گزارش فنی تیم شبیه سازی دوبعدی سایرس

نادر زارع، محسن صادق پور، اشکان کشاورزی، مهتاب سرومیلی، سینا الاهی منش، آراد فیروزکوهی، محمد ابولنژاد

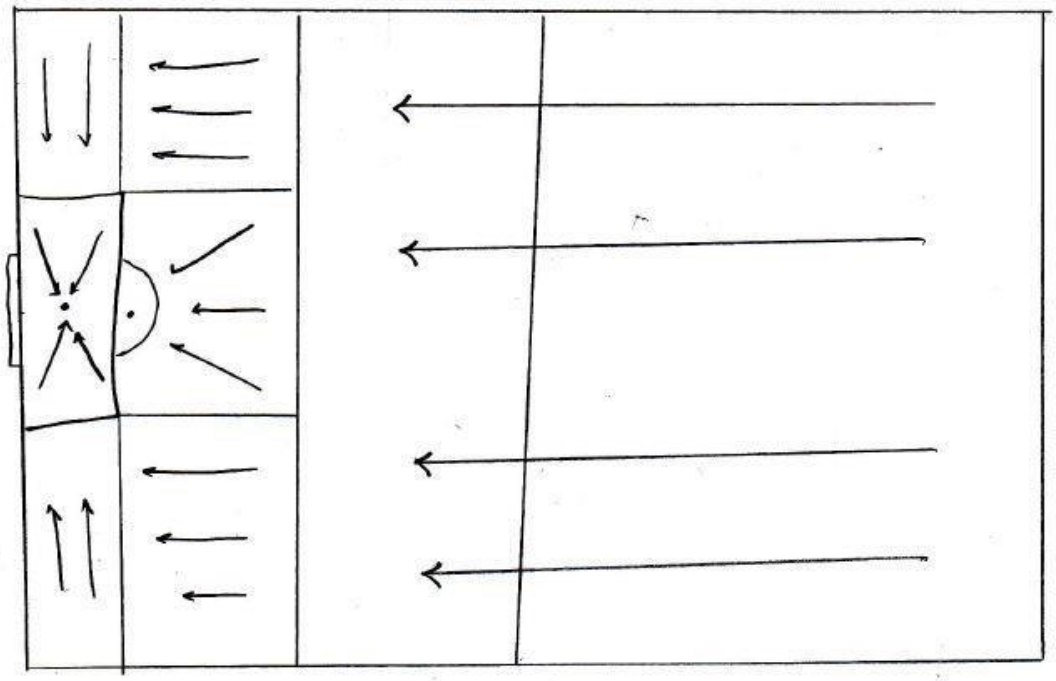
چکیده
تیم شبیه‌ساز دوبعدی سایرس در سال‌های اخیر توانسته به مسابقات جهانی و iranopen راه یافته و افتخاراتی نیز کسب کند. این گزارش شامل توضیحاتی در خصوص اهداف ، ایده‌ها و الگوریتم های مورد استفاده‌ی تیم سایرس است. در این گزارش به اختصار الگوریتم دریبل توضیح داده شده است. قابل ذکر است بیس مورد استفاده در تیم سایرس، بیس Agent 2D می باشد.

کلمات کلیدی: شبیه‌ساز دوبعدی ، سایرس ، گزارش فنی ، بلاک

مقدمه
تیم سایرس متشکل از دانشجویان دانشگاه‌های خواجه نصیرالدین طوسی، دانشگاه تهران و دانش‌آموزان دبیرستان انرژی‌اتمی است که با هدف پیشرفت علمی و تلاش در جهت پیش برد اهداف روبوکاپ فعالیت می‌کنند. با توجه به شرایط موجود، هر ساله در اعضا این تیم تغییراتی به وجود می‌آید. اعضای گذشته این تیم هر ساله با شرکت در مسابقات iranopen و روبوکاپ جهانی افتخارات زیادی کسب کرده‌اند. این تیم سعی نموده با جایگزینی رفتار های دریبل بهینه‌سازی شده و الگوریتم های مناسب‌تر رفتار دریبل را ارتقا دهد که در ادامه به بررسی آن‌ها می پردازیم.

در حالت هجومی و دفاعی یکی از مهمترین رفتار ها دریافت توپ یا بلاک می باشد که این رفتار خود به دو قسمت حرکت به سمت مقصد برای گرفتن توپ و ضربه به توپ جهت کنترل توپ یا همچنین تکل زدن در لحظه آخر اگر نتوانیم توپ را کنترل کنیم که در فصل گذشته به صورت مفصل در این باره توضیح داده شد .در این فصل ابتدا به بررسی الگوریتم های توپ گیری پرداخته و سپس مفهوم بلاک در شبیه سازی دوبعدی فوتبال را مورد بررسی قرار خواهیم داد.

بلاک به عنوان یکی از مهمترین رفتار ها در زمان دفاع شناخته میشه که قابلیت هوشمند سازی بسیار زیادی داره به این دلیل که باید برای این رفتار حرکت آینده حریف را پیش بینی کرد. در الگوریتم فرض می کنیم یک بازیکن حریف صاحب توپ هست و یک بازیکن خودی قصد داره توپ را تصاحب کند و همچنین تمایل دارد در کوتاه ترین زمان این عمل را به انجام برساند. ابتدا سعی می کنیم راهکاری برای پیش بینی حرکت حریف پیدا کنیم. برای این مورد در ساده ترین نوع الگوریتم فرض می شود حریف هدفی برای دریبل دارد و با یک سرعت متوسط به سمت آن هدف حرکت خواهد کرد. برای مثال حرکت حریف در هنگام دریبل زدن می تواند به صورت شکل زیر بسیار ساده باشد .در شکل های زیر راستا با توجه به نقطه شروع مشخص خواهد شد.



تصویر ۱ پیش بینی حرکت حریف در بلاک

اگر الگوریتم بالا را به صورت زیر پیاده سازی کنیم که ابتدا راستا مشخص شده سپس مسیر مشخص شود:

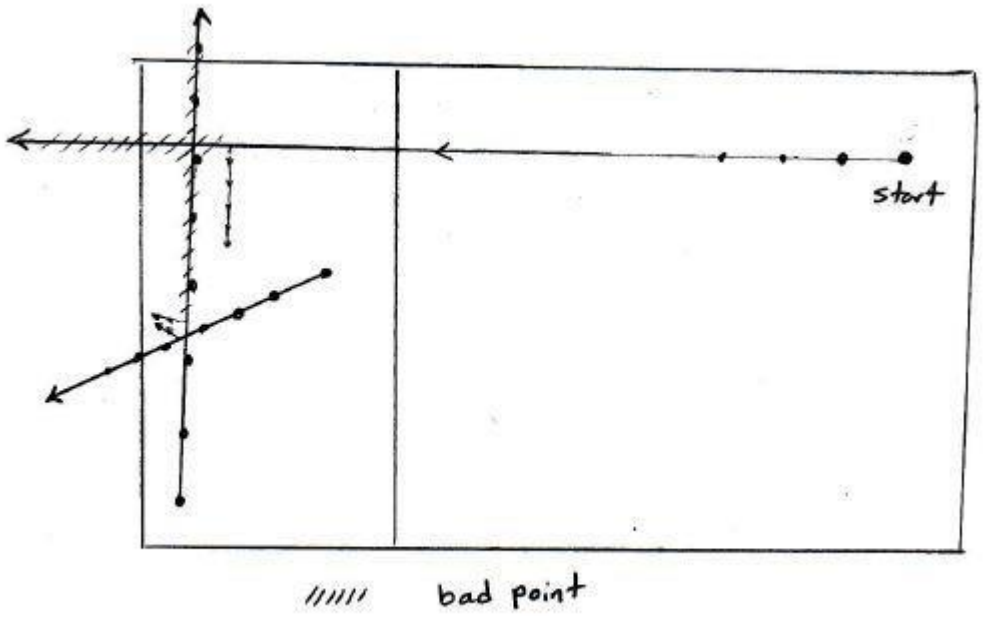
```
Set Angle()

For ( all simulation point )

{

If intercept then Go to Point
```

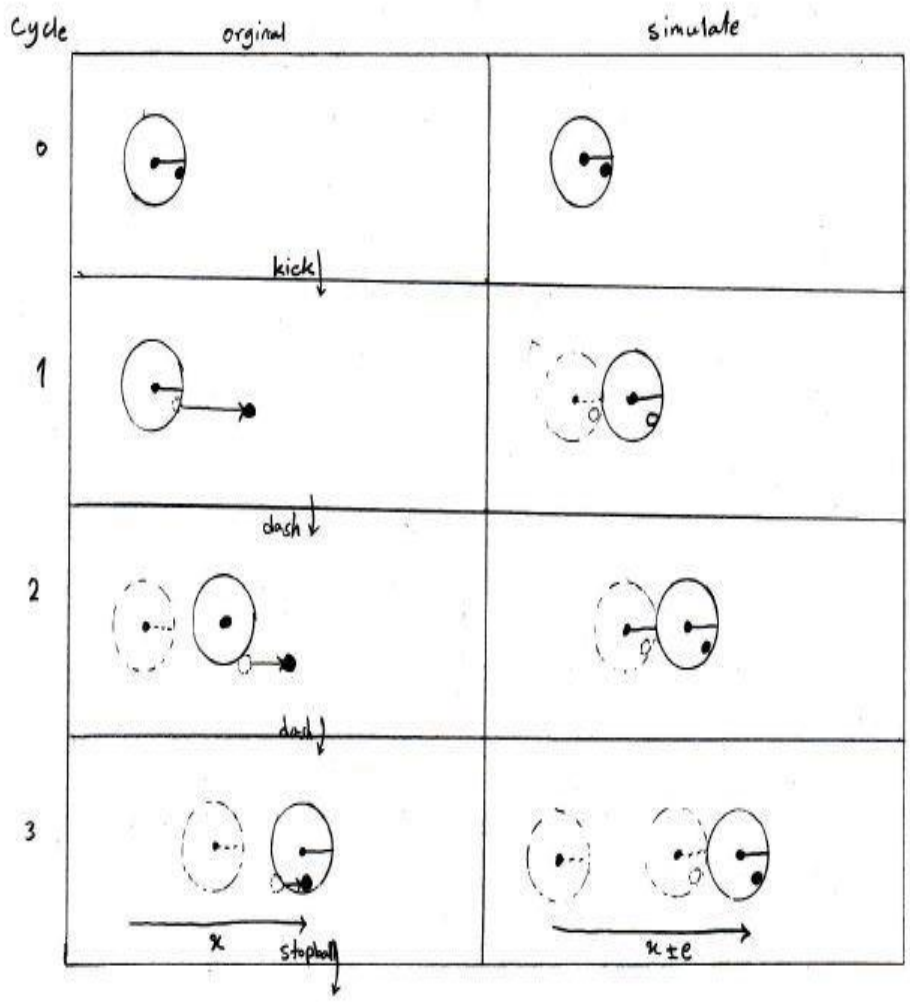
{الگوریتم به صورت شکل زیر خواهد بود زیرا قبل از حلقه راستا مشخص شده است.



تصویر ۲ اجرای الگوریتم به صورتی که اول راستا بعد مسیر مشخص شود

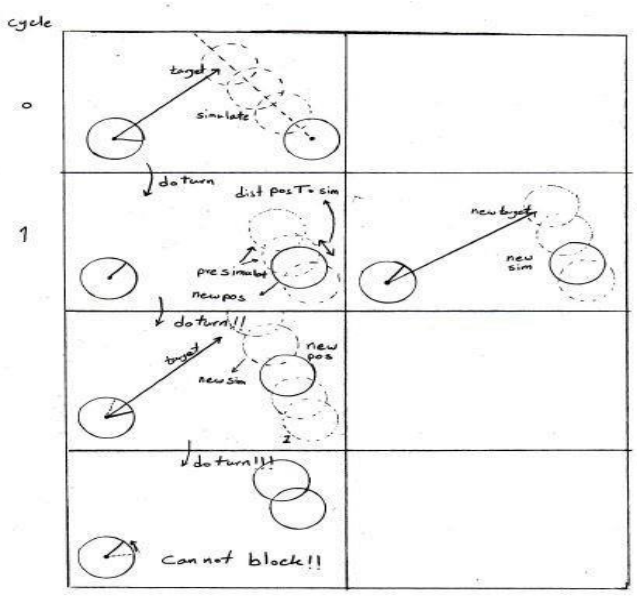
برای اینکه بازیکن متوجه شود در چه نقطه ای می تواند توپ را به تصاحب خود در بیاورد حرکت دریبل بازیکن حریف را همانند شبیه سازی حرکت توپ اما با سرعت متوسط شبیه سازی می کنیم. این شبیه سازی به هیچ عنوان همانند واقعیت نیست اما می توان این موضوع را به واقعیت نزدیک کرد. ما در شبیه سازی فرض می کنیم بازیکن حریف با توپ حرکت می کند هر چنداین فرض کاملا اشتباه است و بازیکن حریف ابتدا به توپ ضربه زده و توپ با سرعت از این بازیکن دور شده سپس که سرعت توپ کم می شود بازیکن حریف به توپ خواهد رسید. بازیکنان معمولا هنگام دریبل با بیشترین سرعت حرکت نمی کنند و همچنین یک سیکل برای ضربه زدن به توپ از دست می دهند به همین دلیل سرعت حدودی دریبل را برابر با 2.7 در نظر خواهیم گرفت.

تفاوت شبیه سازی ما با واقعیت را در شکل زیر مشاهده می کنید.



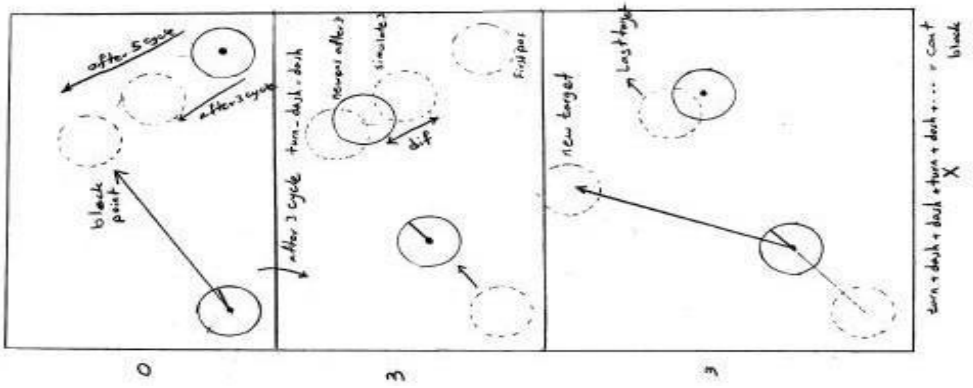
تصویر ۳ تفاوت بلاک واقعی و شبیه سازی شده

در شکل زیر یکی از مشکلات شبیه سازی اشتباه بلاک را مشاهده می کنید که بازیکن با در نظر گرفتن اشتباه حرکتحریف مجبور به چرخش های زیادی می شود و باعث اتلاف زمان به صورت کاملا محسوسی می شود. در بعضی موارد چندین سیکل صرف چرخش های بی مورد برای بلاک کردن بازیکن حریف می شود.



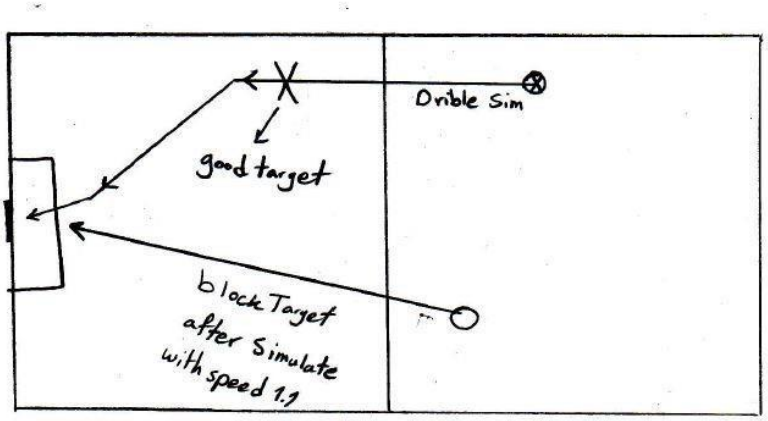
تصویر ٤ اشتباه در پیش بینی حرکت حریف در بلاک

یکی دیگر از موارد که کار را سخت تر می کند این است که عامل دفاعی در پیش بینی سرعت بازیکن حریف اشتباه کند ،این موضوع ممکن است باعث شود بازیکن دریبل بخورد زیرا بازیکن فرض می کند بازیکن حریف با سرعت متوسط 2.7 حرکت می کند و اگر این بازیکن با سرعت بیشتر حرکت کند محاسبات اشتباه خواهد بود. در شکل زیر این مورد را متوجه خواهید شد:



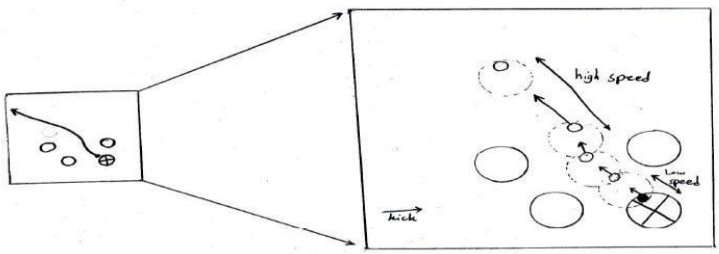
تصویر ٥ زیاد در نظر گرفتن سرعت بازیکن حریف در بلاک

اگر این سرعت را نیز زیاد در نظر بگیریم مشکل بزرگتری بوجود خواهد آمد که بازیکن بسیار عقب تر از جایی که می تواند توپ رادریافت کند مکانی را برای گرفتن توپ در نظر خواهد گرفت که در زیر قابل مشاهده است:



تصویر 6 مشکل در کمتر در نظر گرفتن سرعت بازیکن حریف در بلاک

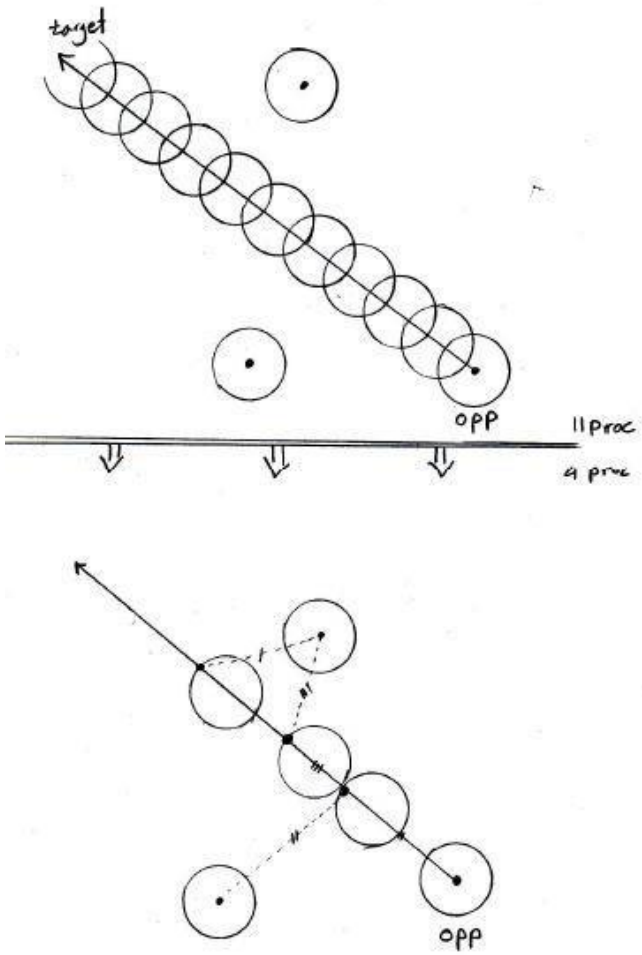
برای رفع این مشکلات راهکاری وجود دارد که در زمان های متفاوت این سرعت بیشتر در نظر گرفته شود. برای مثال هنگامی که بازیکن حریف بازیکنی در اطراف خود ندارد پس با سرعت بیشتری دریبل خواهد زد و برعکس در زمانی که بازیکنان ما اطراف بازیکن حریف هستند این بازیکن با سرعت کمتری دریبل خواهد زد، اما چرا این اتفاق رخ می دهد؟ دلیل این موضوع این می باشد که بازیکن صاحب توپ حریف هنگامی که در مکان شلوغ قصد دریبل زدن دارد نمی تواند توپ را از خودش فاصله بدهد پس مجبور است برای رسیدن به یک نقطه تعداد دفعات زیادی به توپ ضربه بزند و این موضوع باعث میشود که تعداد سیکل های بیشتری را از دست بدهد و هرچه تعداد سیکل ها برای عبور از یک مسافت ثابت بیشتر شود در نتیجه سرعت میانگین کاهش پیدا می کند. برای درک بهتر موضوع به شکل زیر توجه کنید:



تصویر 7 سرعت پیش بینی شده در وضعیت های مختلف

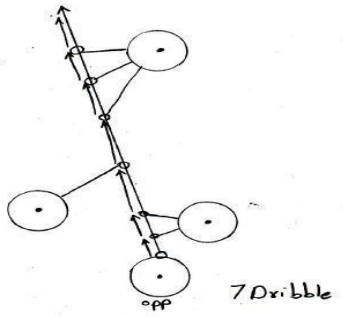
برای بهینه سازی این الگوریتم می توانیم تمام موارد ثابت که در الگوریتم قبلی بررسی شد را به صورت پویا بدست آوریم .برای مثال به جای اینکه سرعت میانگین را بدست آوریم می توانیم تعداد ضربات لازم برای به حرکت در آوردن توپ را نیز محاسبه کنیم. برای این منظور در شبیه سازی حرکت بازیکن حریف یا همان دریبل می توانیم این شبیه سازی را به تعدادی دریبل تقسیم کنیم که دارای نقطه ابتدایی یعنی همان مکانی که به توپ ضربه زده می شود و نقطه پایانی که بازیکن توپ را مجدد تصاحب کرده و این نقطه پایانی برابر با نقطه شروع دریبل بعدی می باشد.

برای بدست آوردن این نقاط فرض می کنیم بازیکن به نقاطی توپ را می تواند بفرستد که فاصله آن نقاط تا ابتدای دریبل[20] نسبت به تمام بازیکنان حریف نزدیک تر باشد. برای مثال به شکل زیر توجه کنید که نقاطی حذف شده اند:



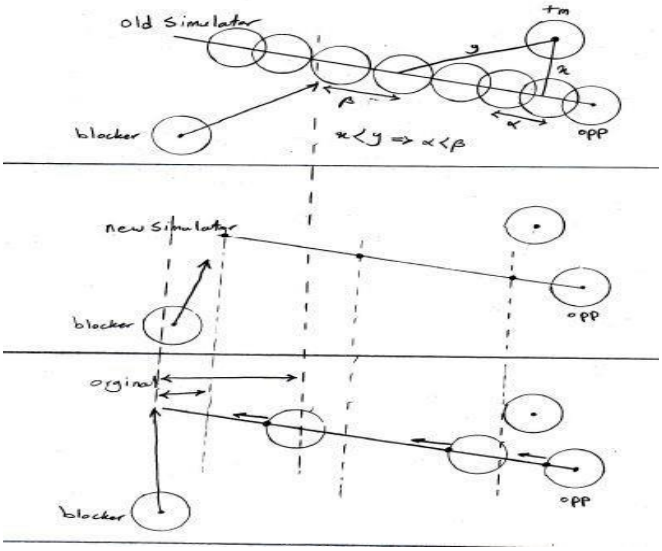
تصویر ۸ محاسبه نقاط به وسیله یک الگوریتم پویا

سپس انتهای دریبل همان نقطه ای می باشد که بازیکن به توپ ضربه زده و دریبل جدیدی را بوجود خواهد آورد. و این عمل همچنان ادامه پیدا خواهد کرد. در این الگوریتم فرض می شود بازیکن حریف همیشه با حداکثر سرعت خود حرکت خواهد کرد. در شکل زیر این نوع دریبل را مشاهده می کنید:



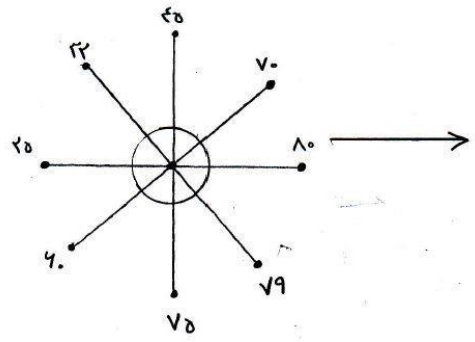
تصویر ۹ محاسبه نقاطی که بازیکن حریف به توپ ضربه میزند)یک دریبل خاص(

در این حالت دیگر احتیاجی به گرفتن میانگین نمی باشد و با شبیه سازی و ادامه فرضی حرکت دریبل نقاطی که بازیکن به توپ ضربه می زند محاسبه خواهند شد. برای مثال در شکل زیر بازیکن هدف خود را با توجه به محاسبات زمان ضربه و حرکت بدست آورده است و تفاوت را با الگوریتم سرعت میانگین متوجه می شوید.



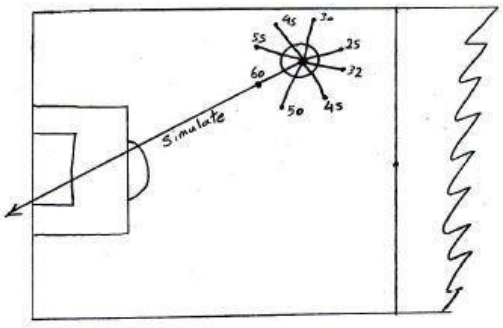
تصویر ۱۰ مقایسه بلاک در واقعیت با دو الگوریتم مطرح شده

یکی دیگر از مواردی که باید پویا شود مسیر حرکت است، می دانیم در هنگام دریبل زدن فقط زمانی بازیکن می تواند مسیرش را تغییر دهد که می خواهد به توپ ضربه بزند. برای این منظور می توانیم در اطراف بازیکن نقاطی را مانند محیط یک دایره گسسته در نظر بگیریم و نقطه ای که امتیاز بیشتری دارد را به نقطه توپ متصل نموده و این راستا [21]را مسیر دریبل در نظر بگیریم . به صورت شکل زیر:



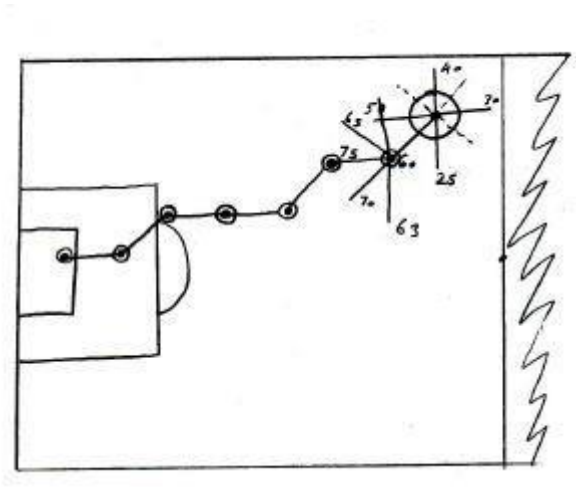
تصویر ۱۱ به دست آوردن مسیر حرکت

اما مشکلی در این قسمت وجود دارد که بازیکن در هر ضربه ممکن است مسیر خود را تغییر دهد، برای مثال الگوریتم قبل باعث می شود مسیر دریبل به صورت زیر در بیاید:



تصویر ۱۲ تغییر مسیر بازیکن در هر سیکل

پس باید در هر ضربه زدن به توپ یک بار تابع مسیر یاب راستای دریبل را مشخص نماید که مسیر حرکت همانند شکل زیر در بیاید :



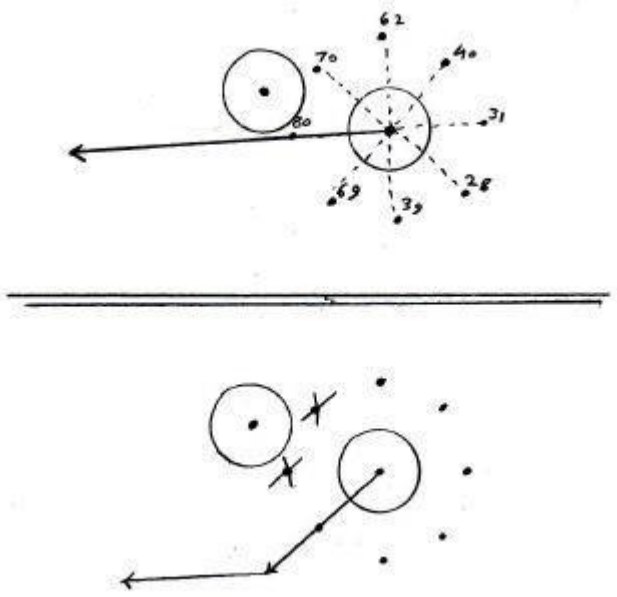
تصویر ۱۳ محاسبه مسیر با هر بار ضربه به توپ

یک مورد مهم در این قسمت تابع امتیاز دهی می باشد، برای مثال می توان تابع امتیاز دهی را معکوس فاصله نقطه تا دروازه خودمان در نظر گرفت. این تابع نسبتا جواب خوبی را به ما خواهد داد اما پیشنهاد می کنیم به جای این تابع از تابع امتیاز دهی هجومی بیس ایجنت استفاده شود زیرا بیشتر تیم های شبیه سازی دوبعدی برای حمله از این تابع استفاده می کنند که در زیر موجود می باشد:

```
double point = state.ball().pos().x;
point += std::max( 0.0,40.0 - ServerParam::i().theirTeamGoalPos().dist( state.ball().pos() )
```

این کد با توجه مقدار x توپ یا نقطه فرضی و همچنین فاصله توپ تا مرکز دروازه یک امتیاز را بوجود می آورد که تابع اصلی و کامل برای محاسبه به صورت زیر می باشد که در بیس جهت امتیاز دادن به حالت های بوجود آمده بر اساس رفتار ها استفاده می شود.

با پویا کردن سرعت حرکت و در نظر گرفتن نقاط ضربه زدن تابع بلاک به خوبی بهینه شده است اما می توان هنوز این تابع را بهینه تر نمود. برای مثال یکی از مشکلات که ممکن است بوجود آید در شکل زیر نشان داده شده است .در شکل بهترین مسیر دقیقا جایی است که بازیکن خودی ایستاده است. این مورد باعث می شود که بازیکن حریف این راستا را انتخاب نکند پس احتمالا مسیر دریبل به صورت شکل زیرین بوجود خواهد آمد:

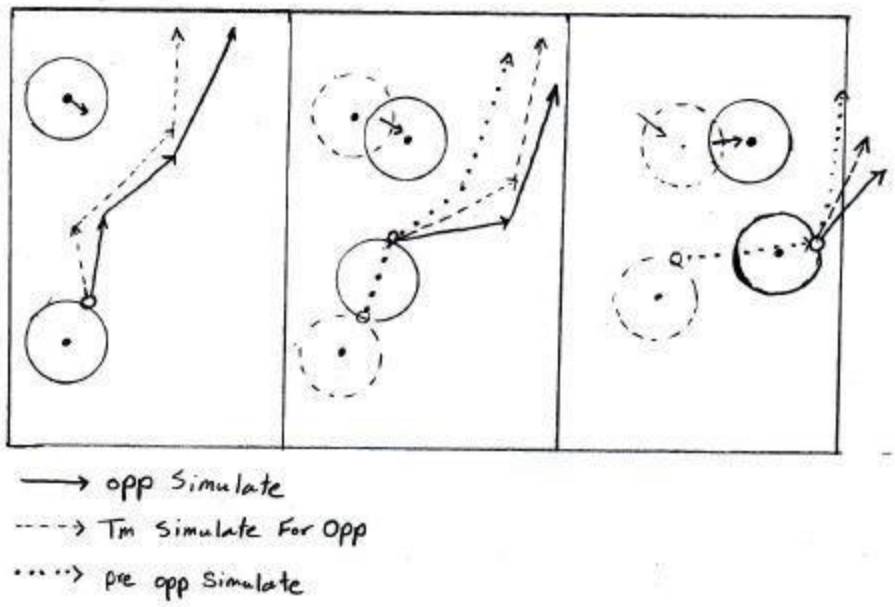


تصویر ۱٤ تغییر مسیر در صورت مزاحمت برای حریف

اما ممکن است بازیکن حریف ابتدا به بازیکن ما نزدیک شده سپس مسیر خودش را تغییر دهد که مسیر دریبل به صورت زیر خواهد شد:

برای این موضوع می توان تابع امتیاز دهی را پویا تر نمود، به این منظور می توان با توجه به فاصله نقطه تا بازیکن خودی مقداری از امتیاز نقطه را پایین آورد، این نوع امتیاز دهی به احتمال زیاد بسیار نزدیک به واقعیت خصوصا تیم های قدرت مند می باشد.

راههایی برای بهتر نمودن این الگوریتم وجود دارد، برای مثال می توان با استفاده از اطلاعات برداری در زمان بازی به صورت هوشمند دریبل حریف را با توجه به دریبل های قبلی پیش بینی کرد، البته این نکته نیز وجود دارد که هر چه ما پیش بینی دقیق تری از حرکت و دریبل حریف داشته باشیم حریف نیز رفتار خود را به صورت پویا تغییر خواهد داد. برای مثال در شکل زیر بازیکن حریف با توجه به رفتار بلاک بازیکن ما حرکت دریبل خود را تغییر می دهد.



تصویر ۱۵ تغییر حرکت بازیکنان حریف به دلیل رفتار دفاعی بازیکنان خودی

References

1. Training a Simulated Soccer Agent how to Shoot Using Artificial Neural Network. M. Dezfoulian, N. Kaviani, A. Nikanjam, M. Rafaee, 13th Multi-disciplinary Iranian Researchers Conference in Europe (IRCE), Leeds, UK, 2005.
2. "Agent 2D-3.1.0 RoboCup tools - OSDN." [Online]. Available: http://en.osdn.jp/projects/rctools/downloads/51943/agent2d-3.1.0.tar.gz/. [Accessed: 22-Jan-2016].
3. Teshneh Lab, Mohammad, and Pouria Jafari. Neural Networks and Advanced Neuro-Controlers, A Rough Neural Network Approach. Khajeh Nasir University of Technology, 2015.
4. H. Akiyama, T. Nakashima, and S. Mifune, "HELIOS2015 Team Description Paper," pp. 1–6, 2015.
5. H. Zhang, M. Jiang, H. Dai, A. Bai, and X. Chen, "WrightEagle 2D Soccer Simulation Team Description 2015," *Wrighteagle.Org*, pp. 3–8, 2015.
6. M. Prokopenko, P. Wang, and O. Obst, "Gliders2015 : Opponent avoidance with bio-inspired flocking behaviour," pp. 1–5, 2015.
7. S. Marian, D. Luca, B. Sarac, and O. Cotarlea, "OXSY 2015 Team Description," 2015.
8. Akiyama, H., Noda, I.: Multi-agent positioning mechanism in the dynamic environment . In: Visser, U., Ribeiro,F., Ohashi, T., Dellaert, F., eds.: RoboCup 2007: Robot Soccer World Cup XI, Lecture Notes in Artificial Intelligence. Volume 5001., Springer (2008) 377-C384.
9. Slotine, Jean-Jacques E., and Weiping Li. Applied nonlinear control. Vol. 60. Englewood Cliffs, NJ: Prentice-Hall, 1991.
10. Haykin S., "Neural Networks: A Comprehensive Foundation (2 ed.)", Prentice Hall, 1998.
11. N.Zare, M.Karimi, A.Keshavarzi, E.Asali, H.Alipour, A.Aminian, E.Beheshtian, H.Mowla, H.Jafari, M.J.Khademian,

"CYRUS 2D Simulation Team Description Paper 2015", The 19th annual RoboCup International Symposium, China, Hefei, 2015.
12. R.Khayami, N.Zare, M.karimi, P.Mahor, F.Tekara, E.Asali, A.Keshavarzi, A.Afshar, M.Asadi, M.Najafi, "CYRUS 2D Simulation Team Description Paper 2014", The 18th annual RoboCup International Symposium, Brazil, Joao Pessoa, 2014.