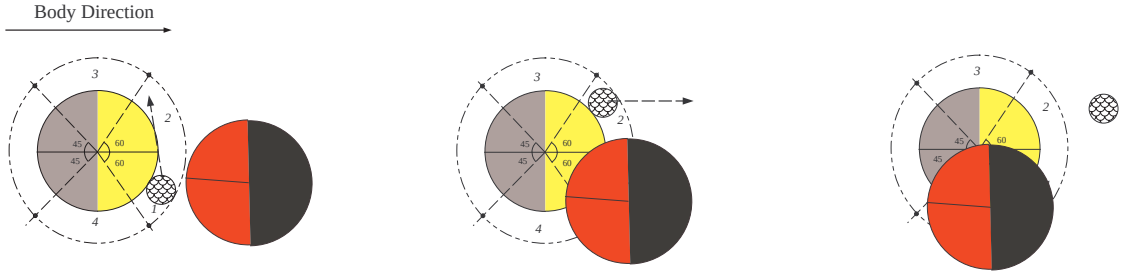
فریب دادن اصلی‌ترین فعالیت این مهارت است و بر یکی از اصول زیر استوار است :

◇ تغییر مکان توپ در شرایطی که بازیکن مدافع در قسمت جلوی بازیکن مهاجم قرار دارد باعث اشتباه بازیکن مدافع می‌شود. به این ترتیب که اگر بازیکن مدافع در یکی از وضعیت های ۲ یا ۳ باشد و فاصله‌ای بیشتراز ۱/۷ از بازیکن حامل توپ داشته باشد بنا بر الگوریتم زیر عمل می شود.

```
method mislead1
    if opponentState does not equal 1 or 2 then
        return false
```

```
        end if
        if opponentState equals 1 then
                firstState = 2
                secondState = 4
        else
                firstState = 1
                secondState = 3
        end if
        if canSendBallTo( firstState ) and isSafeInNextCycle( firstState ) then
                decision = createNewDecisionMislead( firstState, misleadType1 )
        else if canSendBallTo (firstState) and isSafeInNextCycle (firstState) then
                decision = createNewDecisionMislead( secoendState, misleadType1 )
        else
                return false
        end if
        initializeDribbleMemory( )
        setNewDribbleMemory( decision )
end method
```

در شکل ۳—۷ یک نمونه از عملیات مرحله به مرحله‌ی این مهارت مشخص شده است. این نوع فریب دادن همانند دریبل یک و دو در فوتبال سالنی عمل می‌کند و سریع‌ترین و موفق‌ترین نوع فریب است که به طور خاص در هنگام حرکت عرضی قدرت عملکرد خود را بیشتر نشان می‌دهد.

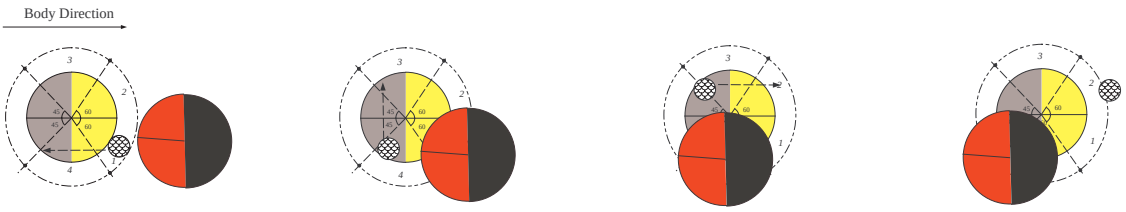


شکل ۳—۷: یک نمونه از دریبل **یک — دو** موفق

◇ توپ به جایی انتقال پیدا کند که بازیکن مدافع فرض کند که در سیکل بعد می‌تواند توپ را در $kickable\_area$ خود داشته باشد. بازیکن حامل توپ با انجام این عمل در طی یک یا دو مرحله می‌تواند بازیکن حریف را به سمت پشت خود بکشاند تا برای اجرای مرحله‌ی نهایی فرصت را به دست آورد و بازیکن مدافع کاملاً از جریان جا بماند و زمان بیشتری برای بازیکن حامل توپ جهت فرار و شروع یک حرکت با توپ موفق ایجاد کند.

```
method mislead2
    if opponentState equals 1 then
        firstState = 4
        secondState = 3
    else if opponentState equals 2 then
        firstState = 3
        secondState = 4
    else if opponentState equals 3 then
        firstState = 2
        secondState = 4
    else if opponentState equals 4 then
        firstState = 1
        secondState = 3
    end if
    if canSendBallTo( firstState ) and isSafeInNextCycle( firstState ) then
        decision = createNewDecisionMislead( firstState, misleadType2 )
    else if canSendBallTo( firstState ) and isSafeInNextCycle( firstState ) then
        decision = createNewDeciosnMislead( secondState, misleadType2 )
    else
        return false
    end if
    initializeDribbleMemory( )
    setNewDribbleMemory( decision )
end method
```

در شکل ۳—۸ یک نمونه از عملیات مرحله به مرحله‌ی این مهارت مشخص شده است.



شکل ۳—۸: یک نمونه از دریبل بلند موفق

بعد از فریب بازیکن توپ در وضعیتی قرار می‌گیرد که برای یک حرکت پا به توپ خوب شرایط مناسبی دارد. در نهایت الگوریتم نهایی فریب بازیکن را می‌توان به صورت زیر نوشت :

```
method mislead
    if mislead1( ) = true then
```

```
            return true
        end if
        return mislead2( )
end method
```

حفظ توپ

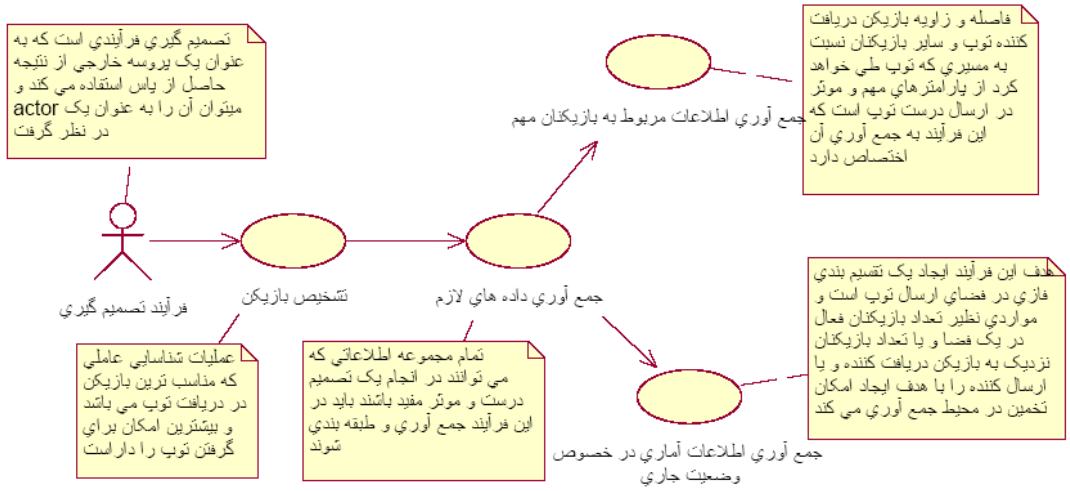
در این قسمت انتظاری برای انتقال توپ به نقطه‌ی خاص نیست بلکه بازیکن باید توپ را در محلی که قرار دارد حفظ کند تا شرایط بهتری برای انجام یکی از مهارت‌های خود پیدا کند. ساده‌ترین حالت پیاده‌سازی این قسمت به صورتی است که توپ را در وضعیتی قرار دهیم که توپ بر روی خط گذرنده ازمرکز بدن بازیکن مدافع و بازیکن مهاجم باشد به طریقی که بدن مهاجم بین توپ و مدافع باشد.(شکل ؟؟).

۳—۴—۳ پاس مستقیم (Direct Passing)

پاس از دسته اعمالی است که تنها به یک تصمیم‌گیری فردی وابسته نیست بلکه هم عامل فرستنده و هم عامل گیرنده باید درک درست از وضعیت داشته باشند. بازیکن فرستنده‌ی توپ باید تضمین کند که توپ مسیر را بدون هیچ تهدیدی از سوی بازیکنان حریف طی می‌کند و به بازیکن مقابل می‌رسد از سوی دیگر بازیکن دریافت کننده باید در درجه اول فهم درست از وضعیتی که به واسطه ارسال توپ در آن قرار گرفته داشته باشد و از سوی دیگر مهارت دریافت توپ را داشته باشد. توجه به این نکته ضروری است که در مهارت پاس با توجه به وجود خطا در محیط رسیدن توپ به یک نقطه خاص هدف نیست، هدف اصلی حرکت توپ در جهتی است که بازیکن دریافت کننده امکان دریافت آن در بهترین موقعیت و با کمترین امکان تصاحب از سوی حریف را داراست.

عملیات پاس به این شکل انجام می‌شود که یک سری شرایط برای انتخاب بازیکن مناسب بررسی می‌شوند و پس از یافتن تعدادی از بازیکنان خودی که این شرایط را دارا هستند امکان دریافت توپ توسط هر یک از آنان بررسی می‌شود و بازیکنی که مناسب‌ترین وضعیت در دریافت توپ را داراست در نهایت به عنوان گیرنده توپ فرض می‌شود و با استفاده از دستور لایه‌ی پایین‌تر ارسال توپ به یک نقطه خاص، توپ به سمت او فرستاده می‌شود.

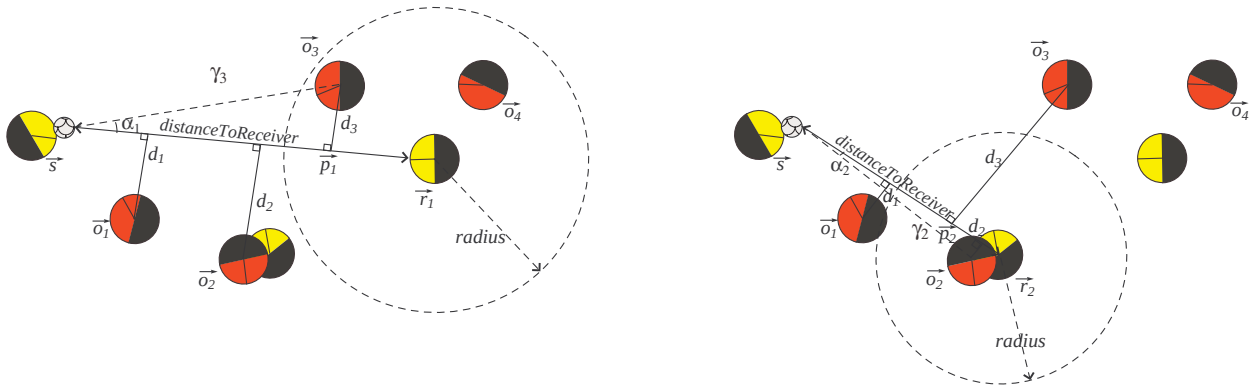
در مقایسه وضعیت جاگیری بازیکنان شرایط بازیکنان اطراف که می‌توانند خودی یا غیر خودی باشند مهم است. فاصله و زاویه و همین‌طور تعداد بازیکنان اطراف بازیکن دریافت کننده می‌توانند تعیین کننده باشند. شکل ۳—۹ مدل use case برای عمل پاس را نشان می‌دهد.با توجه به شرایطی که باید پیش از ارسال توپ به بازیکن دیگری بررسی شوند مناسب‌ترین راه استفاده از یک روش یادگیری است که در بخش ۵—۲ بدان خواهیم پرداخت. در این مرحله مفاهیم اولیه برای ارسال یک پاس درست و سالم را به همراه یک روش ابتدایی مورد مطالعه قرار می‌دهیم. همان طور که در شکل ۳—۹ نشان داده شده است عمل پاس از دیدگاه ارسال کننده آن باید با جمع آوری اطلاعات در خصوص مسیری که توپ می‌بایست طی کند همراه باشد. پس از جمع کردن اطلاعات کافی مناسب‌ترین مسیر می‌تواند به عنوان مسیری که توپ باید به سمت آن پرتاب شود انتخاب شود. ساده‌ترین الگوریتم در این حالت می‌تواند به شکل زیر تعریف شود:



شکل ۳—۹: مدل use case برای عمل پاس

method directPass
  foreach receiver $\vec{r_i}$ who satisfies decision conditions **do**
    create $passLine_i$ from sender $\vec{s}$ to $\vec{r_i}$
    $\overrightarrow{o_{close}}$ = closest opponent player $\vec{o_i}$ to $passLine_i$
    $\vec{p_i}$ = intersection point of $passLine_i$ with its perpendicular line drawn from $\overrightarrow{o_{close}}$
    if `predictNrCyclesToPoint` ($\overrightarrow{o_{close}}$, $\vec{p_i}$) < `predictNrCyclesToPoint` (`BALL`, $\vec{p_i}$) **then**
      return
    end if
    α_i = $passLine_i$ angle relative to $\overrightarrow{o_{close}}$
    λ_i = Number of players around r_i and within $\pi(init\_ball\_speed, distanceToReceiver)$
  end foreach
  α_{max} = find max α_i
  foreach r_i **do**
    α_i = normalize α_i with α_{max}
    $pass\_rate_i = \alpha_i/(\lambda_i + ۱)$
  end foreach
  $\overrightarrow{r_{max}}$ = $\vec{r_i}$ with maximum $pass\_rate_i$
end method

ابتدا باید رسیدن توپ به r_i پیش از قطع آن توسط بازیکن حریف تضمین شود که برای این منظور از الگوریتم قطع توپ استفاده شد. اگر تعداد سیکل‌هایی که طول می‌کشید تا نزدیک‌ترین بازیکن حریف یعنی $\vec{p}$ روی نقطه $\overrightarrow{o_{close}}$ بر روی خط ارسال پاس قرار بگیرد کمتر از تعداد سیکل‌های عبور توپ از نقطه $\vec{p}$ باشد، بازیکن واقع در $\vec{r_i}$ برای دریافت پاس مناسب نیست. در ادامه بهترین بازیکن برای دریافت پاس مشخص می‌شود. در الگوریتم بالا π تابعی است که فضای اطراف یک بازیکن را برای حضور بازیکنان مؤثر



(الف) (ب)

شکل ۳—۱۰: نمایشی از پیاده‌سازی مدل محاسباتی الگوریتم پاس

در قطع توپ بررسی می‌کند. عملکرد π به این شکل است که با توجه به سرعتی که در هنگام ضربه به توپ، توپ پیدا می‌کند یعنی $init\_ball\_speed$ و فاصله بازیکن دریافت کننده تا بازیکن ارسال کننده توپ یعنی $distanceToReceiver$ تعداد سیکل‌هایی از بازی را که طول می‌کشد تا توپ به بازیکن برسد محاسبه می‌کند و با فرض اینکه بازیکن حریف می‌تواند در بدترین حالت در هر سیکل مقدار $player\_speed\_max$ یعنی ۱ متر را طی کند دایره‌ای به شعاع **تعداد سیکل‌ها** $\times$ ۱m را در اطراف بازیکن برای حضور سایر بازیکنان بررسی می‌کند.برای محاسبه تعداد سیکل‌ها می‌توان از روابط زیر استفاده کرد :

$$distanceToReceiver = init\_ball\_speed \cdot \frac{۱ - \boldsymbol{ball\_decay}^{cycles}}{۱ - \boldsymbol{ball\_decay}} \Rightarrow$$

$$cycles = \left\lceil \log_{ball\_decay} \left(۱ - \frac{distanceToReceiver \cdot (۱ - \boldsymbol{ball\_decay})}{init\_ball\_speed} \right) \right\rceil + ۱$$

$$radius = \boldsymbol{player\_speed\_max} \times cycles$$

در روابط بالا می‌توان تعداد $cycles$ را با یک عمل پیمایش ساده و بدون نیاز به محاسبه رابطه لگاریتمی نیز به دست آورد.

شکل ۳—۱۰ نمونه‌ای از پیاده‌سازی الگوریتم فوق را نشان می‌دهد. این شکل الگوریتم تصمیم‌گیری برای بازیکن فرستنده در مکان $\vec{s}$ را در یک سیکل و هنگام بررسی دو بازیکن گیرنده که به ترتیب در مکان‌های $\vec{r_1}$ و $\vec{r_2}$ قرار دارند نشان می‌دهد. d_i فاصله بازیکن حریف واقع در $\vec{o_i}$ تا خط ارسال پاس از $\vec{s}$ به $\vec{r_i}$ و γ_i فاصله $\vec{o_i}$، با کمترین مقدار d_i، تا توپ می‌باشد. $\vec{p_i}$ نیز در هر شکل نقطه برخورد خط عمود از $\vec{o_{close}}$ بر خط ایجاد شده توسط دو بردار $\vec{s}$ و $\vec{r_i}$ یعنی همان مسیر حرکت توپ می‌باشد. برای امکان اعمال الگوریتم پاس فرض می‌کنیم که `predictNrCyclesToPoint` ($\vec{o_{close}}$, $\vec{p_i}$) در هر شکل

از predictNrCyclesToPoint (BALL, $\vec{p_i}$) در آن شکل بزرگتر یا با آن مساوی است. با اعمال این فرضیات در شکل ۳—۱۰—(الف) داریم:

$$d_۳ < d_۲ < d_۱ \Rightarrow \alpha_۱ = \arcsin\left(\frac{d_۳}{\gamma_۳}\right), \qquad \lambda_۱ = ۲$$

به صورت مشابه برای شکل ۳—۱۰—(ب) نیز داریم:

$$d_۲ < d_۱ < d_۳ \Rightarrow \alpha_۲ = \arcsin\left(\frac{d_۲}{\gamma_۲}\right), \qquad \lambda_۲ = ۱$$

از آنجا که $pass\_rate_۱ \gg pass\_rate_۲$ لذا بازیکن $\vec{r_۱}$ برای دریافت توپ شرایط بهتری را داراست. این الگوریتم حساسیت بالایی به تعداد بازیکنان اطراف گیرنده دارد و مقدار $pass\_rate$ به صورت چشمگیری با افزایش این تعداد کم می‌شود.

الگوریتم پاسی که بدان اشاره شد به دلیل ساختار بسیار ساده‌ای که از آن پیروی می‌کند دارای قابلیت اطمینان چندان بالایی نیست و احتمال تصاحب توپ توسط حریف در آن زیاد است اما می‌تواند به عنوان یک الگوی اولیه در عمل پاس مورد استفاده قرار بگیرد چنان که در تیم روبوسینا در مسابقات آزاد آمریکا در سال ۲۰۰۳ مورد استفاده قرار گرفت. این الگوریتم را می‌توان با توجه به خاصیتی که دارد برای دور کردن توپ (clear)، با قرار دادن حد پایین مشخص روی α و حذف اثر λ مورد استفاده قرار داد.

۳—۴—۴ پاس در عمق (Through Passing)

پاس در عمق مدل پیشرفته‌تری از پیاده‌سازی پاس است که در آن توپ مستقیماً به بازیکن خودی فرستاده نمی‌شود بلکه در یک فضای باز، بین بازیکنان حریف، و در شرایطی که احتمال دریافت توپ توسط هیچ یک از بازیکنان حریف نیست فرستاده می‌شود و بازیکن گیرنده با دریافت توپ می‌تواند یک موقعیت مناسب را به دست آورد. در ارسال یک پاس در عمق ایجاد درک درست در بازیکن گیرنده بسیار مهم‌تر از ارسال توپ در فضا است چرا که اگر بازیکن گیرنده به هر دلیل به سمت توپ حرکت نکند توپ و موقعیت به سادگی از دست می‌روند. اهمیت این نکته زمانی روشن‌تر می‌شود که درمی‌یابیم بازیکن ارسال کننده باید مکانی را که گیرنده می‌تواند توپ را در آن قطع کند محاسبه کند و اگر هریک از دو طرف در ارسال و یا حرکت حتی یک سیکل تعلل کنند محاسبات دچار اختلال می‌شود.

مسئله پاس در عمق را می‌توان به دو زیر مسئله تقسیم کرد:

۱) ارسال توپ به یک نقطه خاص $\vec{t}$ که دریافت توپ تا قبل از رسیدن به آن نقطه توسط هیچ یک از بازیکنان حریف و یا حتی بازیکنان خودی ممکن نیست.

۲) تضمین رسیدن بازیکن گیرنده $\vec{r_i}$ به نقطه $\vec{t}$ پیش از هر بازیکن دیگری و اطمینان از قطع توپ توسط این بازیکن گیرنده.

برای حل قسمت اول می‌توان به سادگی از روش مطرح شده در بخش ۳—۴—۳ استفاده کرد. برای این منظور کافی است در الگوریتم متغیری را که مکان یک بازیکن خودی را به عنوان بازیکن دریافت کننده‌ی توپ ذخیره می‌کند یعنی $\vec{r_i}$ را با نقطه‌ای که باید توپ بدان ارسال شود، $\vec{t}$، جایگزین کرد. پس از آن روند ادامه‌ی الگوریتم مشابه قبل خواهد بود. توجه به این نکته اهمیت بسیاری دارد که مشابه پاس مستقیم که در

آن انتخاب بازیکنان مناسب برای دریافت پاس در لایه تصمیم‌گیری انجام می‌شد در این بخش نیز انتخاب نقاطی که می‌توانند به عنوان هدف پاس در عمق انتخاب شوند در لایه تصمیم‌گیری انجام می‌شود. تعداد این نقاط می‌توانند بسته به شرایط برای هر بازیکن واقع در $\vec{r_i}$ از یکی تا بیشتر در فضای اطراف آن بازیکن باشند.

روش محاسباتی که می‌توان برای حل قسمت دوم از مسئله مورد استفاده قرار داد محاسبه‌ی زمانی است که بازیکن گیرنده باید برای رسیدن به نقطه هدف صرف کند. با محاسبه درست این زمان می‌توان سرعت توپ را به شکلی تنظیم کرد که با صرف همان مقدار از زمان به نقطه مورد انتظار برسد و در این شرایط بازیکن گیرنده قادر خواهد بود در موقع مناسب مسیر توپ را قطع کند و توپ را در اختیار بگیرد. اهمیت ایجاد درک درست بین بازیکن فرستنده و گیرنده برای محاسبه زمان ارسال و زمان حرکت در این بخش بهتر مشخص می‌شود. برای انجام این محاسبات می‌توان از فرمول‌های زیر استفاده کرد:

$$cycles = \texttt{predictNrCyclesToPoint } (\vec{r_i}, \vec{t}\ )$$

$$init\_ball\_speed = \min(\boldsymbol{ball\_speed\_max}, \sigma(\texttt{dist}(\texttt{BALL}, \vec{t}\ ), \boldsymbol{ball\_decay}, cycles))$$

$$\sigma(dist, \boldsymbol{ball\_decay}, nCycles) = dist \cdot \frac{۱ - \boldsymbol{ball\_decay}}{۱ - \boldsymbol{ball\_decay}^{nCycles}}$$

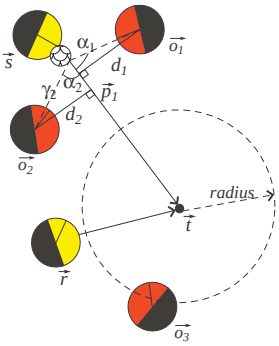
در دو رابطه بالا با استفاده از predictNrCyclesToPoint $(\vec{r_i}, \vec{t}\ )$ تعداد سیکل‌های رسیدن بازیکن $\vec{r_i}$ به نقطه $\vec{t}$ محاسبه می‌شود و تابع σ نیز سرعت اولیه‌ای که باید به توپ داده شود تا توپ به $\vec{t}$ برسد را با توجه به فاصله‌ی توپ تا $\vec{t}$ یعنی dist $(\texttt{BALL}, \vec{t}\ )$، مقدار اُفت سرعت توپ در هر سیکل یعنی $ball\_decay$ و تعداد $cycles$ محاسبه می‌کند. با توجه به این دو تابع بازیکن فرستنده می‌تواند توپ را به سمت $\vec{t}$ ارسال کند اما مسئله دیگری که مطرح می‌شود مقدار $init\_ball\_speed$ است که اگر از حد معینی پایین‌تر باشد قطع توپ را برای سایر بازیکنان ساده می‌کند در این شرایط باید تعادل در سرعت ارسال توپ و امکان دریافت آن از سوی $\vec{r_i}$ با سرعت انتها و امکان قطع توپ توسط بازیکنان دیگر برقرار باشد تا بتوان توپ را به درستی به نقطه مذکور ارسال کرد که ایجاد این تعادل چالش بزرگی است.

شکل ۳—۱۱ نمایشی از پیاده‌سازی الگوریتم پاس در عمق می‌باشد که در آن اعمال روش بخش ۳—۴—۳ بر روی نقطه هدف $\vec{t}$ نشان داده شده است. با توجه به شکل در این حالت داریم:

$$d۲ < d۱ \Rightarrow \alpha_۲ = \arcsin\left(\frac{d_۲}{\gamma_۲}\right)$$

$$\lambda = ۰$$

همان طور که در ابتدای این بخش اشاره شد، در کنار پیچیدگی محاسباتی و تحلیلی برای ارسال پاس در عمق، ارسال یک پاس در عمق به شکل درست می‌تواند در عبور کردن از فضای بسته شده توسط بازیکنان حریف و موفقیت در به ثمر رسانیدن گل بسیار سودمند باشد مجموعه اشکال ۳—۱۲ نمونه‌ای از این پیاده‌سازی را در تیم روبوسینا نشان می‌دهند.



شکل ۳—۱۱: نمایش پیاده‌سازی الگوریتم پاس در عمق

۳—۴—۵ شوت به سمت دروازه (Shooting toward Goal)

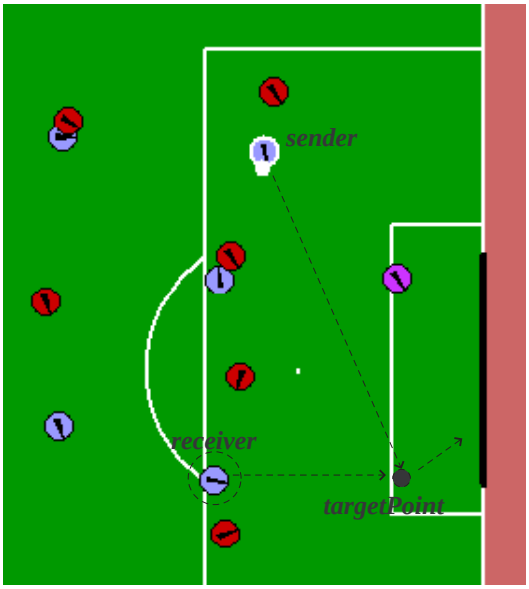
شوت به سمت دروازه آخرین عملی است که در راستای رسیدن به هدف تیمی انجام می‌شود. در این عمل بازیکن زمانی که شرایط را برای گرفتن امتیاز از تیم حریف مناسب می‌بیند به سمت دروازه شوت می‌کند و در صورت به ثمر رسیدن توپ تیم در دستیابی به هدف خود موفق شده است. شوت از مجموعه مسائلی است که به دلیل اهمیت و ویژگی‌هایی که از آن برخوردار است پیاده‌سازی‌های متعددی را توسط تیم‌های مختلف به خود اختصاص داده است. مشابه مدل‌های قبل استفاده از روش یادگیری برای شوت به دلیل گستردگی در مجموعه حالت‌ها و نیاز به درک شرایط خاص اهمیت فراوانی دارد که در بخش ۵—۱ در خصوص اِعمال یک الگوریتم یادگیر روی مسئله شوت بحث خواهیم کرد. در این بخش مشابه با بخش‌های قبل یک روش ساده و قابل پیاده‌سازی محاسباتی را مورد بررسی قرار می‌دهیم.

شکل ۳—۱۳—(الف) نمایشی از مدل محاسباتی شوت است که در آن بازیکن واقع در $\vec{s}$ قصد شلیک توپ به سمت دروازه را دارد. همانطور که در شکل نشان داده شده دروازبان در سیستم دوبعدی معمولاً به شکلی پیاده می‌شود که روی خط مشخص l و به فاصله r از دروازه حرکت کند چرا که چرخش دروازبان به دلیل سرعت بالای توپی که به سمت دروازه شلیک می‌شود می‌تواند در عمل دریافت توپ بسیار مشکل‌ساز باشد. برای یک شوت موفق باید دو فاکتور را در نظر داشته‌باشیم:

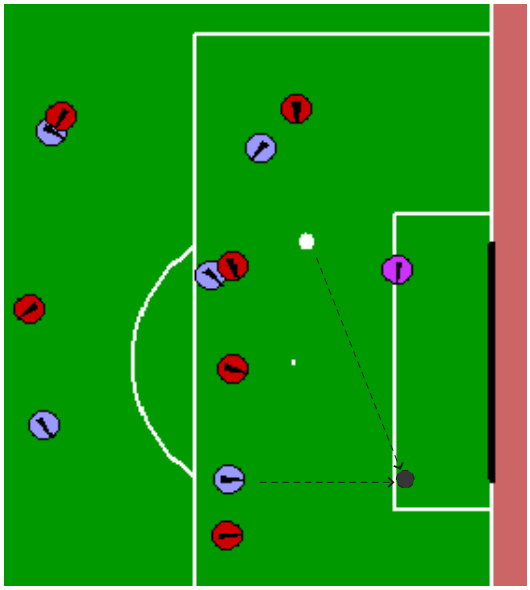
۱) ورود توپ به دروازه

۲) عدم تصاحب توپ از سوی دروازبان حریف

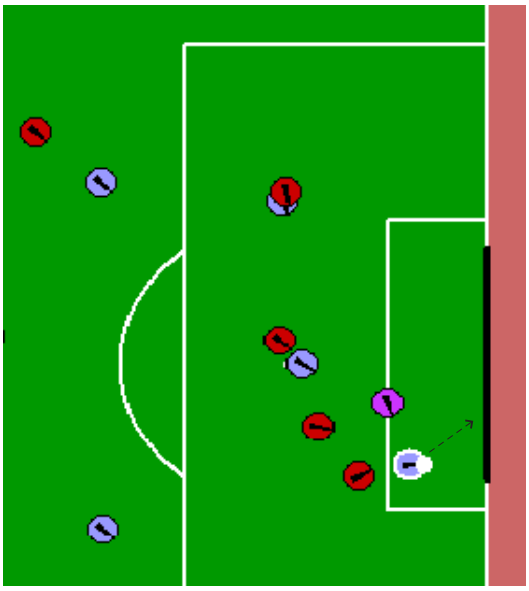
در روش محاسباتی ساده این بخش از احتمال ورود توپ به دروازه صرف‌نظر می‌کنیم و برای اطمینان از عدم خروج توپ بردارهای $\vec{p_1}$ و $\vec{p_2}$ را به فاصله نیم‌متر داخل دروازه یعنی در مختصات (۶.۵—، ۵۲.۵) و (۶.۵+، ۵۲.۵) در نظر می‌گیریم و شوت را به سمت این دو نقطه محاسبه می‌کنیم. اما در قسمت دوم از مسئله که اطمینان از عدم توانایی دروازبان حریف در تصاحب توپ است دو بردار $\vec{c_1}$ و $\vec{c_2}$ را از تقاطع خط l با خط ایجاد شده از برخورد $\vec{b}$ با $\vec{p_1}$ و $\vec{p_2}$ به دست می‌آوریم. حال، مشابه با الگوریتم پاس، چنانچه تعداد سیکل‌هایی که دروازبان برای رسیدن به نقطه $\vec{c_1}$ یا $\vec{c_2}$ صرف می‌کند به ترتیب از زمان رسیدن توپ به یکی از این دو نقطه بیشتر باشد می‌توان استنباط کرد که توپ توسط دروازبان catch نخواهد شد. با توجه به آنچه گفته شد الگوریتم زیر را داریم:



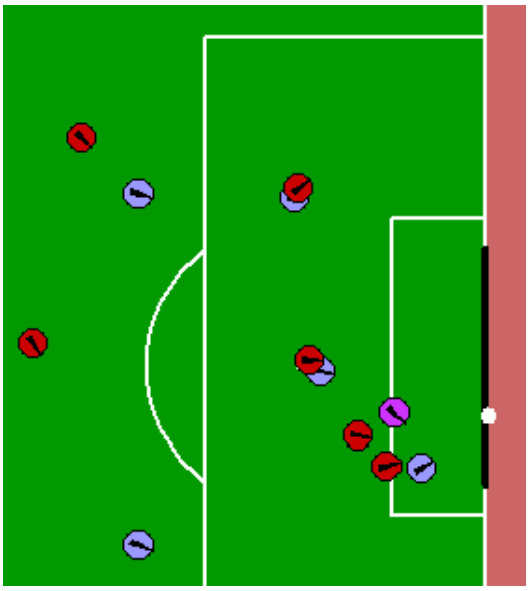
(الف)



(ب)

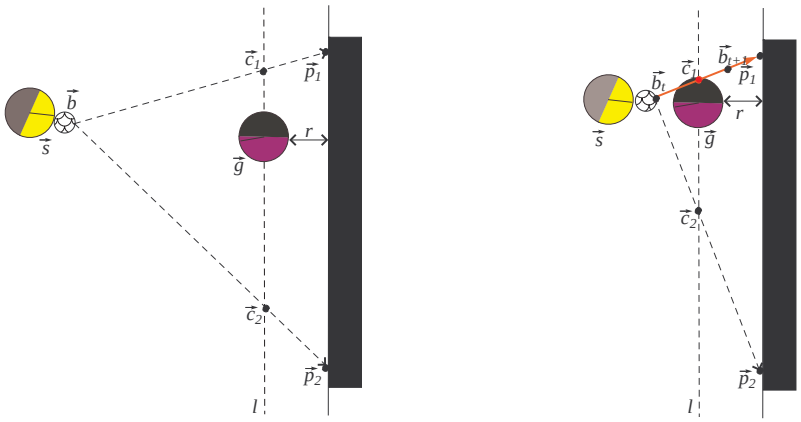


(ج)



(د)

شکل ۳–۱۲: UTUTD در سمت راست در مقابل روبوسینا در سمت چپ. بازی فینال مسابقات آزاد آمریکا در سال ۲۰۰۴، از سیکل ۱۴۴۵۳ تا سیکل ۱۴۴۶۹، گل طلایی.
(الف) بازیکن sender با درک فضا، نقطه هدف و موقعیت receiver، توپ را ارسال می‌کند.
(ب) بازیکن receiver موقعیت را درک می‌کند و به سمت نقطه هدف حرکت می‌کند.
(ج) receiver توپ را در نقطه هدف دریافت می‌کند.
(د) گُل!



(الف) (ب)

شکل ۳—۱۳: مدل محاسباتی الگوریتم شوت

method shoot
  l = vertical line which passes $\vec{g}$
  $shootLine_1$ = line from $\vec{b}$ to $\vec{p_1}$
  $shootLine_2$ = line from $\vec{b}$ to $\vec{p_2}$
  $\vec{c_1}$ = intersect ($shootLine_1$, l) and $\vec{c_2}$ = intersect ($shootLine_1$, l)
  $diffCycle_1$ = `predictNrCyclesToPoint` ($\vec{g}$, $\vec{c_1}$) - `predictNrCyclesToPoint` ($\vec{b}$, $\vec{c_1}$)
  $diffCycle_2$ = `predictNrCyclesToPoint` ($\vec{g}$, $\vec{c_2}$) - `predictNrCyclesToPoint` ($\vec{b}$, $\vec{c_2}$)
  if $\max(diffCycle_1, diffCycle_2) > \circ$ **then**
    if $\max(diffCycle_1, diffCycle_2) == diffCycle_1$ **then**
      `shootToward` ($\vec{p_1}$)
    else
      `shootToward` ($\vec{p_2}$)
    end if
  end if
end method

اما اتفاق جالب، عجیب و مهمی که می‌تواند در جریان این محاسبه رخ دهد حالتی مشابه با شکل ۳—۱۳—(ب) می‌باشد که در آن بازیکن واقع در $\vec{s}$ در فاصله نزدیکی از دروازبان قرار گرفته است. در چنین شرایطی این امکان هست که توپ را به گونه‌ای به سمت دروازه شلیک کرد تا در سیکل t در مکان $\vec{b_t}$ و بیرون از ناحیه catchable برای دروازبان تیم حریف و در زمان $t+1$ در مکان $\vec{b_{t+1}}$ و مجدداً بیرون از مکان catchable برای دروازبان باشد. همان طور که در شکل مشخص شده است، هر چند این امکان وجود دارد که مسیر حرکت توپ حتی از روی دروازبان حریف نیز بگذرد (مشابه با نقطه $\vec{c_1}$) اما با توجه به ساختار گسسته‌ای که کارگزار در محاسبه حرکت اشیاء مورد استفاده قرار می‌دهد و در بخش ۲—۲—۲ توضیح داده شد، دروازبان قادر به تصاحب توپ نیست و توپ وارد دروازه می‌شود. این خاصیت در شرایطی که دروازبان

به شدت زاویه را تنگ کرده است می‌تواند مورد استفاده قرار بگیرد [۱۶].

۳—۵ آرایش تیمی شناور(Floating Formation)

هر بازیکن در زمین پست (مسئولیت) منحصر به فردی برای خود دارد و قائدتاً نسبت به پست خود و جایگاه توپ در زمین باید در مکان مناسبی باشد تا وظیفه‌ی خود را به خوبی انجام دهد. تعیین مکان بازیکنان به وسیله‌ی سیستم **آرایش تیمی**[۱۷] انجام می‌شود. نمایش این آرایش براساس تعداد بازیکنانی که در یک خط عرضی در زمین بازی و در کنار هم قرار می‌گیرند و بدون در نظر گرفتن دروازبان معرفی می‌شود، مثلاً برای آرایش ۳—۴—۳، ۳ بازیکن در یک خط و ۴ بازیکن در خط جلوتر و در نهایت ۳ بازیکن در خط آخر و جلوتر از قبلی قرار دارند. نسبت به گرایش بازیکنان به فعالیت های دفاعی، انتقال توپ و هجومی به ترتیب به عنوان مدافع، هافبک و مهاجم تقسیم می‌شوند مثلاً برای مثال فوق ۳ مدافع و ۴ هافبک و ۳ مهاجم می توان در نظر گرفت.

تیم‌های مختلف نسبت به ایده‌هایی که در بحث استراتژی خود دارند و توانایی در اجرای مهارت‌های خاص و یا نسبت به سرعت تیمی، آرایش‌های مختلفی را در زمین بازی اجرا می‌کنند. بعد از سال ۲۰۰۴ که تیم UVA_Trilearn آرایش جدید خود را در مسابقات ارائه کرد استفاده از مدل‌های شناور که در وضعیت‌های مختلف تغییر می‌کنند و حتی بعضاً باعث تغییر پست بازیکنان می‌شوند جایگاه مهمتری به خود گرفت.

آرایش تیمی در روبوسینا

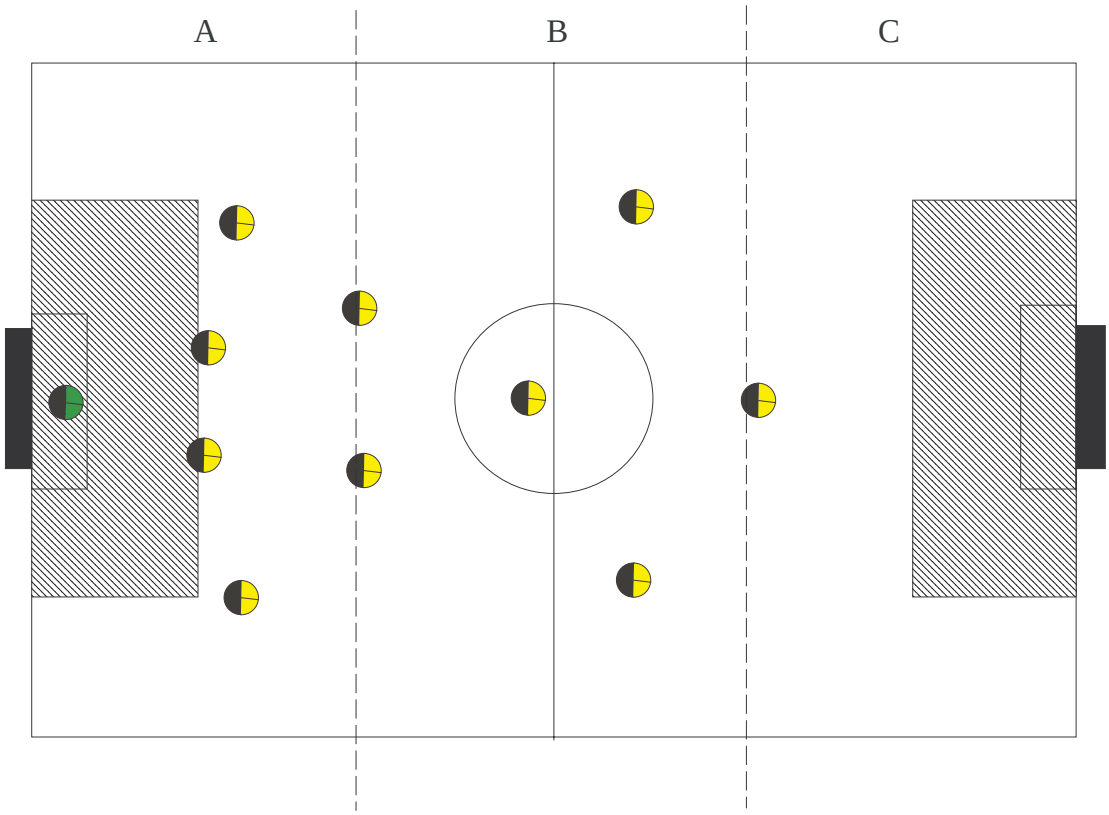
تیم روبوسینا در ابتدا از سیستم ۳—۳—۴ استفاده می‌کرد ولی این آرایش پاسخگوی انتظارات و توانایی‌های تیم نبود و در نتیجه آرایش جدیدی نیاز بود که این مسئله را حل کند.برای پیاده‌سازی مدل جدیدی که کارایی لازم را برای تیم داشته باشد قاعدتاً باید نکات زیر در نظر گرفته می شد.

- تیم روبوسینا یک تیم سرعتی است و باید همیشه چند بازیکن در فضاهای خالی وجود داشته باشند تا با گرفتن توپ‌های برگشتی از عقب زمین بتوانند یک ضد حمله را برنامه‌ریزی کنند و مهارت دریبل این تیم به عنوان یک عمل سریع، به این حرکت کمک بسیاری می کند و با فراهم آمدن این امکان توانایی‌ها این مهارت هم به نحو احسن استفاده می‌شود.
- تیم UVA_Trilearn در سال ۲۰۰۴ نشان داد که چگونه ۴ بازیکن در هنگام حمله با حمایت ۲ بازیکن هافبک می توانند عمل موفقی در هنگام تهاجم داشته باشند. چه با تجمع جلوی دروازه حریف و چه پخش شدن در میانه‌ی زمین و چه انتقال عرضی توپ.

---

[۱۶] نمونه‌ای از این پیاده‌سازی در بازی‌های دور دوم مسابقات جهانی روبوکاپ در سال ۲۰۰۵ در بازی روبوسینا در برابر BrainStormers رخ داد که تیم روبوسینا توانست با استفاده از این روش با یک گل از حریف پیشی بگیرد. این بازی در نهایت یک بر یک به پایان رسید.

[۱۷] Formation

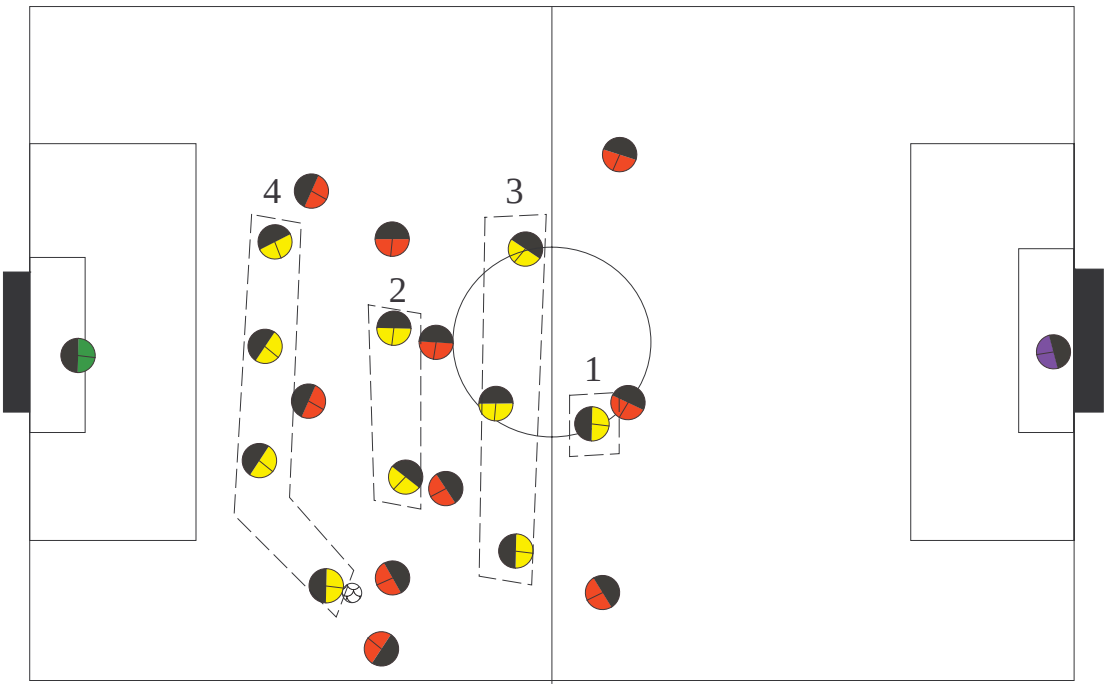


شکل ۳—۱۴: آرایش تیمی

- سیستم استراتژی دفاعی تیم روبوسینا برای ۴ بازیکن طراحی شده بود و قاعدتاً برای خط دفاعی باید ۴ بازیکن در نظر گرفته می‌شد.
- توانایی استراتژی دفاعی تیم روبوسینا نشان داد که نیاز به حمایت بیشتر از دو هافبک ندارد و اگر هافبک‌ها فضای میانی و جلوی بازیکنان مدافع را پوشش دهند نیازمندیهای دفاعی تیم برآورده می‌شود. و جالب‌تر اینکه این دو هافبک نیاز حرکتی چندانی برای کمک به کار دفاع ندارند.

آرایش تیمی روبوسینا از نگاه جزئی

آرایش پایه‌ای تیم روبوسینا ۱—۳—۲—۴ است و نسبت به شرایط مختلف به مدل‌های ۲—۲—۲—۴ و ۴—۲—۴ تبدیل می‌شود. برای توضیح بیشتر لازم است که زمین را مطابق شکل ۳—۱۴ به ۳ قسمت تقسیم کنیم . به قسمت A، $\frac{1}{4}$ دفاعی ،به B، $\frac{2}{4}$ میانی، به C، $\frac{1}{4}$ حمله و به قسمت‌های حاشور خورده محوطه‌ی جریمه گفته می‌شود.

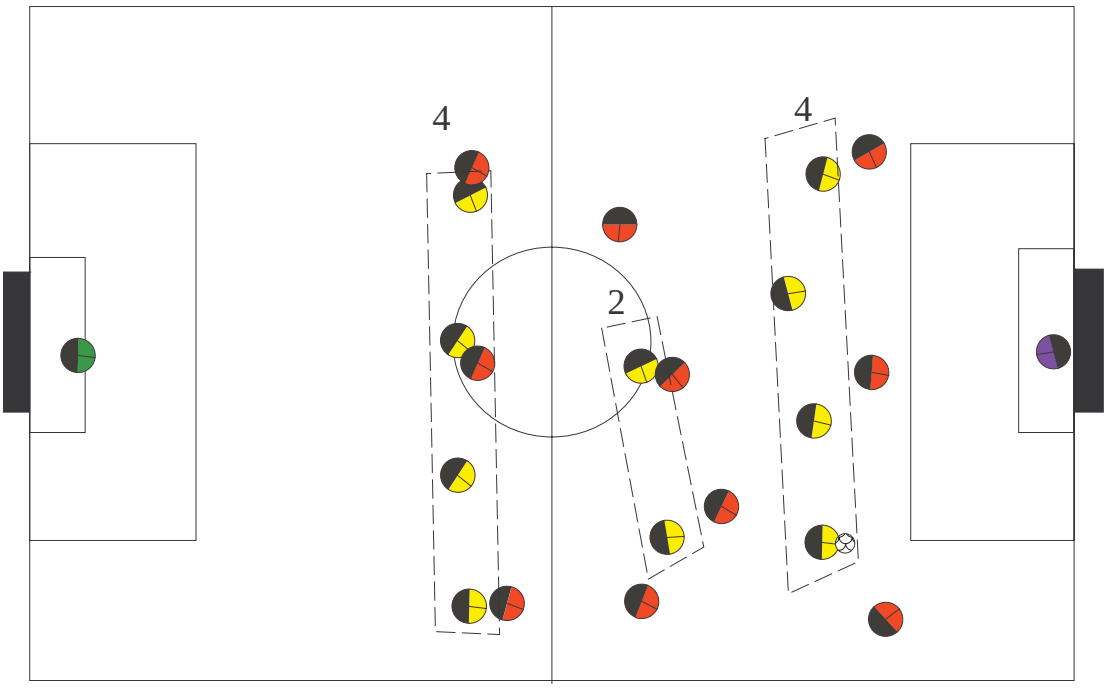


شکل ۳—۱۵: نمایشی از مدل ۱—۳—۲—۴. روبوسینادر سمت چپ

اگر توپ درقسمت ½ دفاعی باشد چه در دست بازیکن حریف و چه خودی و یا در قسمت ½ میانی باشد و توپ در دست بازیکنان مهاجم باشد. سیستم ۱—۳—۲—۴ فعال می شود که بتوان هم حمایت کاملی از خط دفاعی کرد و هم انتقال بازی راحت‌تر باشد. در این حالت بازیکن شماره ۱۰ از خط حمله جدا شده و به عنوان هافبک آزاد عمل می‌کند و بازیکنان ۸ و ۹ هم به عنوان هافبک در کنار ۱۰ والبته با اختلاف در یک خط قرار می‌گیرند (شکل ۳—۱۵).

اگر توپ در ½ میانی در دست بازیکن خودی باشد و یا در ½ حمله دست تیم حریف سیستم ۲—۲—۴ فعال می‌شود. در این حالت بازیکن شماره ۱۰ از خط هافبک جدا شده و به عنوان هافبک آزاد عمل می‌کند و بازیکنان ۸ و ۹ هم به عنوان هافبک عقب‌تر از بازیکنان ۱۰ و ۱۱ قرار می گیرند. این حالت امکان درگیری سریعتر بازیکنان ۱۰ و ۱۱ را با بازیکنان مدافع میانی حریف ایجاد می‌کند که در صورت موفقیت در رد کردن توپ از این بازیکنان به طرق مختلف امکان حصول یک گل بسیار است ولی اگر موفق به این عمل نشد با انتقال توپ به بازیکنان ۸ و ۹ که در فضای بازی از زمین قرار دارند می توان یک حمله خوب و سریع از کناره زمین طرح ریزی کرد.

اگر توپ در ½ حمله در دست بازیکنان خودی باشد سیستم ۴—۲—۴ فعال می‌شود. در این حالت بازیکنان ۸ و ۹ به جمع مهاجمان اضافه می‌شوند و هافبک‌های ۶ و ۷ فاصله خود را با مهاجمان کمتر می کنند تا بتوانند حمایت کاملی از بازیکنان مهاجم انجام دهند. تجربه نشان داده است که وقتی توپ در محوطه جریمه حریف قرار دارد تجمع بازیکنان در محوطه جریمه امری مفید است و بدین ترتیب سعی می‌شود که هر چه توپ به دروازه حریف نزدیکتر شد تجمع و فشردگی مهاجمان بیشتر شود (شکل ۳—۱۶).



شکل ۳–۱۶: نمایشی از مدل ۴–۲–۴. روبوسینادر سمت چپ

نکته‌ای که در نهایت باید ذکر کرد این است که اگر وضعیت تصاحب توپ عوض شود اختلاف زیادی در مکان بازیکنان بین وضعیت قبل و جدید ایجاد می‌شود که برای رفع این مشکل وضعیت بازیکنان توسط تابع اصلاح کننده‌ای تعیین می شود و در همه حال این تابع مکان برگشتی توسط هر مدل آرایشی را بر اساس شرایط بازی فیلتر می‌کند.

۳–۶ بازنگری فصل

در این فصل از رساله تعدادی از ویژگی‌های خاص تیم روبوسینا به همراه پاره‌ای از الگوریتم‌های محاسباتی معرفی شد. در بخش اول از این فصل معماری یک عامل به همراه نمودارهای مبین ترتیب زمانی در عملکرد بخش‌های مختلف یک روبات شبیه‌سازی شده توضیح داده شدند. دیاگرام زمانی فعالیت فرآیندها به همراه معماری کلی سیستم و نحوه‌ی قرار دادن ماجول‌های مختلف در کنار هم در این بخش بررسی شدند. بخش دوم از این فصل به معرفی هماهنگ شدن زمانی عامل با سیستم پرداخت. همان‌طور که اشاره شد محیط شبیه‌سازی یک محیط با توانایی عمل به صورت زمان‌حقیقی می‌باشد و عامل در آن باید قادر باشد که با انجام یک تقسیم زمانی درست مجموعه‌ی اعمال درک محیط، پردازش داده و تصمیم‌گیری، و عمل بر اساس تصمیم اتخاذ شده را انجام دهد. استفاده از روش‌های درست هماهنگ شدن با کارگزار، عامل را در انجام این عمل توانا می‌کند.

بخش سوم از رساله به استفاده از یک روش سریع و دقیق در مکان‌یابی روبات پرداخت و در این بخش یک الگوریتم با سرعت عملکرد $O(n \log n)$ معرفی شد که توانایی محاسبه مکان روبات در زمین را با خطای بسیار اندک دارا بود. این الگوریتم مبتنی بر روش‌های هندسه‌ی محاسباتی و تبدیل فضای محتمل برای وجود بازیکن به مجموعه‌ای از چندضلعی‌های محدب و انجام عمل اشتراک روی آن‌ها بود.

در بخش بعد تعدادی از مهارت‌های مهم و سطح بالا برای یک عامل روبوسینا معرفی شد. این مهارت‌ها بازیکن را در انجام اعمال فردی و تلاش در جهت رسیدن به هدف تیمی توانا می‌کنند. تعدادی از این مهارت‌ها نظیر شوت و پاس در قالب یک الگورتیم محاسباتی ساده معرفی شدند و استفاده از روش‌های یادگیر برای آن‌ها در فصل‌های آتی مورد بررسی قرار خواهد گرفت. اما سایر مهارت‌ها با جزئیات بیشتر مورد بررسی قرار گرفتند.

استفاده از روش آرایش تیمی جدید در تیم روبوسینا یکی از عوامل مهم در موفقیت این تیم بود که به همراه روش‌های تدافعی و تهاجمی استفاده شده در تیم در بخش‌های بعدی مورد بررسی و تحلیل قرار گرفتند. در این بخش‌ها نمونه‌هایی از استفاده‌ی روش‌های فنی فوتبال در تقابل با مفاهیم محاسباتی و ریاضی مورد بررسی قرار گرفته‌اند.

۴

گذری بر الگوریتم‌های یادگیری

همان‌طور که در ابتدای این رساله عنوان شد، روبوکاپ در واقع تلاشی در جهت ارتقاء الگوریتم‌های هوشمند است و محیط شبیه‌سازی بستر مناسبی برای تست و پیاده‌سازی ایده‌هایی است که می‌توانند روی یک سیستم هوشمند مؤثر باشند. در این فصل مجموعه روش‌های یادگیری که تیم روبوسینا در پیاده‌سازی یک عامل مورد استفاده قرار داد معرفی می‌کنیم و در فصل بعد به توضیح مهارت‌هایی که با استفاده از این تکنیک‌ها به روبات آموزش داده شد خواهیم پرداخت.

بخش ۴—۱ توضیح مختصری در خصوص درخت‌های تصمیم و الگوریتم ID3 ارائه می‌کند. در بخش ۴—۲ شبکه‌های عصبی مصنوعی [۱] را مورد مطالعه قرار می‌دهیم و الگوریتم BackPropagate را به عنوان کاربردی‌ترین الگورتیم شبکه عصبی مورد بررسی قرار خواهیم داد. در نهایت در بخش ۴—۳ روش یادگیری تقویتی [۲] را به همراه مروری بر روش Q توضیح خواهیم داد.

۴—۱ درخت‌های تصمیم

۴—۱—۱ تعریف

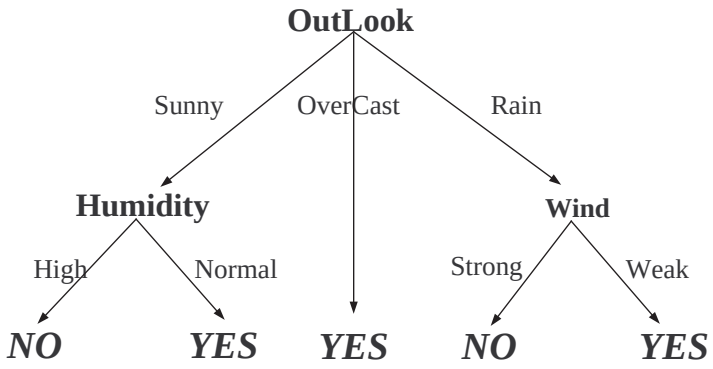
یکی از پرکاربردترین و عملی‌ترین روش‌ها برای استنتاج قیاسی [۳] یادگیری با استفاده از درخت‌های تصمیم می‌باشد که الگویی برای تقریب توابع گسسته ارائه می‌دهد. درخت‌های تصمیم نسبت به داده‌های خطادار مقاومت خوبی را دارا هستند و می‌توانند عبارات فصلی [۴] را یاد بگیرند. این الگو نمونه‌ها را در یک ساختار

[۱] Artificial Neural Networks

[۲] Reinforcement Learning

[۳] Inductive Inference

[۴] Disjunctive Expressions



شکل ۴—۱: الگویی از قوانین که می‌توان از درخت استنباط کرد

درختی مرتب می‌کند و از این رو در مواردی که برد داده‌ها گسسته است می‌تواند مورد استفاده قرار بگیرد. در درخت تصمیم هر رأس متناظر با سنجش یکی از مشخصات داده‌ی آموزشی بوده و هر یال بیان‌گر یکی از نتایجی است که از این سنجش خارج می‌شود. در روش مقایسه برای یک درخت تصمیم به این شکل عمل می‌شود که با توجه به سنجشی که در هر رأس بر اساس کلید موجود در آن رأس انجام می‌دهیم یک یال مشخص شده و ما را به سمت زیر درخت دیگری می‌برد و این عمل تا رسیدن به تفکیک مورد انتظار در یکی از برگ‌های درخت پیش می‌رود. در مواردی که با داده‌ی پیوسته مواجه هستیم، می‌توانیم با تفکیک فضا و آزمایش کلید با یک مقدار مرزی در این فضا الگوریتم را پیش ببریم. در این حال سنجش با مقدار مرزی هر گره انجام شده و ملاک حرکت روی زیردرخت‌ها بیشتر یا کمتر بودن مقدار ورودی به گره از مقدار مرزی برای کلید مورد نظر می‌باشد[۱۷، ۱]. شکل ۴—۱ نمونه‌ای از تبدیل یک درخت به مجموعه‌ای از قوانین را نشان می‌دهد که در واقع پایه‌ی ایجاد درخت برای تصمیم است.

با توجه به کلیدهای مختلفی که برای یک داده‌ی آموزشی وجود دارد، درخت‌های تصمیم متعددی را می‌توان در نظر گرفت که همه می‌توانند تمام فضای حالت را پوشش دهند و پاسخی برای مسئله طبقه‌بندی باشند. حال سؤال این است که با داشتن یک مجموعه از داده‌های آموزشی و تعدادی درخت تصمیم متفاوت که این داده‌ها را تقسیم‌بندی می‌کنند کدام درخت بیشترین احتمال برای تقسیم‌بندی درست را با توجه به نمونه‌های از فضای حالت که در داده‌های آموزشی رؤیت نشده‌اند داراست. الگوریتم ID3 پاسخی برای این سؤال است.

۴—۱—۲ الگوریتم قیاسی ID3

با توجه به آن‌چه اشاره شد باید بین مجموعه درخت‌های پوشش دهنده‌ی فضا برای داده‌های آموزشی مناسب‌ترین را در پاسخ به داده‌هایی از فضای حالت که در الگوریتم یادگیری به درخت اعمال نشده‌اند یافت. الگوریتم ID3 فرض می‌کند که ساده‌ترین درخت با قابلیت پوشش تمام فضای حالت مناسب‌ترین درخت می‌باشد. مبنای این فرض نتیجه‌ی تجربی است که ساده‌سازی و اجتناب از فرضیات و پیچیدگی‌های غیر ضروری را ارجح می‌شمارد. این مفهوم پایه‌ای که تحت عنوان "Occam's Razor" شناخته می‌شود اولین بار توسط منطق‌دان قرون وسطی William of Occam در سال ۱۳۲۴ میلادی مطرح شد:

"عبث است انجام کاری با امکانات زیاد که با منابع کمتر نیز قابل انجام است...موجودیت‌ها نباید فراتر از ضرورت تکثیر شوند [۱۸]."

الگوریتم ID3 اولین بار در سال ۱۹۸۶ میلادی توسط آقای Russ Quinlan ارائه شد و مفاهیم را با استنتاج از روی نمونه‌ها به دست می‌آورد. این الگوریتم درخت تصمیم را در یک مدل بالا به پایین ایجاد می‌کند و همان‌طور که اشاره شد برای هر خصوصیت باید مجموعه داده‌های آموزشی را به زیرمجموعه‌های گسسته تقسیم کنیم به نحوی که تمام نمونه‌هایی که در یک مجموعه قرار می‌گیرند مقدار مشترکی را در ازای آن خصوصیت دارا باشند. ID3 در هر مرحله یک خصوصیت را برای سنجش در گره حاضر انتخاب می‌کند و با استفاده از آن مجموعه داده‌ها را تقسیم می‌کند و الگوریتم به شکل بازگشتی همین عمل را برای زیر درخت‌ها انجام می‌دهد و تا زمانی ادامه می‌دهد که تمام اعضا یک بخش در همان بخش قرار بگیرند. از آنجا که ID3 به شدت به شرطی که برای تست در هر گره استفاده می‌کند متکی می‌باشد زمان صرف شده برای انتخاب این معیار، ملاک بحرانی در ایجاد یک درخت تصمیم ساده است. در این بخش با فرض این که الگوریتم انتخاب معیار تست در هر گره بهینه است به توضیح الگوریتم می‌پردازیم و در بخش بعد روش ایجاد شاخص را توضیح می‌دهیم.

method induceTree ($training - examples$, $Properties$)
  if all enteries in $training - examples$ are in the same class **then**
    return a leaf node labeled with that class
  else if $Properties$ is Empty **then**
    return leaf node labeled with disjunction of all classes in $training - examples$
  else
    select a property, P, and make it the root of the current tree
    delete P from $Properties$
    foreach value, V, of P **do**
      create a branch of the tree labeled with V
      let $partition_v$ be elements of $trainig - examples$ with values V for property P
      call induceTree ($partition_v$, $Properties$)
      attach results to branch V
    end foreach
  end if
end method

که در الگوریتم بالا $training - examples$ نشان دهنده مجموعه‌ی داده‌های آموزشی و $Properties$ در بر گیرنده‌ی مجموعه خصوصیاتی است که هر عضو از داده‌ی آموزشی داراست.

۴—۱—۳ انتخاب شاخص سنجش در ID3 بر اساس محتوای علمی آن

هر خصوصیت در یک نمونه‌ی آزمایشی حاوی مقدار مشخصی از اطلاعات لازم برای تقسیم‌بندی آن نمونه است. ID3 سعی می‌کند تا این مقدار اطلاعات را برای هر خصوصیت محاسبه کند و سپس خصوصیتی که بیشترین مقدار اطلاعات را دارد در هر مقطع به عنوان شاخص مناسب برای قرار گرفتن در ریشه زیر درخت انتخاب می‌شود. تئوری اطلاعات که در سال ۱۹۴۸ میلادی توسطShannon معرفی شد یک پایه‌ی ریاضی را برای اندازه‌گیری محتوای اطلاعاتی یک پیام فراهم می‌آورد که در این نظریه یک پیام می‌تواند هر

نمونه‌ای در یک دنیا از پیام‌های ممکن باشد و عمل انتقال پیام همان انتخاب یکی از مجموعه پیام‌های ممکن است. از این نقطه نظر منطقی است که محتوای اطلاعاتی یک پیام را تابعی از دو عامل اندازه‌ی دنیایی که در آن وجود دارد و تکرر وجود آن پیام دانست.

Shannon محتوی اطلاعاتی یک پیام را به عنوان تابعی از احتمال تکرار آن پیام فرموله کرد. با داشتن یک فضای $M = \{m_1, m_2, ..., m_n\}$ از پیام‌ها و احتمال $p(m_i)$ برای رخداد یک پیام محتوای اطلاعاتی یک پیام در M برابر است با:

$$I(M) = \sum_{i=1}^{n} -p(m_i) \log_2(p(m_i))$$

ID3 تئوری اطلاعات را برای انتخاب شاخص آزمایشی که بیشترین محتوای اطلاعاتی را در تقسیم‌بندی نمونه‌های آموزشی داراست به کار می‌گیرد. بهره‌ی اطلاعات[۵] که از انجام یک سنجش در ریشه درخت فعلی به دست می آید برابر است با تمام مجموعه اطلاعات در درخت منهای مقدار اطلاعاتی که برای کامل کردن عملیات طبقه‌بندی پس از اعمال این سنجش لازم است. مقدار اطلاعاتی که برای کامل کردن درخت لازم است تحت عنوان میانگین وزن‌دار اطلاعات در تمام زیر درخت‌های آن تعریف می‌شود که این مقدار با ضرب کردن محتوای اطلاعاتی هر زیر درخت در درصد نمونه‌هایی که در زیر درخت وجود دارند و جمع این حاصل ضرب‌ها انجام می‌شود. فرض می‌کنیم C نشان دهنده‌ی مجموعه‌ی داده‌های آموزشی است. اگر خصوصیت P را در بین n مقدار به عنوان ریشه درخت فعلی در نظر بگیریم در این حال C با توجه به مقادیر ممکن برای P به زیرمجموعه‌های $\{C_1, C_2, ..., C_n\}$ تقسیم می‌شود. در این صورت مقدار اطلاعاتی که برای کامل کردن درخت پس از قرار دادن P به عنوان شاخص ریشه برابر است با:

$$E(P) = \sum_{i=1}^{n} \frac{|C_i|}{|C|} I(C_i)$$

در این حال بهره‌ی اطلاعاتی از خصوصیت P را می‌توان با تفاضل اطلاعات مورد انتظار برای کامل کردن درخت از کل محتوای اطلاعاتی درخت به دست آورد، یعنی:

$$gain(P) = I(C) - E(P)$$

۴—۱—۴ C4.5 چیست؟

C4.5 یک نرم‌افزار توسعه یافته بر مبنای الگوریتم پایه‌ای ID3 است که توسط Quinlan طراحی شده تا مفاهیم زیر را که توسط ID3 به تنهایی برآورده نمی‌شوند قابل دستیابی کند:

- اجتناب از بارگذاری بیشینه‌ی داده
  - ⋆ تعیین عمق رشد برای یک درخت تصمیم
- کاهش خطا در حرص کردن درخت

[۵] Information Gain

- قانونمند کردن حرص پس از ایحاد درخت
- کارکردن با داده‌های پیوسته
  - ⋆ به عنوان مثال داده‌های متریک در یک روبات
- انتخاب یک مقیاس مناسب در گزینش خصوصیت
- کارکردن با داده‌های آموزشی که فاقد بعضی از خصوصیات هستند
- کارکردن با خصوصیاتی با هزینه‌های متغیر
  - ⋆ تأثیرگذاری یک خصوصیت روی سیستم برای تمام داده‌های آموزشی یکسان نیست
- بهینه‌کردن کیفیت محاسباتی

این نرم‌افزار مجموعه‌ای از داده‌های آموزشی را در تعدادی فایل با فرمت مشخص می‌گیرد و سپس با اعمال الگوریتم ID3 و بهینه‌سازی با توجه به خواص فوق یک درخت تصمیم را ایجاد می‌کند که قابلیت تفکیک داده‌ها برای نمونه‌های بیرون از فضای آموزشی را فراهم می‌کند. این نرم‌افزار طبقه‌بندی، خصوصیت، و مقادیر ممکن برای هر خصوصیت را، که می‌تواند یک بازه‌ی گسسته یا پیوسته باشد، از یک فایل با عنوان X.names می‌خواند و با توجه به این مقادیر و فایل مربوط به داده‌های آموزشی با عنوان X.data الگوریتم ID3 را اجرا نموده و نتیجه را در فایل‌هایی با عناوین X.unpruned، X.tree و X.test برمی‌گرداند که در آن X نام فایلی است که عمل مقایسه را مشخص می‌کند. X.unpruned درخت حرص نشده‌ی حاصل از اجرای الگوریتم و X.tree درخت نهایی است که ساخته می‌شود. یادآوری می‌شود که فایل X.names شامل معرفی تمام مجموعه اطلاعاتی است که برای شروع ایجاد یک درخت باید از محیط جمع‌آوری شوند.

با استفاده از این نرم‌افزار می‌توان به سادگی داده‌های آموزشی را در یک محیط جمع‌آوری نمود، طبقه‌های لازم برای قرارگرفتن هر داده را تشخیص داده، خصوصیات را استخراج کرد و با مشخص نمودن دامنه هریک از این خصوصیات فایل X.names را ساخت. سپس با ارسال داده‌های آموزشی به این نرم‌افزار ظرف مدت کوتاهی مدل درختی ممکن برای تفکیک داده‌ها را پس از اِعمال ID3 به دست آورد. توضیح بیشتر در خصوص این نرم‌افزار را می‌توان در مرجع [۱۹] یافت.

۴—۲ شبکه‌های عصبی مصنوعی

۴—۲—۱ تعریف

مغز انسان به طور تقریبی متشکل از بیش از ۱۰۰ میلیارد ($۱۰^{۱۱}$) نورون می‌باشد و تعداد $۱۰^{۱۴}$ اتصال عصبی در بدن آدمی انتقال دهنده جریانی است که سیستم استنتاج را از ساختار مغز به اعضاء منتقل می‌کند و این بدان معنی است که در ابتدا و انتهای هر نورون عصبی حدود ۱۰۰۰ اتصال عصبی وجود دارد. شبکه عصبی مصنوعی یک شبیه‌سازی مجرد از یک سیستم عصبی حقیقی است که شامل مجموعه‌ای از

واحدهای عصبی با قابلیت ارتباط از طریق آکسون‌های عصبی است که در مدل ریاضی با یال‌های یک گراف مشخص می‌شوند. این مدل نمایش بسیار نزدیکی از شرایط عصبی واقعی برای موجودات زنده است. با توجه به ساختار هماهنگ کننده و خودسازمانده، مدل به شکل بالقوه‌ای یک بازه جدید از پردازش موازی را پیشنهاد می‌کند که می‌تواند در مقایسه با مدل‌های مرسوم قبلی پایدارتر و ساده‌تر باشد. این مدل اولین بار در سال ۱۹۴۳ و توسط آقایان McCulloch و Pitts تحت عنوان یک مدل محاسباتی از فعالیت‌های عصبی مطرح شد.

از جنبه‌ی استفاده‌ی کاربردی، می‌توان قدرت شبکه‌های عصبی را در حل مسائل غیرخطی، نگاشت متناظر و پردازش موازی دانست اما به طور کلی بازه‌ی کاربرد شبکه‌های عصبی به محورهای اصلی تناظر[۶]/خوشه‌بندی[۷] /طبقه‌بندی[۸]، تکمیل الگو، بازگشت/تعمیم، و بهینه‌سازی قابل تقسیم است[۱۶]. شبکه‌های عصبی بر حسب الگوریتم یادگیری متناظر به سه دسته زیر تقسیم می‌شوند:

۱) شبکه‌های وزن ثابت[۹]: در این حالت روش یادگیری مطرح نیست و لذا شبکه‌های عصبی یادگیر شامل دو دسته زیر می‌باشند.

۲) شبکه‌های عصبی باناظر[۱۰]: در این مدل داده‌ی آموزشی شامل مجموعه‌ای از زوج‌های ورودی/خروجی است و بنابراین روش یادگیری از کمک داده‌ی ناظر یا معلم در آموزش شبکه سود خواهد برد.

۳) شبکه‌های عصبی بدون‌ناظر[۱۱]: در این حالت داده‌ی آموزشی تنها شامل داده‌ی ورودی است، بنابراین شبکه بدون حضور معلم آموزی داده می‌شود. در این روش یادگیری با توجه به تجربه‌ای که از یادگیری الگوهای پیشین حاصل شده است پیش می‌رود که می‌توان از مهم‌ترین مدل‌های این روش به **یادگیری رقابتی** اشاره کرد.

۴—۲—۲ بیان ریاضی برای یک شبکه عصبی

یک شبکه عصبی پایه‌ای در شکل ۴—۲ نشان داده شده است که می‌تواند با توضیح عملکرد اتصال شبکه و فعالیت نورون مشخص شود. هر سلول عصبی (واحد پردازش) دارای مقدار نورونی a_j می‌باشد که این مقدار به کمک انتشار از طریق ارتباطات یک سویه در شبکه به سایر سلول‌ها منتقل می‌شود. متناظر به هر اتصال یک وزن سیناپسی وجود دارد که با w_{ij} مشخص می‌شود و اثر سلول شماره‌ی j بر روی سلول شماره‌ل i را معین می‌کند. مقدار ورودی به سلول iام از سوی سایر سلول‌ها به همراه یک مقدار خارجی بایاس با نام θ_i جمع می‌شود تا مقدار خروجی متناظر با u_i را شکل دهد. پس از آن مقدار u_i با گذر از یک تابع غیر خطی تحت عنوان **تابع فعال‌سازی** مقدار فعال‌سازی جدیدی با عنوان a_i ایجاد می‌کند. در واقع مقدار نهایی برای خروجی y می‌تواند به عنوان تابعی از مقدار ورودی x و وزن‌های W به شکل $y = \phi(x, W)$ بیان شود.

---

[۶] Association

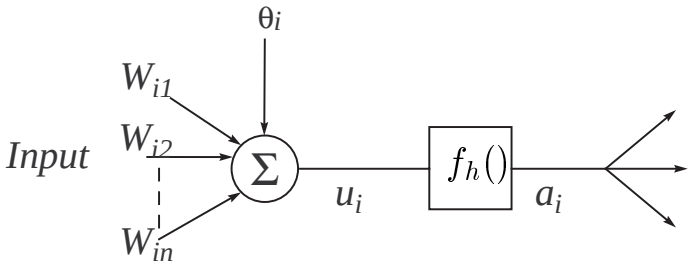
[۷] Clustring

[۸] Classification

[۹] fixed-weight

[۱۰] supervised

[۱۱] unsupervised



شکل ۴—۲: یک شبکه عصبی پایه

دو تابع اصلی را می‌توان برای شبکه در نظر گرفت:

- تابع پایه یا تابع شبکه: که در شکل ساده یک تابع خطی بنیادی[۱۲] یا LBF می‌باشد و مقدار خروجی به شکل ترکیب خطی از مقادیر ورودی تعریف می‌شود:

$$u_i(x,w) = \sum_{j=1}^{n} x_j \cdot w_{ij}$$

- تابع فعال‌سازی: مقادیر شبکه پس از آن که با استفاده از تابع پایه $u(x,w)$ مشخص شدند بلافاصله به یک تابع غیر خطی از نورون منتقل می‌شوند که تابع فعال‌سازی بوده و عمدتاً یک مقدار نرمال‌شده را در خروجی نتیجه می‌دهد. متداول‌ترین نمونه‌هابرای تابع فعال‌سازی $step, ramp, sigmoid$ و $Gaussian$ می‌باشند. در زیر دو نمونه از این توابع آورده شده است:

  ⋆ تابع $sigmoid$:

$$f_h(u_i) = \frac{1}{1 + e^{-u_i/\sigma}}$$

  ⋆ تابع $Gaussian$:

$$f_h(u_i) = ce^{-u_i^2/\sigma^2}$$

با توجه به روابط بالا می‌توان نوشت:

$$\phi(x,W) = f_h\left[\sum_{j=1}^{n} x_j \cdot w_{ij} + \theta_i\right]$$

---

[۱۲] Linear Basis Function

۴—۲—۳ پرکاربردترین شبکه‌ی عصبی

همان طور که اشاره شد، شبکه‌های عصبی در حل مسائل طبقه‌بندی و خوشه‌بندی مفید هستند و از این حیث با درخت‌های تصمیم مشابه بوده و می‌توانند در حل چنین مسائلی جایگزین درخت‌های تصمیم باشند. به علاوه در مواردی که داده‌های ورودی دارای خطا بوده و یا از طریق حسگرهای پیچیده‌ای دریافت می‌شوند استفاده از شبکه‌های عصبی توصیه می‌شود [۱۷]. این شبکه‌ها بر حسب نوع تابع فعال‌سازی به دو دسته خطی و غیرخطی تقسیم می‌شوند که شبکه‌های غیرخطی توانایی تقریب یا طبقه‌بندی در حجم بیشتری از مسائل را دارا می‌باشند. به علاوه تعداد لایه‌هایی که می‌تواند یک مجموعه داده‌ی ورودی را به یک مجموعه‌ی خروجی بنگارد متغیر بوده و استفاده از یک شبکه چند لایه قابلیت‌های تقریب در شبکه را به شکل چشمگیری افزایش می‌دهد.

در این بین الگوریتم BackPropagate یک دستیابی مؤثر برای حل مسائل طبقه‌بندی و تقریب می‌باشد که می‌تواند بر روی هر فرمول بهینه‌سازی اعمال شود. یک شبکه BackPropagate با استفاده از LBF برای تابع شبکه و $sigmoid$ برای تابع فعال‌سازی به صورت زیر بیان می‌شود:

$$u_i(l) = \sum_{j=۱}^{N_{l-۱}} w_{ij} \cdot a_j(l-۱) + \theta_i(l)$$

$$a_i(l) = f(u_i(l)) \qquad l \le i \le N_{li}, \qquad ۱ \le l \le L$$

که در آن داده‌های ورودی با $x_i \equiv a_i(۰)$ و داده‌های خروجی با $y_i \equiv a_i(L)$ معرفی می‌شوند و L تعداد لایه‌ها را نشان می‌دهد. فرمول پایه‌ای که برای یادگیری استفاده می‌شود برای mامین الگوی یادگیری، $a^{(m)}(۰)$، و معلم متناظر با آن $t^{(m)}$، که $m = ۱, ۲, ..., M$ به صورت زیر می‌باشد:

$$w_{ij}^{(m+۱)} = w_{ij}^{(m)}(l) + \triangle w_{ij}^{(m)}(l)$$

که در آن $\triangle w_{ij}^{(m)}(l)$ انحراف از مقدار وزن حقیقی می‌باشد و از مجموعه روابط زیر پیروی می‌کند:

$$\begin{aligned}\triangle w_{ij}^{(m)}(l) &= -\eta \frac{\partial E}{\partial w_{ij}^{(m)}(l)} \\ &= -\eta \frac{\partial E}{\partial a_i^{(m)}(l)} \frac{\partial a_i^{(m)}(l)}{\partial w_{ij}^{(m)}(l)} \\ &= \eta \delta_i^{(m)}(l) f'(u_i^{(m)}(l)) a_j^{(m)}(l-۱)\end{aligned}$$

و

$$\delta_i^{(m)}(l) \equiv -\frac{\partial E}{\partial a_i^{(m)}(l)}$$

در این شرایط هدف آموزش وزن‌های w_{ij} و بایاس θ_i به نحوی است که مقدار حداقل مربع خطا[۱۳] بین داده‌ی ناظر و خروجی واقعی کمینه شود. یعنی

$$E = \frac{۱}{۲} \sum_{m=۱}^{M} \sum_{i=۱}^{N} \left[t_i^{(m)} - a_i^{(m)}(L) \right]^۲$$

---

[۱۳] least-square-error

که M تعداد الگوهای آموزشی و N ابعاد فضای خروجی را نشان می‌دهد. با توجه به مقدار E، برای $\delta_i^{(m)}(l)$ داریم:

$$\delta_i^{(m)}(l) = t_i^{(m)} - a_i^{(m)}(L)$$

۴—۲—۴ الگوریتم BackPropagate

با توجه به فرمول‌هایی که در بالا به آنها پرداختیم می‌توانیم یک مدل ساده از الگوریتم BackPropagate را مطرح کنیم[۱۷].

فرض می‌کنیم هر داده‌ی آموزشی یک جفت به شکل $(\vec{x}, \vec{t})$ می‌باشد که در آن $\vec{x}$ بردار مقادیر ورودی به شبکه و $\vec{t}$ بردار مقادیر هدف می‌باشد، η برابر با **"learning-rate"** و به عنوان مثال دارای مقدار ۰/۰۵ بوده، n_{in} تعداد ورودی‌ها به شبکه n_{out} تعداد واحدهای خروجی و n_{hidden} تعداد واحدهای پنهان می‌باشد، ورودی از واحد i به واحد j با x_{ij} و وزن یالی که i را به j وصل می‌کند با w_{ij} نشان داده می‌شود، L تعداد لایه‌ها در شبکه و $m = ۱, ۲, ...M$ دفعات تکرار الگوریتم می‌باشد.

method backPropagation ($training - examples$, η, n_{in}, n_{out}, n_{hidden}, L)
  create a feed-forward network with n_{in} inputs, n_{hidden} hidden units, and n_{out} output units
  initialize all weights to small random numbers (e.g. between -0.005, 0.005)
  while the termination condition is not met for m^{th} time **do**
    foreach pair $(\vec{x^{(m)}}, \vec{t^{(m)}})$ in $training - example$ **do**
      input $\vec{x^{(m)}}$ to the network and compute y_i for every unit u of the network
      foreach output y_i in network layer l **do**
        calculate its error term $\delta_i^{(m)}(l)$ as:
        if $l == L$ **then**
          $\delta_i^{(m)}(l) = y_i^{(m)}(۱ - y_i^{(m)})(t_i^{(m)} - y_i^{(m)})$
        else
          $\delta_i^{(m)}(l) = y_i^{(m)}(۱ - y_i^{(m)}) \sum_{j \in output} w_{ji}\delta_j$
        end if
      end foreach
      update each network weight $w_{ji}^{(m+۱)}(l) = w_{ji}^{(m)}(l) + \triangle w_{ji}^{(m)}$, where $\triangle w_{ji}^{(m)} = \eta\delta_j^{(m)}x_{ji}$
    end foreach
  end while
end method

۴—۳ یادگیری تقویتی

۴—۳—۱ تعریف

یادگیری تقویتی فراگیری آنچه است که باید انجام شود، یعنی نحوه نگاشت وضعیت‌ها به اعمال قابل انجام، به نحوی که سیگنال عددی پاداش انجام آن عمل بیشینه شود[۲۰]. عامل یادگیر با آزمایش تمام مجموعه اعمالی که می‌تواند انجام دهد در نهایت درمی‌یابد که کدام یک از این اعمال بیشترین میزان پاداش را در صورت انجام به دنبال خواهد داشت. در حالت مهم‌تر و پیچیده‌تر اعمال انجام شده از سوی عامل نه تنها در پاداش دریافت شده‌ی آنی تأثیرگذار است بلکه وضعیت بعدی و به دنبال آن تمام پاداش‌های متعاقب را نیز تحت تأثیر قرار می‌دهد. جستجو با آزمایش و خطا و پاداش با تأخیر دو خصوصیتی از یادگیری تقویتی هستند که آن را نسبت به سایر روش‌های یادگیری متمایز می‌کنند. روش یادگیری تقویتی به کل مسئله به صورت یک مسئله **هدف‌گرا**[۱۴] می‌نگرد که در آن عامل با یک محیط غیر قطعی در ارتباط است. این نوع از نگرش از محدودیت‌های بسیاری که به واسطه تمرکز روی زیرمسئله‌ها به طور مجزا ایجاد می‌شوند جلوگیری می‌کند.

یادگیری تقویتی مبتنی بر جستجو[۱۵] و بهره‌برداری[۱۶] می‌باشد و از آنجا که معمولاً این عمل در یک فضای یادگیری بزرگ انجام می‌شود و عامل باید برای به دست آوردن پاداش مناسب این فضا را به شکل کامل جستجو کند ایجاد یک تعادل مناسب بین میزان اکتشاف و میزان استخراج نقش بسیار مهمی در دقت و سرعت یادگیری دارد. جستجو برای ایجاد انتخابات بهتر در آینده مورد استفاده قرار می‌گیرد و بهره‌برداری نیز استفاده از اطلاعاتی است که زودتر به آن دست یافته‌ایم، اما هیچ یک از آن‌ها نمی‌توانند به شکل انحصاری دنبال شوند. از سوی دیگر یادگیری تقویتی از یک مدل یادگیر با ناظر، که وابسته به نمونه‌های از پیش تجربه شده است، پیروی نمی‌کند چرا که در تقابل یک عامل با محیط، یادگیری از طریق داده‌های از پیش فراهم شده که بر تمام فضای حالت پوشا باشند و اطلاعات صحیحی داشته باشند چندان ممکن نیست و در چنین محیطی عامل باید قادر به یادگیری بر پایه‌ی تجربه‌های خود باشد[۲۰].

به طور کلی یادگیری تقویتی بر پایه‌ی سه مفهوم اصلی استوار است که عبارتند از:

۱) State: هر چیزی که باید در ارتباط با محیط دانست و برای یادگیری سودمند می‌باشد.

۲) Action: هر تصمیمی که به دنبال راهی برای اتخاذ آن هستیم.

۳) Goal: نتیجه‌ی نهایی که باید به آن دست یافت.

با توجه به این سه مفهوم در یک سیستم یادگیر مکانیزمی تعریف می‌شود که اطلاعات فوق را به یک الگوی یادگیری تبدیل کند. این مکانیزم مشتمل بر مفاهیم زیر است:

- Policy: نگاشت بین Stateها و actionها.
- Reward: تأثیر بلادرنگ یک action روی محیط و نتیجه‌ی آنی آن.
- Value: اثر بلند مدت پرداختن به actionها در محیط.
- مدل محیطی[۱۷]: آنچه که رفتار محیط را منعکس می‌کند.

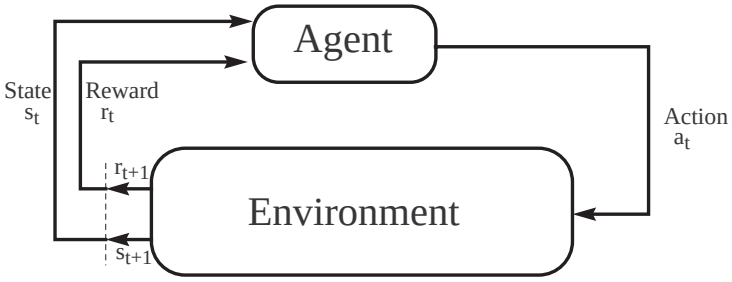
---

[۱۴] goal-directed

[۱۵] exploration

[۱۶] exploitation

[۱۷] Model of the environment



شکل ۴—۳: مدل یادگیری تقویتی

با توجه به مجموعه مفاهیم فوق، با تشخیص Stateها، Actionها و Goal یا Goalها، و با ایجاد Policy، Reward، Value و مدل محیطی، امکان یادگیری فراهم می‌شود.

۴—۳—۲ فرآیند یادگیری

تحقیقات مدرن در زمینه یادگیری تقویتی از چارچوب رسمی پردازش تصمیم Markov[۱۸] استفاده می‌کنند. در این چارچوب عامل و محیط در یک مجموعه از سیکل‌های زمانی گسسته با هم تقابل می‌کنند $t = ۰, ۱, ۲, ۳, ...$. در هر گام عامل محیط را در قالب یک حالت s_t درک می‌کند و عمل a_t را انتخاب می‌کند. در پاسخ، محیط، یک گذر تصادفی را به حالت جدید $s_{t+۱}$ انجام می‌دهد و به شکل اتفاقی یک مقدار عددی $r_{t+۱} \in \Re$ را به عنوان پاداش ارسال می‌کند (شکل ۴—۳[۲۰]).

عامل به دنبال راهی در جهت افزایش پاداشی است که از سیستم دریافت می‌کند. به عنوان مثال متداول‌ترین انگیزه انتخاب هر عمل a_t به شکلی است که مقدار *expected discount return* بیشینه شود. این مقدار به شکل زیر تعریف می‌شود :

$$E\{r_{t+۱} + \gamma r_{t+۲} + \gamma^۲ r_{t+۳} + ...\}$$

که در آن γ پارامتر discount-rate و $۰ \leq \gamma \leq ۱$ می‌باشد.

روش‌های یادگیری تقویتی به دنبال بهینه‌سازی در سیاست اتخاذ تصمیم برای یک عامل می‌باشند. این سیاست یا Policy همان‌طور که گفته شد نگاشتی از حالت‌ها به اعمال و یا به احتمال توزیع شده روی اعمال است. سیاست در یک قالب نسبتاً شفاف ذخیره شده است به گونه‌ای که در مواجهه با حالات دور از انتظار پاسخ‌های مناسبی قابل تولید است. می‌توانیم ارزش بودن در حالت s تحت سیاست π را به عنوان expected discount return با شروع از آن حالت و پیروی از سیاست π تعریف کنیم. تابعی که در این راستا مقادیر همه‌ی حالت‌ها را به ارزش آن‌ها می‌نگارد تابع State-Value برای آن Policy خوانده می‌شود.

$$V^\pi(s) = E_\pi\{r_{t+۱} + \gamma r_{t+۲} + \gamma^۲ r_{t+۳} + ... | s_t = s\}$$

[۱۸] Markov decision process

سیاست π تنها در شرایطی بهتر از سیاست π' خوانده می‌شود که برای تمام حالات s داشته باشیم $V^{\pi}(s) \geq V^{\pi'}(s)$. در یک مدل متناهی MDP همواره یک یا چند سیاست وجود دارند که نسبت به بقیه بهتر یا با آن‌ها مساوی هستند. این سیاست‌ها، **سیاست‌های بهینه** خوانده می‌شوند و همه‌ی آن‌ها یک تابع State-Value را با هم به اشتراک می‌گذارند.

۴–۳–۳ مفاهیم یادگیری Q

ساده‌ترین حالت از یادگیری تقویتی مستقیماً به تجربه‌های عاملی که با محیط در تقابل است اعمال می‌شود و سیاست را در یک روش زمان‌حقیقی تغییر می‌دهد. از این نمونه می‌توان به روش *one-step tabular Q-learning* یا یادگیری تک گام جدولی Q اشاره کرد که از ساده‌ترین انواع یادگیری تقویتی بوده و با گذر از هر حالت یکی از عناصر جدول را به‌روزرسانی می‌کند. این جدول، که با Q مشخص می‌شود، برای هر زوج حالت s و عمل a یک ورودی $Q(s,a)$ دارد. با گذر از حالت s_t به s_{t+1} پس از انجام عمل a_t و دریافت پاداش r_{t+1} روش به‌روزرسانی در این الگوریتم به شکل زیر می‌باشد:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t) \right]$$

که در آن α یک پارامتر اندازه‌ی گام[۱۹] مثبت است. با استفاده از قوانین مناسب (با تضمین اکتشافات کافی و کاهش میزان α با مرور زمان) این فرآیند همگرا می‌شود، به صورتی که اِعمال یک سیاست حریصانه بر روی جدول Q جواب بهینه را ایجاد می‌کند. سیاست حریصانه انتخاب عمل a در هر حالت s به شکلی است که برای آن عمل مقدار $Q(s,a)$ بیشینه باشد. بنابراین با استفاده از این الگوریتم امکان رسیدن به یک سیاست بهینه تنها با توجه به تجربیات به دست آمده و بدون داشتن مدل پویا از محیط ممکن می‌شود.

آنچه در بالا اشاره شد یک نمونه بسیار ساده از روش‌های یادگیری تقویتی است. روش‌های پیچیده‌تر Q را در قالب یک جدول طراحی نمی‌کنند بلکه آن را به عنوان یک تابع پارامتریک قابل آموزش نظیر یک شبکه عصبی پیاده می‌کنند. با استفاده از این عمل امکان تعمیم خصوصیت روی حالت‌ها به سادگی فراهم می‌شود که به مقدار قابل ملاحظه‌ای زمان یادگیری و حافظه‌ی مورد نیاز را کاهش می‌دهد. در شرایطی که نگاشت فضای حالت به یک جدول هزینه‌ی محاسباتی و حافظه‌ی زیاد را تحمیل می‌کند، استفاده از روش‌های یادگیر برای قالب Q بسیار سودمند است. در زیر الگوریتم یادگیر Q برای مدل ساده و تک گام جدولی آورده شده است. مطالب بیشتر در خصوص سایر انواع یادگیری تقویتی را می‌توان در مرجع [۲۰] یافت.

method Q-Learning
  initialize $Q(s,a)$ arbitrary
  foreach possible episode **do**
    initialize s
    repeat foreach step of episode **do**
      choose a from s using policy derived from Q
      take action a, observe r, s'
      $Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$

[۱۹] step-size

```
        s ← s'
      end foreach
    until s is not terminal
  end foreach
end method
```

۴—۴ بازنگری فصل

در این فصل از رساله تعدادی از الگوریتم‌های یادگیری استفاده شده در تیم روبوسینا مورد مطالعه و بررسی قرار گرفتند. این الگوریتم‌ها در سه بخش درخت‌های تصمیم، شبکه‌های عصبی و یادگیری تقویتی معرفی شدند و در حال حاضر از پرکاربردترین روش‌های یادگیری محسوب می‌شوند. در بخش اول درخت‌های تصمیم و به طور خاص الگوریتم ID3 و نحوه‌ی انتخاب شاخص گزینش در آن به همراه نرم‌افزار C4.5 معرفی شدند. در بخش دوم مفاهیم شبکه‌های عصبی به همراه الگوریتم BackPropagate معرفی شدند و در نهایت در بخش آخر یادگیری تقویتی با توضیح مختصر روی الگوریتم Q مورد مطالعه قرار گرفتند. در ادامه‌ی این رساله بخش‌هایی از عامل روبوسینا که با استفاده از این ایده‌ها پیاده‌سازی شده‌اند را مورد بررسی قرار می‌دهیم.

۵

عامل روبوسینا و یادگیری

با توجه به آنچه که در ابتدا به آن پرداختیم سیستم شبیه‌سازی بستر مناسبی برای آزمایش الگوریتم‌های هوش مصنوعی معرفی شد. در سیستم‌های موجود در دنیای واقعی هزینه‌های سنگین زمانی و مالی همواره برای ایجاد یک بستر مناسب سخت‌افزاری صرف می‌شوند تا قابلیت آزمایش روش‌های هوشمند را فراهم آورند. سیستم شبیه‌سازی با حذف این پیچیدگی‌های سخت‌افزاری امکان آزمایش روش‌های مختلف یادگیری را بدون نیاز به امکانات مالی زیاد فراهم نموده است. متداول‌ترین روش‌هایی که می‌توان برای تصحیح و آموزش عملکرد یک روبات به کار گرفت درخت‌های تصمیم، الگوریتم‌های هوش مصنوعی، الگوریتم‌های ژنتیک و درنهایت یادگیری تقویتی است. در فصل ۴ توضیح مختصری در رابطه با مفاهیم اصلی روش‌های استفاده شده در آموزش یک عامل روبوسینا ارائه شد و در این فصل سعی بر آن است که با استفاده از این روش‌ها به توضیح یادگیری دو مهارت مهم که در عامل روبوسینا مورد آزمایش و پیاده‌سازی قرار گرفته است بپردازیم.

بخش ۵—۱ استفاده از یک شبکه عصبی BackPropagate را برای آموزش مهارت شوت توضیح خواهد داد. هدف از پیاده‌سازی این مهارت توانا نمودن بازیکن در اتخاذ تصمیم مناسب در شلیک توپ به سمت دروازه و تضمین عبور توپ از دروازبان و ورود آن به داخل دروازه می‌باشد. در بخش ۵—۲ استفاده از الگوریتم ID3 و نرم‌افزار C4.5 برای آموزش مهارت پاس دادن بررسی خواهد شد. این مهارت از زاویه‌ی دید بازیکن ارسال کننده پاس بررسی می‌شود و در آن هدف عبور توپ از بین مجموعه‌ای از بازیکنان خودی و حریف و در نهایت تضمین رسیدن آن به یک نقطه‌ی خاص خواهد بود.

۵—۱ استفاده از شبکه عصبی در مهارت شوت

از مهم‌ترین انگیزه‌ها در یک بازی فوتبال به ثمر رسانیدن گل می‌باشد و لذا برای یک عامل فوتبالیست اتخاذ تصمیم درست در امکان به دست آوردن امتیاز با توجه به وضعیت فعلی بسیار مهم می‌باشد. در هنگام تصمیم برای شوت به سمت دروازه همان طور که در بخش ۳—۴—۵ اشاره شد بازیکن باید هم از ورود توپ به دروازه اطمینان پیدا کند و هم مناسب‌ترین نقطه برای شلیک توپ را تشخیص دهد. این نقطه باید به نحوی انتخاب شود که نه دروازبان در تصاحب توپ توانا باشد و نه توپ از دروازه خارج شود. استفاده از این

روش در شوت نه تنها بازیکن را در تشخیص مناسب‌ترین نقطه برای شلیک توپ توانا می‌کند بلکه احتمال ورود توپ به این نقطه را نیز تخمین می‌زند.

با توجه به مطالب بخش ۴—۲ شبکه‌های عصبی ابزار مناسبی برای طبقه‌بندی و تعمیم هستند و این ویژگی امکان استفاده از آن‌ها را به عنوان ابزاری برای نگاشت مجموعه‌ای از داده‌ها به یک حالت خاص فراهم می‌آورد. در ادامه الگوی طراحی شده در آموزش روبات را معرفی می‌کنیم.

۵—۱—۱ تعریف و ایجاد محیط مسئله

هدف در این مسئله آموزش تابع f برای شلیک به سمت دروازه به نحوی است که مناسب‌ترین نقطه برای شلیک توپ به سمت دروازه، در صورت امکان ورود توپ به دروازه، خروجی آن باشد. یک بازیکن با دریافت اطلاعات محیطی از طریق حسگرهای خود، اِعمال پارامترهای سودمند به تابع f و یادگیری یک شبکه عصبی برای مدل کردن تابع f می‌تواند به این مهم دست یابد.

برای تبدیل مسئله به یک الگوی قابل حل توسط شبکه‌ی عصبی باید اعمال زیر را انجام دهیم:

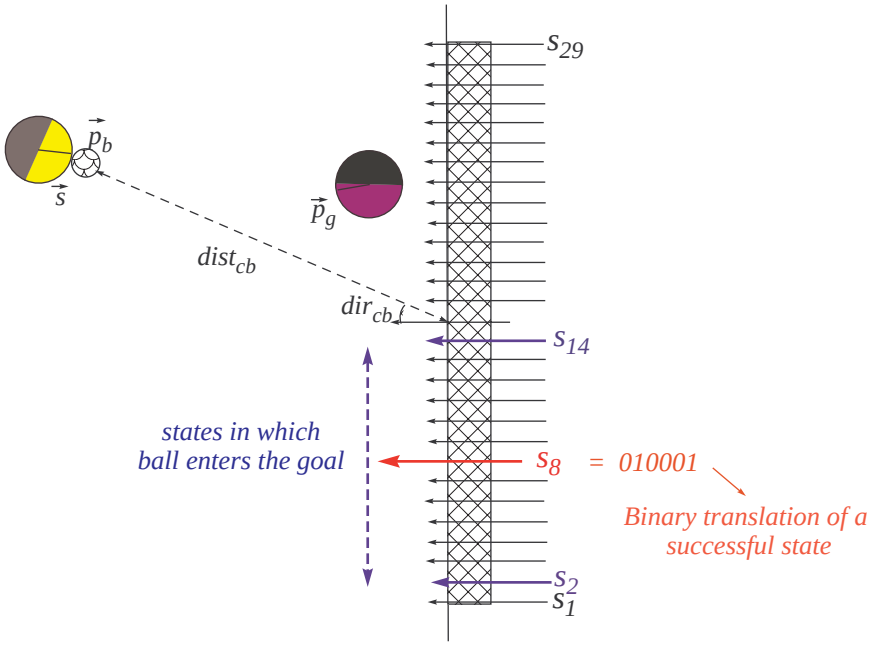
۱) تبدیل فضای پیوسته‌ی زمین بازی به یک فضای گسسته به صورتی که امکان نگاشت حالت‌های جمع‌آوری شده به بخش‌های مختلف این تقسیم‌بندی در قالب یک تناظر یک به یک وجود داشته باشد.

۲) استخراج داده‌های مورد نیاز از محیط که در واقع شامل مجموعه‌ای از اطلاعات است که برای تصمیم‌گیری در خصوص شوت لازم و کافی هستند.

۳) استفاده از trainer در قرار دادن عامل مهاجم، با توانایی در انجام مهارت‌های اولیه نظیر تصاحب توپ و ضربه به سمت یک نقطه خاص، و نیز یک دروازبان متبحر در زمین بازی و ایجاد یک episode آزمایشی که با شلیک توپ توسط بازیکن مهاجم آغاز می‌شود و در صورت گل شدن، خروج از زمین و یا تصاحب توپ از سوی دروازبان پایان می‌پذیرد.

۴) نحوه‌ی نگاشت داده‌ی آموزشی به یک واحد مشخص از فضای گسسته.

با توجه به اعمال فوق در شروع باید عملیات تقسیم فضا انجام شود. تقسیم فضا گام اول در ایجاد یک تابع طبقه‌بندی کننده است. طبقه‌های مختلفی که به عنوان خروجی این عمل ایجاد می‌شوند را می‌توان مجازاً به شکل بسته‌هایی در نظر گرفت که قابلیت پذیرش یک مجموعه داده‌ی ورودی را دارا هستند. به بیان بهتر هر مجموعه داده‌ی ورودی پس از عبور از تابع تخمین زده شده‌ی f معادل یکی از این طبقه‌ها می‌باشد و به واسطه‌ی آن می‌توان داده‌ها را از هم تفکیک کرد. همان‌طور که در فصل ۲ توضیح داده شد، عرض دروازه در محیط شبیه‌سازی دوبعدی برابر با $goal\_width$ یعنی ۱۴٫۰۲m می‌باشد که از مقدار ۷٫۰۱m— تا ۷٫۰۱ برای y، به ازای $x = ۵۲/۵$ برای دروازه حریف، کشیده شده است و در صورت عبور توپ از این فضا یک گل شکل می‌گیرد. ایده‌ای که در این بخش برای تقسیم‌بندی فضا مورد استفاده قرار گرفت تقسیم ناحیه‌ی مقابل دروازه به ۲۹ قسمت، با فاصله‌ی ۵۰cm، بود(شکل ۵—۱). این تقسیم‌بندی برد تابع f را نتیجه می‌دهد که توسط آن می‌توان با وارد کردن داده‌های آموزشی، به یک طبقه که متناظر با یک نقطه از تقسیم‌بندی دروازه است رسید. مطابق شکل ۵—۱ هر پیکان s_i نماینده‌ی یک واحد تقسیم‌بندی است و عمل شوت به سمت دروازه در هر داده‌ی آموزشی برای تمام این پیکان‌ها اعمال می‌شود.



شکل ۵—۱: نحوه‌ی آموزش عامل یادگیر برای عمل شوت

در بخش بعدی باید پارامترهای مؤثر در بررسی وضعیت شوت گزینش شوند. در این میان فاصله بردار مکان دروازبان تیم حریف، $\vec{p_g}$، بردار محل توپ، $\vec{p_b}$، فاصله توپ از مرکز دروازه، $dist_{cb}$، و زاویه توپ با مرکز دروازه، dir_{cb}، به عنوان پارامترهای مفید انتخاب شدند. نمونه‌ای از انتخاب این پارامترها در شکل ۵—۱ نمایش داده شده است. در این شرایط فرض شده است که بازیکن مهاجم توانایی لازم در شلیک توپ با حداکثر توان به نقطه دلخواه را داراست و برای شلیک توپ به دروازه از بیش‌ترین توان بازیکن استفاده شده است. ضمن اینکه بازیکن مهاجم دارای مهارت لازم برای رفع اثر سوء سرعت احتمالی توپ در راستایی غیر از راستای دلخواه فرض شده است. جدول ۵—۱ لیست پارامترهای تمیز داده شده برای شکل‌دهی ورودی یک داده‌ی آموزشی شوت را به همراه بیش‌ترین مقداری که هر یک از آن‌ها می‌توانند بپذیرند نشان می‌دهد.

در گام بعد باید از trainer برای ایجاد episode های آموزش استفاده کرد. هر episode معادل با

جدول ۵—۱: پارامترهای انتخاب شده برای آموزش مهارت شوت

پارامتر	مقدار بیشینه	توضیح
p_{gx}	۵۲٫۵	بیشترین مقداری که دروازبان حریف می‌تواند روی محور x داشته باشد
p_{gy}	۳۴	بیشترین مقداری که دروازبان حریف می‌تواند روی محور y داشته باشد
p_{bx}	۵۲٫۵	بیشترین مقداری که توپ می‌تواند روی محور x داشته باشد
p_{by}	۳۴	بیشترین مقداری که توپ می‌تواند روی محور x داشته باشد
$dist_{cb}$	۳۰	حداکثر مقدار فاصله‌ی مهاجم برای شوت به سمت دروازه
dir_{cb}	۱۸۰	حداکثر زاویه‌ی مهاجم برای شوت به سمت دروازه

رخداد یک حالت آموزشی است و نحوه‌ی عملکرد روبات در شرایط خاص را نشان می‌دهد. با استفاده از مجموعه داده‌های جمع آوری شده در طول n epsiode تعداد n داده‌ی آموزشی ایجاد می‌شوند. با توجه به مطالب فصل ۲، trainer یک مربی با قابلیت شبیه‌سازی حالت خاصی از بازی است به صورتی که می‌تواند به تعداد دفعات مکرر شرایط شوت به سمت دروازه را برای بازیکن مهاجم ایجاد نموده و امکان جمع آوری داده از سوی این بازیکن را فراهم کند. برای این منظور باید یک دروازبان با توانایی بالا را به عنوان حریف انتخاب نموده و از توانایی آن به منظور عامل همکار در فرآیند یادگیری استفاده برد. در جریان آموزش، trainer به تعداد نقاط تقسیم شده روی خط دروازه یعنی ۲۹ مرتبه شرایط یکسان را برای بازیکن مهاجم و دروازبان ایجاد می‌کند و در هر حالت بازیکن مهاجم به یکی از این ۲۹ نقطه شوت می‌کند. چنانچه در شوت به سمت هریک از این ۲۹ نقطه به گل برسیم آن نقطه به عنوان یکی از مجموعه نقاط ممکن برای گل زدن ثبت می‌شود. در صورت عدم موفقیت در به ثمر رسانیدن گل با آزمایش هر ۲۹ نقطه، این وضعیت به عنوان یک حالت ناموفق در به ثمر رسانیدن گل ثبت می‌شود. با پایان یافتن یک مجموعه‌ی ۲۹ تایی با وضعیت شروع یکسان برای epsiode ها و ثبت نتیجه‌ی نهایی هر یک از آن‌ها عمل جمع آوری داده برای یک وضعیت خاتمه یافته و trainer با قرار دادن بازیکن مهاجم و دروازبان در شرایط محیطی جدیدی به شکل تصادفی، جمع آوری داده برای یک مجموعه‌ی ۲۹ تایی دیگر را از سر می‌گیرد. هر مجموعه‌ی ۲۹ تایی در نهایت به عنوان یک نمونه داده‌ی ورودی در شبکه عصبی مورد استفاده قرار خواهد گرفت.

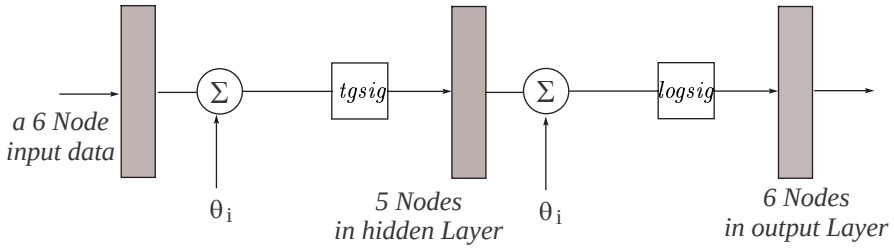
اما پس از جمع آوری داده‌های لازم مطابق آنچه در بالا به آن پرداختیم، باید عمل نگاشت داده‌های جمع آوری شده به یکی از حالات انجام شود. در صورتی که بازیکن با آزمایش هر ۲۹ نقطه در به ثمر رسانیدن گل ناتوان باشد مسئله حل شده است و وضعیت به عنوان یک حالت غیر موفق ثبت می‌شود. اما در شرایطی که بازیکن در به ثمر رسانیدن گل موفق شود سؤال این است که چه فضایی مناسب‌ترین در شلیک توپ به آن می‌باشد. مجدداً متذکر می‌شویم که طبقه‌بندی فضایی در واقع همان تقسیم فضای مقابل دروازه به ۲۹ واحد هم‌اندازه و انتخاب یک نقطه به عنوان معرف این فضا می‌باشد. برای انتخاب نقطه مناسب مشابه با شکل ۵—۱ عمل می‌شود. همان گونه که در شکل ۵—۱ قابل ملاحظه است بین دورترین و نزدیک‌ترین نقاط از فضای گسسته، نسبت به دروازبان، که توپ در آن‌ها به دروازه وارد شده است یعنی نقاط $s_{۲}$ و $s_{۱۴}$، نقطه‌ی میانی که $s_{۸}$ است انتخاب می‌شود و این نقطه با تبدیل به یکی از مجموعه نقاط در فضای گسسته به عنوان هدف مناسب برای شیلک توپ به آن در نظر گرفته می‌شود. به بیان بهتر با استفاده از این روش می‌توان دورترین نقطه از دست‌های دروازبان را که کم‌ترین احتمال برای خارج شدن از دروازه را داراست به عنوان نقطه‌ی هدف در شوت یافت.

۵—۱—۲ ایجاد شبکه عصبی مناسب

با تشخیص داده‌های ورودی، مشخص نمودن طبقه‌بندی داده‌های خروجی و جمع آوری داده‌های آموزشی مناسب، امکانات لازم را برای ایجاد شبکه عصبی فراهم آورده‌ایم. حال باید شبکه عصبی را ساخت.

با توجه به پارامترهای ورودی که در جدول ۵—۱ به آن‌ها اشاره کردیم می‌توان حدس زد که شبکه ۶ گره برای پارامترهای ورودی داراست. این ۶ ورودی پیش از وارد شدن به شبکه باید نرمال شوند که برای این منظور مقدار جمع آوری شده از طریق trainer را بر مقدار بیشینه در جدول ۵—۱ تقسیم می‌کنیم و مقادیر نرمال شده را به دست می‌آوریم.

مهم‌ترین بخش در ایجاد شبکه ساخت تقسیم‌بندی خروجی است که برای ایجاد آن از مراجع [۱، ۵]



شکل ۵—۲: مدل شبکه عصبی استفاده شده در آموزش شوت

ایده برداری شد. گفتیم که در صورت عدم ورود توپ به دروازه امکان شوت وجود ندارد و لذا یک نود[۱] را برای بررسی این امکان در نظر می‌گیریم. چنانچه توپ به دروازه وارد نشود این نود و سایر نودها در هنگام ورود داده‌ی آموزشی صفر می‌گردند و در صورتی که توپ به دروازه وارد شود این نود مقدار یک را می‌گیرد. اما برای تشخیص این که کدام نقطه برای شلیک مناسب است باید حالت مورد نظر را که یکی از ۲۹ حالت می‌باشد و در بخش قبل روش انتخاب آن توضیح داده شد انتخاب کنیم. برای این کار این ۲۹ حالت را به یک فضای باینری نگاشت می‌کنیم یعنی هریک از حالات را با یک عدد در مبنای دو نمایش می‌دهیم. به این ترتیب برای تفکیک حالات به ۵ نود نیاز داریم که با در نظر گرفتن گره تشخیص موفقیت جمعاً ٦ گره خروجی می‌باشد. با توجه به این تقسیم بندی اگر به عنوان مثال توپ در نقطه‌ی هجدهم از تقسیم بندی به دروازه وارد شود داده‌ی آموزشی خروجی به شکل ۱۰۰۱۰۱ بوده و اگر در نقطه دهم وارد دروازه شود تقسیم‌بندی به صورت ۰۱۰۱۰۱ می‌باشد که اولین ۱ در سمت راست موفقیت در ورود توپ به دروازه را نمایش داده و ۵ رقم بعدی شماره‌ی حالت را مشخص می‌کنند.

شبکه مورد نظر یک شبکه دو لایه با یک لایه پنهان در نظر گرفته شد که در این لایه تعداد ۵ گره به عنوان نودهای لایه پنهان به کار رفتند. تابع فعال‌سازی که لایه‌ی ورودی را به لایه‌ی پنهان متصل می‌کند $logsigmoid$ و تابع فعال‌سازی اتصال لایه‌ی پنهان به لایه‌ی خروجی $tgsigmoid$ می‌باشد. در نهایت الگوریتم BackPropagate به عنوان الگوریتم یادگیر به شبکه اعمال شد. شکل ۵—۲ نمایش این شبکه است.

در نهایت پس از ایجاد شبکه می‌توان داده‌های آموزشی جمع آوری شده را به شبکه اعمال کرد. این عمل تا زمانی ادامه پیدا می‌کند که بیشترین مقدار برای least mean square که در بخش ۴—۲—۲ به آن اشاره کردیم از یک مقدار δ کمتر شود.

۵—۱—۳ نتایج استفاده از شوت فرا گرفته شده

با پایان یافتن عمل آموزش در شرایطی که در جمع آوری داده‌ها و اِعمال الگوریتم مشکلی وجود نداشته باشد می‌توان از شبکه به عنوان ابزاری برای تعیین میزان درستی عملکرد شوت استفاده کرد. در این حال مقدار خروجی از گره اول با توجه به خاصت شبکه عصبی می‌تواند به عنوان میزان احتمال ورود توپ به دروازه مورد استفاده قرار بگیرد. در صورتی که این مقدار از احتمال لازم برای شلیک به سمت دروازه یعنی p بزرگتر بود، چنانچه روی ۵ گره دیگر، یک تابع f با شرایط زیر اعمال کنیم، می‌توانیم شماره‌ی طبقه‌ای که

[۱] Node

برای شوت مناسب است و نتیجتاً نقطه مناسب را به دست آوریم.

$$f(x) = \begin{cases} ۰ & if\ x < ۰/۵ \\ ۱ & if\ x \geq ۰/۵ \end{cases}$$

پس از اعمال این الگوریتم و اطمینان از عملکرد درست آن این روش یادگیری با مدل محاسباتی مشابه که در بخش ۳—۴—۵ به آن پرداختیم مورد مقایسه قرار گرفت و جدول ۵—۲ از مقایسه این دو الگوریتم حاصل شد.

جدول ۵—۲: مقایسه‌ی الگوریتم شوت محاسباتی با شوت یادگیر

الگوریتم	تعداد آزمایش‌های تصادفی در ۸ بازی	تعداد موفقیت	درصد موفقیت
محاسباتی	۲۳	۱۴	٪۶۰/۸
یادگیر	۵۴	۴۱	٪۷۵/۹

با توجه به نتایج حاصل از جدول فوق می‌توان دریافت که استفاده از الگوریتم یادگیری در رسیدن به یک مدل موفق‌تر نسبت به الگوریتم محاسباتی سودمند خواهد بود. در جدول ۵—۲ درصد موفقیت در الگوریتم یادگیر نسبت به مورد مشابه محاسباتی حدود ۷٪ بیشتر است که استفاده از این الگوریتم را در کد روبات ارجح می‌کند. ضمن این که در شوت محاسباتی برای اقدام به شوت باید مجموعه‌ای از قوانین صادق باشند و انتخاب آن‌ها چندان ساده نیست به شکلی که می‌تواند تعداد ضربات شوت را کم یا بیش از حد زیاد کند حال آن که در شوت یادگیر مقدار گره اول از شبکه در هر مرحله می‌تواند معیار مناسبی برای اقدام به شوت باشد. این تفاوت در تعداد شوت‌ها با توجه به مقادیر جدول ۵—۲ نیز قابل تصدیق است به طوری که تعداد شوت‌های یادگیر حدوداً دو برابر تعداد شوت‌های محاسباتی است.

اما آنچه در آموزش این شبکه به عنوان یک معضل مطرح است زمان زیادی است که برای آموزش شبکه صرف می‌شود چرا که در هر حالت بازیکن باید به ۲۹ مکان متفاوت شوت کند و سپس داده‌ها را جمع‌آوری نموده و نقطه‌ی هدف را استنباط کند که مجموعه‌ی این اعمال برای جمع‌آوری تعداد زیادی داده بسیار زمان‌بر بوده و موجب بروز مشکل می‌شود. به علاوه عامل مؤثر در گرفتن توپ از سوی تیم حریف در این الگوریتم تنها دروازبان می‌باشد و چنانچه بازیکنان حریف درون دروازه باشند در این الگوریتم از وجود آن‌ها صرف‌نظر می‌شود لذا ممکن است در مواردی که مدافعین تیم مخالف در دروازه قرار گرفته‌اند شوت موفق عمل نکند.

۵—۲ استفاده از درخت تصمیم در مهارت پاس

پاس به عنوان یک مهارت در حیطه‌ی تیمی مطرح می‌شود و آموزش یک عملکرد تیمی به روبات‌ها نسبت به عملکرد فردی به مراتب سخت‌تر است چرا که برای استحصال هدف باید مجموعه‌ای از عوامل درک درست از محیط پیدا نموده و با یکدیگر همکاری نمایند. نمونه‌ای ساده از پیاده‌سازی محاسباتی پاس در بخش ۳—۴—۳ مورد بررسی قرار گرفت اما همان‌طور که در آن بخش اشاره شد این الگوریتم محاسباتی از قابلیت اطمینان بالایی برخوردار نیست و لذا استفاده از یک مدل یادگیر در این شرایط بسیار سودمند می‌باشد.

در این قسمت الگوریتم پاس استفاده شده در تیم روبوسینا را مورد بررسی قرار می‌دهیم. این الگوریتم از درخت‌های تصمیم و الگوریتم ID3 برای برای طبقه‌بندی حالات استفاده می‌کند. با توجه به ویژگی‌هایی که در بخش ۴—۱—۴ در مورد نرم‌افزار C4.5 توضیح داده شد، استفاده از این نرم‌افزار برای تقسیم‌بندی فضایی مشابه با آن‌چه در سیستم کارگزار ایجاد شده است و نمونه‌ای از یک فضای پیوسته است سودمند می‌باشد.

۵—۲—۱ تعریف و ایجاد محیط مسئله

در این مسئله هدف ایجاد تابعی به نام f به گونه‌ای است که چنانچه شرایط محیطی نسبت به یک بازیکن هدف به عنوان پارامترهای ورودی به آن اِعمال شوند، f بتواند احتمال دریافت توپ توسط این بازیکن را تخمین بزند.

ایده‌ی ایجاد این روش یادگیری از [۱۰] برداشته شد. به منظور تبدیل مسئله به یک مسئله قابل حل با درخت تصمیم باید مجموعه فرآیندهای زیر انجام شوند:

۱) شناسایی مجموعه‌ی داده‌ای مهم در آموزش درخت تصمیم.

۲) استفاده از trainer با هدف ایجاد مکرر شرایط پاس برای مجموعه‌ای از بازیکنان در زمین بازی و جمع‌آوری و سازمان‌دهی مجموعه داده‌های جمع‌آوری شده.

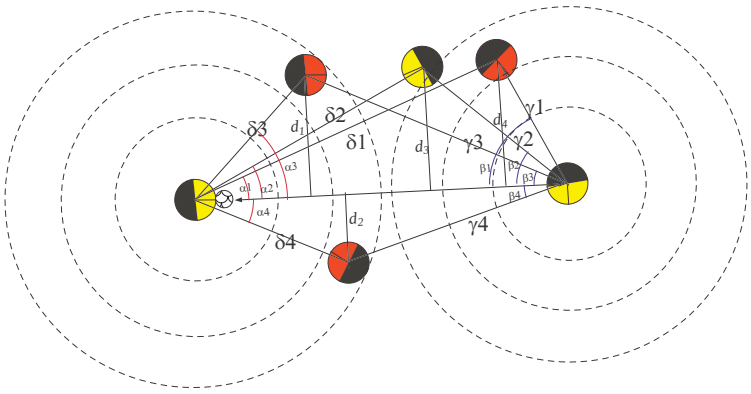
با مقایسه‌ی شرایط بالا با شرایط عنوان شده در ایجاد شبکه عصبی مشاهده می‌کنیم که در این بخش از ایجاد الگوی تقسیم فضای پیوسته به فضای گسسته بی‌نیاز هستیم و این عمل توسط نرم‌افزار C4.5 به طور خودکار انجام خواهد شد.

پارامترهایی که در ایجاد یک پاس موفق مؤثر هستند تماماً وابسته به عوامل تأثیر گذار روی انتقال توپ از فرستنده به گیرنده و یا به بیان بهتر وابسته به مسیر حرکت توپ می‌باشند. در هنگام آموزش و جمع‌آوری داده باید به این مطلب توجه شود که هدف در یک پاس درست رسیدن توپ بدون هیچ مانع به مکان مورد نظر است، نه به بازیکن مورد نظر یا بازیکنان هم تیم. با توجه به این مطالب می‌توان عوامل مؤثر در رسیدن توپ به دروازه را عواملی نظیر فاصله با نقطه‌ی هدف و نیز زاویه‌ی نسبی با این نقطه دانست. ضمن این که فاصله و زاویه‌ی تمام بازیکنان خودی و غیر خودی به استثناء بازیکن هدف، که می‌توانند در عبور توپ ایجاد مشکل نمایند، نسبت به خط حرکت از فرستنده به گیرنده از پارامترهای مهم می‌باشند. فاصله‌ی سایر بازیکنان از بازیکن فرستنده و گیرنده نیز از پارامترهای مؤثر دیگر در ارسال توپ می‌باشد. در نهایت اطلاعات آماری از زمین بازی مانند تعداد بازیکن در فواصل مشخص از فرستنده و گیرنده نیز می‌توانند به عنوان بخش دیگری از پارامترهای لازم در جمع‌آوری داده‌ی آموزشی مطرح شوند. در جدول ۵—۳ لیست پارامترهای مورد نیاز برای آموزش بازیکن و ایجاد مهارت در یادگیری پاس آمده‌اند.

قسمت دوم ایجاد فضای مناسب آموزشی برای ارسال پاس می‌باشد. در این بخش trainer باید مجموعه اعمالِ آرایش تعداد مناسبی از بازیکنان در زمین بازی، انتخاب بازیکن دریافت کننده و نیز بازیکن ارسال کننده پاس و در نهایت کنترل شروع و پایان هر episode آموزشی را انجام دهد. شروع یک episode زمانی است که بازیکن به توپ شلیک می‌کند و پایان آن با دریافت توپ از سوی بازیکن هدف، یا سایر بازیکنان و یا خروج از زمین می‌باشد. در حالتی که توپ به بازیکن هدف برسد یک حالت موفق از دریافت توپ شکل گرفته، حال آن که در صورت بروز هر وضعیت دیگری پاس ناموفق بوده و به عنوان یک داده‌ی

جدول ۵—۳: پارامترهای مورد استفاده در آموزش پاس

- فاصله و زاویه با بازیکن گیرنده:
  - ⋆ فاصله‌ی فرستنده از گیرنده: `_senderReceiverDist`
  - ⋆ زاویه‌ی فرستنده از گیرنده: `_passAngle`
- تعداد سیکل‌هایی که از آخرین مرتبه‌ی رؤیت بازیکن هدف گذشته: `_pointConf`
- فاصله و زاویه بازیکن فرستنده از سایر بازیکنان هم‌تیم که بر حسب زاویه از فرستنده مرتب شده‌اند:
  - ⋆ مرتب کردن ۹ بازیکن هم‌تیم دیگر در صورت وجود آن‌ها در World Model بر حسب زاویه‌ای که با فرستنده ایجاد می‌کنند
  - ⋆ ذخیره‌ی `teammatesFromPasser[i].dist` برای بازیکن i
  - ⋆ ذخیره‌ی `teammatesFromPasser[i].dir` برای بازیکن i
- فاصله و زاویه از بازیکنان تیم مقابل که بر حسب زاویه از دریافت کننده مرتب شده‌اند:
  - ⋆ مرتب کردن ۱۱ بازیکن حریف در صورت وجود آن‌ها در World Model بر حسب زاویه‌ای که با فرستنده ایجاد می‌کنند
  - ⋆ ذخیره‌ی `opponentsFromPasser[i].dist` برای بازیکن i
  - ⋆ ذخیره‌ی `opponentsFromPasser[i].dir` برای بازیکن i
- فاصله و زاویه‌ی بازیکن گیرنده از سایر بازیکنان هم‌تیم که بر حسب زاویه از گیرنده مرتب شده‌اند:
  - ⋆ مرتب کردن ۹ بازیکن هم‌تیم دیگر در صورت وجود آن‌ها در World Model بر حسب زاویه‌ای که با گیرنده ایجاد می‌کنند
  - ⋆ ذخیره‌ی `teammatesFromReceiver[i].dist` برای بازیکن j
  - ⋆ ذخیره‌ی `teammatesFromReceiver[i].dir` برای بازیکن j
- فاصله و زاویه‌ی بازیکن گیرنده از بازیکنان تیم حریف که بر حسب زاویه از گیرنده مرتب شده‌اند:
  - ⋆ مرتب کردن ۱۱ بازیکن حریف در صورت وجود آن‌ها در World Model بر حسب زاویه‌ای که با گیرنده ایجاد می‌کنند
  - ⋆ ذخیره‌ی `opponentsFromReceiver[i].dist` برای بازیکن j
  - ⋆ ذخیره‌ی `opponentsFromReceiver[i].dir` برای بازیکن j
- آمار جمع‌آوری شده برحسب توزیع بازیکنان در زمین بازی برای بازیکن گیرنده و فرستنده برای فواصل ۸، ۱۲، ۲۴ و زاویه‌های ۴، ۸، ۱۲ و ۲۴ درجه. به عنوان نمونه:
  - ⋆ تعداد بازیکنان خودی با فاصله‌ی کمتر از ۸ متر و زاویه‌ی کمتر از ۱۲ درجه با بازیکن فرستنده `passerOurDistLessThan8AngLessThan12Num`
  - ⋆ تعداد بازیکنان خودی با فاصله‌ی کمتر از ۱۲ متر و زاویه‌ی کمتر از ۲۴ درجه با بازیکن گیرنده `rcivrOurDistLessThan12AngLessThan24Num`
  - ⋆ تعداد بازیکنان حریف با فاصله‌ی کمتر از ۲۴ متر و زاویه‌ی کمتر از ۴ درجه با بازیکن فرستنده `passerOppDistLessThan24AngLessThan4Num`
  - ⋆ تعداد بازیکنان حریف با فاصله‌ی کمتر از ۲۴ متر و زاویه‌ی کمتر از ۸ درجه با بازیکن گیرنده `rcivrOppDistLessThan24AngLessThan8Num`و



شکل ۵—۳: نحوه‌ی آموزش عامل یادگیر برای عمل پاس

آموزشی ناموفق ثبت می‌شود. بنابراین داده‌های آموزشی ورودی در دو کلاس Success و Fail طبقه‌بندی می‌شوند.

در شکل ۵—۳ نمونه‌ای از روش جمع‌آوری داده برای تعدادی از بازیکنان نمایش داده شده است. دوایر هم مرکز با مرکزیت بازیکن فرستنده و گیرنده به ترتیب تقسیم‌بندی های مرزی به منظور جمع آوری داده از تعداد بازیکنان که در این فواصل قرار دارند را نشان می‌دهد و می‌توانند به عنوان بخشی از پارامترهای آماری در داده‌ی آموزشی استفاده شود. همچنین α_i و β_i به ترتیب زاویه‌ی بازیکن i از فرستنده و دریافت کننده‌ی توپ را نشان می‌دهند و δ_i و γ_i به ترتیب فاصله‌ی بازیکن فرستنده و بازیکن گیرنده از سایر بازیکنان i در زمین بازی را نشان می‌دهند.

همان طور که اشاره شد هدف رسیدن توپ به نقطه‌ی مورد نظر است، برای این منظور بازیکن هدف باید ثابت بایستد و در صدد قطع توپ بر نیاید. در مرحله جمع آوری داده به این صورت عمل شد که تعداد متفاوتی از بازیکنان در زمین بازی قرار داده شدند و عملیات یادگیری با استفاده از مجموعه‌های با تعداد متفاوت از بازیکنان هم‌تیم و غیر هم‌تیم به دفعات مکرر انجام شد. از ویژگی‌های C4.5 کارکردن با داده‌های آموزشی است که فاقد بعضی از خصوصیات هستند(۴—۱—۴) که این ویژگی به بازیکن آزادی عمل در استفاده از یک مجموعه اطلاعات غیر کامل را می‌دهد. یعنی در هر داده‌ی آموزشی بازیکن مجبور نیست تا تمام خصوصیات را جمع آوری نماید و این ویژگی در روبات‌های فوتبالیست با دید جزئی بسیار مفید است. با شروع عمل آموزش ابتدا بازیکن هدف به شکل تصادفی از میان مجموعه‌ی بازیکنان هم‌تیم حاضر در زمین انتخاب می‌شود و سپس پارامترهای لازم توسط بازیکن فرستنده جمع آوری می‌شوند. با اتمام جمع آوری داده‌ها توپ فرستاده می‌شود و سایر بازیکنان غیر از بازیکن هدف نسبت به وظیفه‌مندی خود در گرفتن و یا رها کردن توپ عمل می‌کنند و چنانچه توپ به منطقه‌ی kickable هریک از بازیکنان وارد شود یا از زمین بازی خارج شود و یا تعداد سیکل زمانی معین و نسبتاً زیادی صرف شده و هیچ بازیکنی توپ را به تصاحب در نیاورد (به عنوان مثال ۱۰۰ سیکل) trainer بازی را قطع می‌کند.

۵—۲—۲ ساخت درخت تصمیم

با توجه به مطالب بخش ۴—۱—۴ برای استفاده‌ی C4.5 از داده‌های جمع‌آوری شده توسط عامل باید دو فایل اصلی ایجاد شوند که این دو فایل را pass.names و pass.data می‌نامیم. فایل نخست طبقه‌بندی نهایی که همان Fail یا Success می‌باشد و نیز عناوین تمام پارامترهای جمع‌آوری شده را با توجه به جدول ۵—۳ در بر دارد. در فایل دوم هم داده‌های جمع‌آوری شده در هنگام آموزش قرار می‌گیرند. با اعمال این دو فایل به نرم‌افزار C4.5 فایل‌های pass.tree، pass.unpruned، و pass.test ایجاد می‌شوند. استفاده از درخت تصمیم ایجاد شده در داخل کد برنامه می‌تواند نیاز برای استفاده از یک الگوریتم یادگیر را مرتفع کند. شکل ۵—۴ نمونه‌ای از پیاده‌سازی نهایی برای درخت تصمیم پاس را که توسط نرم‌افزار C4.5 و با استفاده از الگوریتم ID3 ایجاد شده است نمایش می‌دهد.

در نهایت و پس از ایجاد درخت تصمیم، با استفاده از تابع Φ برای یک بازیکن مانند x و مجموعه شرایط محیطی y_x برای این بازیکن، می‌توان احتمال تصاحب توپ توسط x را به دست آورد. Φ تابعی است که متغیرهای x و y_x را به عنوان ورودی دریافت می‌کند و با حرکت روی درخت تصمیم در نهایت احتمال را در برد [۱—۰] محاسبه می‌کند. برای انجام یک پاس موفق به x می‌توان به شکل تابع Υ عمل نمود:

$$\Upsilon(x) = \begin{cases} PassTo(x) & if\ \Phi(x, y_x) \geq \delta \\ \\ ignorePass & if\ \Phi(x, y_x) < \delta \end{cases}$$

که در آن δ مقدار احتمال مناسب برای ارسال پاس بوده و بسته به شرایط در لایه‌ی تصمیم مقداردهی می‌شود.

۵—۲—۳ نتایج استفاده از پاس فراگرفته شده

برای بررسی نحوه‌ی عملکرد الگوریتم پاس مذکور تعداد ۱۴۰۸ تصمیم برای ارسال پاس در هشت بازی بررسی شد و در این میان تعداد پاس‌های موفق مورد شمارش و بررسی قرار گرفت. جدول ۵—۴ تعداد پاس‌های موفق را در شرایط استفاده از مقادیر احتمال مختلف نشان می‌دهد. این مقدار احتمال که خروجی تابع Φ می‌باشد همان‌طور که اشاره شد بسته به شرایط مختلف بازی در لایه‌ی تصمیم انتخاب می‌شود.

جدول ۵—۴: تعداد پاس‌های موفق با بررسی احتمالات مختلف

احتمال برای Φ	تعداد در ۸ بازی	تعداد موفقیت	درصد موفقیت
٪۸۵—۱۰۰	۹۱۲	۷۵۳	٪۷۸/۳
٪۷۰—۸۵	۳۲۸	۱۸۷	٪۵۷/۰۱
٪۵۵—۷۰	۱۶۸	۹۲	٪۵۴/۷

تفاوتی که در جدول بین تخمین احتمال در درخت و درصد موفقیت حقیقی وجود دارد عموماً ناشی از مشکلاتی است که در هنگام آموزش روبات از آن‌ها چشم‌پوشی شده است. به عنوان نمونه زاویه‌ی بدن دریافت کننده‌ی توپ در هنگام آموزش لحاظ نشده است و فرض بر این است که بدن بازیکن گیرنده همواره به سمت توپ است حال آن که در بازی حقیقی در بسیاری از موارد این مسئله صادق نیست. دید جزئی از محیط نیز چالش بزرگ دیگری است که در ارسال پاس پیش از جمع‌آوری داده‌های کافی اثر منفی به سیستم تحمیل می‌کند.

```
---------------------------------------
Decision Tree:

passerAllDistLessThan24AngLessThan24Num <= 0
|    senderReceiverDist_ <= 20.4369
|    |    rciverOppDistLessThan24AngLessThan24Num <= 0 Success (1536.0/179.2)
|    |    rciverOppDistLessThan24AngLessThan24Num > 0
|    |    |    passAngle_ <= -66.1046
|    |    |    |    passerOppDistLessThan12AngLessThan8Num <= 0  Success (178.0/90.1)
|    |    |    |    passerOppDistLessThan12AngLessThan8Num > 0 Fail (15.0/6.8)
|    |    |    passAngle_ > -66.1046
|    |    |    |    senderReceiverDist_ <= 16.1679
|    |    |    |    |    rciverOurDistLessThan24AngLessThan8Num <= 0 Success (608.0/99.8)
|    |    |    |    |    rciverOurDistLessThan24AngLessThan8Num > 0
|    |    |    |    |    |    rciverAllDistLessThan12AngLessThan4Num > 0 Fail (3.0/1.1)
|    |    |    |    |    |    rciverAllDistLessThan12AngLessThan4Num <= 0
|    |    |    |    |    |    |   pointConf_ <= -4 subTree1
|    |    |    |    |    |    |    pointConf_ > -4 subTree2
|    |    |    |    senderReceiverDist_ > 16.1679
|    |    |    |    |    rciverAllDistLessThan24AngLessThan4Num <= 0
|    |    |    |    |    |    opponentsFromPasser[1].dir <= -49.6407
|    |    |    |    |    |    |    pointConf_ <= -1 Fail (19.0/4.8)
|    |    |    |    |    |    |    pointConf_ > -1 Success (3.0/1.1)
|    |    |    |    |    |    opponentsFromPasser[1].dir > -49.6407
|    |    |    |    |    |    |    passAngle_ <= 94.1748 Success (180.0/30.0)
|    |    |    |    |    |    |    passAngle_ > 94.1748
|    |    |    |    |    |    |    |    opponentsFromPasser[1].dir <= -3.01519 Fail (10.0/2.4)
|    |    |    |    |    |    |    |    opponentsFromPasser[1].dir > -3.01519 Success (4.0/1.2)
|    |    |    |    |    rciverAllDistLessThan24AngLessThan4Num > 0 subTree3
|    senderReceiverDist_ > 20.4369
|    |    senderReceiverDist_ <= 24.5858
|    |    |    opponentsFromPasser[1].dist > 23.0835 Fail (175.4/57.4)
|    |    |    opponentsFromPasser[1].dist <= 23.0835
|    |    |    |    rciverAllDistLessThan24AngLessThan4Num > 0 Fail (57.0/25.1)
|    |    |    |    rciverAllDistLessThan24AngLessThan4Num <= 0
|    |    |    |    |    passAngle_ <= -78.1088 Fail (57.4/23.7)
|    |    |    |    |    passAngle_ > -78.1088 Success (151.2/39.7)
|    |    senderReceiverDist_ > 24.5858
|    |    |    opponentsFromPasser[1].dir <= 22.9849 Fail (171.2/11.8)
|    |    |    opponentsFromPasser[1].dir > 22.9849
|    |    |    |    rciverOppDistLessThan24AngLessThan24Num <= 0 Fail (35.8/8.2)
|    |    |    |    rciverOppDistLessThan24AngLessThan24Num > 0
|    |    |    |    |    rciverOurDistLessThan24AngLessThan24Num > 0 Success (7.0/2.4)
|    |    |    |    |    rciverOurDistLessThan24AngLessThan24Num <= 0
|    |    |    |    |    |    opponentsFromReceiver[8].dir <= -82.925 Success (8.8/2.2)
|    |    |    |    |    |    opponentsFromReceiver[8].dir > -82.925
|    |    |    |    |    |    |    teammatesFromPasser[1].dir <= 104.414 Fail (8.4/2.6)
|    |    |    |    |    |    |    teammatesFromPasser[1].dir > 104.414 Success (4.8/1.2)
```

شکل ۵—۴: بخشی از درخت تصمیم ایجاد شده بوسیله‌ی نرم‌افزار

ایده‌ی اصلی این الگوریتم در [۱۰] مطرح شده است با این تفاوت که در پیاده‌سازی آن برای یک عامل روبوسینا تعداد طبقه‌بندی نهایی از ۳ حالت Success، Fail، و Miss به دو حالت Success و Fail تقلیل یافت، ضمن این‌که در مدل جمع‌آوری داده از عوامل تیم روبوسینا استفاده شد و الگوی حرکت بازیکنان به سمت توپ به شکل دیگری پیاده شد که در بخش ۵—۲—۱ به آن پرداختیم. همچنین پارامتر pointConf_ به مجموعه‌ی پارامترهای آموزش افزوده شد که با توجه به دید جزئی بازیکن از محیط تأثیر بسیار زیادی در بهینه شدن یادگیری داشت.

۵—۳ بازنگری فصل

این فصل از رساله استفاده از شبکه‌های عصبی برای استفاده در آموزش شوت به روبات و استفاده از درخت‌های تصمیم برای آموزش ارسال پاس به سایر بازیکنان را توضیح داد. استفاده از شبکه‌های عصبی و الگوریتم BackPropagate در آموزش شوت به سمت دروازه در بخش اول از این فصل مورد بررسی قرار گرفت. در این بخش با تقسیم فضای مقابله دروازه به مجموعه‌ای از کلاس‌ها و استفاده از خاصیت طبقه‌بندی در شبکه عصبی موفق به ایجاد مدلی برای شوت به سمت دروازه‌ی حریف شدیم. همچنین در بخش دوم از این فصل با استفاده از خاصیت درخت تصمیم در تفکیک حالت‌ها و استفاده از C4.5 در تبدیل مدل فضایی پیوسته به یک فضای گسسته توانستیم یک روبات را در استفاده از مهارت پاس به سایر بازیکنان هم‌تیم آموزش دهیم.

٦

پیشنهادات و نتیجه‌گیری

در این رساله روند افزایشی تحلیل، طراحی و پیاده‌سازی یک مجموعه از عوامل همکار خودکار مورد بررسی قرار گرفت. در این راستا محیط عملکرد عوامل، معماری یک عامل، روش‌های زمان‌بندی و مکان‌یابی، مجموعه‌ای از اعمال سطح بالا، و استفاده از روش‌های یادگیری در آموزش یک روبات نرم‌افزاری مورد بررسی قرار گرفت.

همان‌طور که در ابتدا اشاره شد، محیط عملکرد عوامل، یعنی Soccer Server، یک محیط خطادار و غیرقطعی می‌باشد که عامل‌ها را وادار به استفاده از الگوریتم‌های یادگیری می‌کند. اما مقایسه نزدیک بین روش‌های محاسباتی و یادگیری نشان می‌دهند که استفاده از الگوهای یادگیری به نسبت زمان زیادی که برای پیاده‌سازی آن‌ها صرف می‌شود و با توجه به این‌که دقت الگوریتم‌های محاسباتی در صورت پیاده‌سازی درست کاملاً با انواع یادگیر قابل مقایسه است، چندان نسبت به مدل محاسباتی برتر نمی‌باشند. این مسئله با توجه به تعداد تیم‌هایی که از الگوریتم‌های یادگیر استفاده می‌کنند و در مسابقات جهانی در بین ۸ تیم برتر قرار می‌گیرند نیز قابل مشاهده است. دلیل عمده در این ارتباط عدم تأثیر زیاد خطا در بسیاری از فرآیندهاست. لذا افزایش خطای سیستم و استفاده از عوامل ایجاد کننده‌ی آن در محیط شبیه‌سازی، نظیر باد، در مسابقات رسمی می‌تواند جلوه‌ی استفاده از الگوریتم‌های یادگیر را بیشتر کند.

محیط روبوکاپ یک فضای حالت بسیار بزرگ را مدل می‌کند که در آن نگاشت یک عمل موفق به یک حالت در بسیاری از موارد دشوار می‌باشد و استفاده از الگوریتم‌های یادگیر در چنین فضایی همواره با چالش جمع‌آوری داده رو به روست. استفاده از روش‌های تقریب توابع برای حالتی که نیاز به نگاشت محیط به یک تابع است بسیار سودمند می‌باشد. در این راستا شبکه‌های عصبی در ایجاد توابع Action-Value برای استفاده در یادگیری تقویتی موفق عمل می‌کنند. تیم روبوسینا در ادامه روند تصمیم‌گیری استفاده از این روش را در ایجاد یک تصمیم تیمی از سوی یک بازیکن کاپیتان در دستور کار خود دارد.

کتاب‌نامه

[۱] محمدعلی صفری قهساره. استفاده از تکنیک‌های هوش مصنوعی در شبیه‌سازی روبات فوتبالیست. پایان‌نامه‌ی کارشناسی ، دانشگاه صنعتی شریف، دانشکده‌ی مهندسی کامپیوتر، مرداد ۱۳۷۹.

[۲] حمید ضرّابی‌زاده، مصطفی رفائی جوکندان، مرتضی شعبانی، نیما کاویانی. مکان‌یابی روبات با استفاده از اطلاعات بینایی. در مجموعه مقالات دومین دوره‌ی کنفرانس ملی کامپیوتر(*NCC 2003*)، صفحات ۵۷—۵۵، مشهد، ایران، ۱۳۸۳.

[۳] H. Kitano and M. Asada. *RoboCup Humanoid Challenge: That's One Small Step for A Robot, One Giant Leap for Mankind*. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-98), 1998.

[۴] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. *RoboCup: The Robot World Cup Initiative*. In Proceedings of the First International Conference on Autonomous Agents (Agent-97), 1997.

[۵] R. de Boer, J. Kok. *The Incremental Development of a Synthetic Multi-Agent System: The UVA Trilearn 2001 Robotic Soccer Simulation Team*, Master Thesis, UVA, February 2002.

[۶] A. H. Bond and L. Gasser. *An Analysis of Problems and Research in DAI*, pages 3-35. MorganKaufmann Publishers Inc. Los Angeles, CA, 1988.

[۷] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ, 1995.

[۸] P. Stone and M. Veloso. *Multi-Agent Systems: A Survey from a Machine Learning Perspective.* Autonomous Robotics, 8(3), July 2000.

[۹] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. *RoboCup: The Robot World Cup Initiative.* In Proceedings of the IJCAI-95 Workshop on Entertainment and AI/Alife, 1995.

[۱۰] P. Stone. *Layered Learning in Multi-Agent Systems.* PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, Dec. 1998.

[۱۱] M. Chen, et al. *Soccer Server Manual.* http://sserver.sourceforge.net. June 2001.

[۱۲] M. de Burg, M. V. Kreveld, M. Overmars, O. Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer Verlag, 1997.

[۱۳] K. Mehlhorn and St. Naher. *The LEDA Platform of Combinatorial and Geometric Computing.* Cambridge University Press, 1999.

[۱۴] M. Butler, M. Prokopenko, and T. Howard. *Flexible Synchronisation within the RoboCup Environment: a Comparative Analysis.* In P. Stone, T. Balch, and G. Kraetszchmar, editors, RoboCup-2000: Robot Soccer World Cup IV, pages 119-128, Springer Verlag. Berlin, 2001.

[۱۵] W. Richard Stevens. *UNIX network programming.* Prentice Hall, 1990.

[۱۶] S. Y. Kung. *Digital Neural Networks.* Prentice Hall, Englewood Cliffs. NJ, 1993.

[۱۷] Tom M. Mitchell. *Machine Learning.* McGraw-Hill. NY, 1991.

[۱۸] G. F. Luger, W. A. Stubblefield. *Artificial Intelligence : Structures and Strategies for Complex Problem Solving.* Addison-Wesley, 1997.

[۱۹] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, San Mateo, California, 1993. http://www.rulequest.com.

[۲۰] R. S. Sutton, and A. G. Barto. *Reinforcement Learning: An Introduction.* Cambridge, MA, MIT Press, 1998.

[۲۱] A. Visser, J. Lagerberg, A. van Inge, L. Hertzberger, J. van Dam, A. Dev, L. Dorst, F. Groen, B. Kröse, and M. Wiering. *The Organization and Design of Autonomous Systems.* University of Amsterdam, Sept. 1999.

RoboSina from Scratch

Abstract

In this thesis incremental development, design and implementation of a soccer simulation team named *RoboSina* along with it's prominent aspects are reviewed. RoboSina is a soccer simulation team consists of 11 autonomous agents which interact together in a physical simulated environment called *soccer server*. The soccer server environment enables two teams of autonomous agents to play a soccer game against each other using predefined rules managed by the system. Soccer Server provides a fully distributed and real-time system in such a way that in which each of the agents must cooperate with other teammates in order to achieve the ultimate goal of winning the game. This simulator system models many aspects and properties of a real environment such as: Noise in movment of objects, Noise in sensors and actuators, limited physical abilities, and restricted communication. Our main contributions in this thesis include defining the architecture of an agent in a multi-layer model, synchronizing an agent with the simulator, new localization method for the agent, analysis and development of agent's world model using partial sensation of the environment, tactics and strategies used in defending against opponent's penetration and attacking to it's defending lines, and finally focusing on algorithms to break the defence line of the opponent and scoring a goal. Throughout the project RoboSina has participated in several internal and international competitions which it's prominent results can be summarized as : Championship of the 3^{rd} American Open RoboCup Competitions in May 2005, 2^{nd} place of the 3^{rd} Iranian Open RoboCup Competitions in April 2005, 5^{th} palce of RoboCup World Cup Competitions 2004 in Portugal, Championship of the 2^{nd} American Open RoboCup Competitions in May 2004 and Championship of the 2^{nd} Iranian Open RoboCup Competitions in March 2004.

Keywords: 1)*Soccer Robots,* 2)*Simulation System,* 3)*Soccer Server,* 4)*Artificial Neural Networks,* 5)*Decision Trees,* 6)*Reinforcement Learning,* 7)*Distributed Artificial Intelligence*

Bu-Ali Sina University

Department of Computer Engineering

Submitted in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Science
in
Software Engineering

Title

RoboSina from Scratch

Supervisor

Dr. Mir Hossein Dezfoulian

Author

Nima Kaviani, Mostafa Rafaie-Jokandan

May 2005