Sir2d 2014: Descrição da Time – LARC 2014

Lucas M. Moreira, Johnathan Fercher R., Frederico Gomes, Marlon Nelson G. S. Ferreira
Eduardo Krempser e Alberto T. Angonese

Laboratório de Sistemas Inteligentes e Robótica – SIRLab
Faculdade de Educação Tecnológica do Estado do Rio de Janeiro – FAETERJ Petrópolis

***Abstract*—In this article we discuss the work done by Sir2d staff of the Intelligent Systems and Robotics Laboratory of the FAETERJ Petropolis. We will demonstrate the work done to carry out the modifications made in the formations of standard defense team. Therefore, we used the tool Fedit (Formation Editor tool) to modify the pre-defined formations, in addition we use evolutionary methods for the generation and modification of formations. Considering that we are a new team, we test some frameworks based teams known as the Agent2d and UVATrilearn. For this work, we choose to work with agent2d, because even though it has not extensive documentation available as UVATrilearn, it proved to be superior in plays and formations in standard in realized tests.**

Keywords: formation, sir2d, agents, robocup simulation 2d.

***Resumo*—Neste artigo abordaremos os trabalhos feitos pela equipe Sir2d do Laboratório de Sistemas Inteligentes e Robótica da FAETERJ Petrópolis. Demonstraremos os trabalhos feitos para a realização das modificações feitas nas formações de defesa padrão do time. Para tal, utilizamos a ferramenta fedit (Formation Editor tool) para modificar as formações já pré-definidas além de usarmos métodos evolutivos para a geração e modificação das formações. Por sermos uma equipe nova, testamos alguns frameworks de times base conhecidos como, o Agent2d e o UVATrilearn. Para este trabalho, escolhemos trabalhar com o agent2d, pois mesmo que não se tenha vasta documentação disponível como o UVATrilearn, o mesmo demonstrou-se superior em jogadas e formações padrão nos testes realizados.**

Palavras chave: formação, sir2d, agentes, robocup simulation 2d.

Lucas M. Moreira é estudante de graduação em Tecnologia da Informação e Comunicação da Faculdade de Educação Tecnológica do Estado do Rio de Janeiro em Petrópolis (e-mail: lmarques.moreira@gmail.com).

Johnathan Fercher R., é estudante de graduação em Tecnologia da Informação e Comunicação da Faculdade de Educação Tecnológica do Estado do Rio de Janeiro em Petrópolis (e-mail: johnathanfercher22@gmail.com).

Frederico Gomes, é estudante de graduação em Tecnologia da Informação e Comunicação da Faculdade de Educação Tecnológica do Estado do Rio de Janeiro em Petrópolis (e-mail: fredd.demolay@gmail.com).

Marlon Nelson G. R. Ferreira, é estudante de graduação em Tecnologia da Informação e Comunicação da Faculdade de Educação Tecnológica do Estado do Rio de Janeiro em Petrópolis (e-mail: marlon_nelson_ar@hotmail.com).

Eduardo Krempser, doutor em Modelagem Computacional pelo Laboratório Nacional de Computação Científica, professor da Faculdade de Educação Tecnológica do Estado do Rio de Janeiro em Petrópolis (e-mail: krempser@gmail.com).

Alberto T. Angonese, doutorando em Engenharia de Defesa no Instituto Militar de Engenharia, professor da Faculdade de Educação Tecnológica do Estado do Rio de Janeiro em Petrópolis (e-mail: angonesealberto@gmail.com).

I. INTRODUÇÃO

A Faculdade de Educação Tecnológica do Estado do Rio de Janeiro (Faeterj) – Petrópolis possui o curso de Tecnologia da Informação e Comunicação, que além de disciplinas previstas no currículo, o aluno também participa de atividades que possam agregar sua formação profissional. E dessa iniciativa surgiu-se o Laboratório de Sistemas Inteligentes e Robótica (SIR Lab), que vem trabalhando no aprendizado de tecnologias voltadas para a área de sistemas inteligentes e robótica. Este mesmo laboratório que em 2013 levou sua primeira equipe, o SirSoccer, para disputar a CBR (Campeonato Brasileiro de Robótica) na categoria IEEE Very Small Size. Com os resultados obtidos na CBR 2013 e as interações realizadas por esta equipe com outras de diversas categorias, formou-se o interesse de participação na categoria ROBOCUP Simulation 2d, o que se concretizou no início de 2014, com a criação da equipe Sir2d.

A ROBOCUP Simulation 2d[1] é uma categoria que proporciona o estudo e desenvolvimento de várias áreas da programação, como programação orientada a agentes e uso de métodos evolutivos e de aprendizagem. Uma partida é composta por dois times com onze jogadores de cada lado. Estes jogadores são chamados de agentes, que são inteligentes e autônomos e recebem informações limitadas da situação do jogo e devem decidir sua ação na equipe como um todo. A partida é realizada num estádio virtual bidimensional de futebol, conhecido como Soccer Server. Este que possui todas as informações do jogo, ou seja, a posição atual de todos os jogadores e da bola, a física e assim por diante. O jogo depende da comunicação entre o SoccerServer e os agentes. Que por um lado, recebem dados relativos nos seus sensores virtuais como visão e audição. E por outro lado, executam ações como chutar, andar, entre outras, com a finalidade de influenciar seu ambiente.

II. CONCEITOS PRELIMINARES

A. Ambiente do Robocup Soccer Server Simulation 2d

O ambiente em que os agentes jogadores de futebol interagem é o simulator 2d conhecido como Soccer Server. Ele trabalha seguindo as mesmas regras do futebol humano e

fornece um campo de futebol virtual que tem, proporcionalmente, as mesmas medidas de um campo de futebol real. O servidor, provido pela ROBOCUP, é implementado utilizando as linguagens C e C++ e que possui como uma das características bases, uma arquitetura cliente/servidor. Onde clientes (jogadores/agentes) se comunicam com o servidor através de interfaces UDP/IP. O servidor é um sistema em tempo real que trabalha com intervalos discretos de tempo, conhecidos por ciclos. Atualmente uma partida de futebol é disputada em dez minutos, divididos em dois tempos de cinco minutos, que são 3000 ciclos.

O Soccer Server inclui também uma ferramenta para a visualização da partida, chamada de Soccer Monitor. Ela mostra o que está acontecendo dentro do servidor durante o jogo. O servidor e monitor também são conectados pela interface UDP/IP. A partir do momento em que o servidor está conectado com o monitor, a cada ciclo, ele começa a enviar informações sobre o atual estado de jogo que são interpretadas pelo monitor e mostradas na tela. (Figura 1).

Figura 1: Visão do jogo pelo Soccer Monitor

B. Arquitetura do Time Base

Um time base, também chamado de framework, provê uma interface de interação entre os jogadores/agentes e o simulador. Usando um time base, não precisamos mais nos preocupar com os detalhes da implementação da comunicação e ainda usar uma representação do modelo de ambiente de forma mais rica e funcional.

Existem vários códigos de times base disponíveis gratuitamente e entre os vários testados, optamos por utilizar o Agent2d na sua última versão para desenvolvermos o time Sir2d. O escolhemos por ter apresentado melhor desempenho em jogadas e formações básicas, oferecendo algumas habilidades básicas já implementadas, tais como a interceptação da bola, do chute, do drible, entre outras. Além de prover a implementação da comunicação com o servidor utilizando sockets, que permitem o envio e o recebimento de mensagens.

C. Agentes e Estratégia

A estratégia de uma equipe de agentes pode ser definida como um plano coletivo para alcançar um objetivo comum, com os recursos atuais disponíveis. Para um time de futebol onde existem diferentes tipos de jogador na equipe, em que eles tem que agir de maneira a realizar seu objetivo comum, que no caso é ganhar a partida.

Uma vez que um agente de futebol possui capacidade para realizar determinadas habilidades individuais, ele deve aprender a agir estrategicamente no contexto da partida selecionando uma habilidade que melhor serve para o propósito da equipe. Com isso, durante uma partida o comportamento de cada agente é formado através de um mapeamento das percepções das ações ocorridas. Ele constrói um modelo de mundo baseado nas suas percepções e o usa para selecionar uma ação apropriada para uma dada situação.

A decisão de qual habilidade que o agente tem de executar depende da estratégia do grupo, que pode ser vista como uma definição da forma em que os comportamentos dos agentes são coordenados. Através disso, surge a cooperação entre os agentes a partir dessa estratégia.

Em suma, a estratégia especifica, a formação da equipe e a posição dos agentes no interior dessa formação, definindo-os nas formações apropriadas para cada situação de jogo. Além disso, define os papéis de cada agente dentro de uma formação, especificando o comportamento associado com a sua função atual. Ela incorpora também, informações sobre como um agente deve adaptar seu comportamento à situação atual. Isso significa que o agente tem que levar em consideração qual parte do campo em que está localizado atualmente, as posições de seus companheiros de equipe e as posições de seus adversários para que ele realize uma ação. E também indica como um agente deve gerenciar sua resistência durante um jogo. No exemplo de quando um jogador estiver cansado, ele não deve fazer uma corrida para a bola a menos que seja absolutamente necessário.

III. FORMAÇÕES

A colaboração entre os agentes pode ser alcançada através da utilização de formações. A formação pode ser vista como um arranjo específico de um grupo de agentes dispostos num espaço de trabalhos através da definição de um conjunto de funções. Formações são parte do conceito motriz de um time de futebol, uma vez que cuidam da distribuição dos jogadores no campo. Quando bem usada, os jogadores serão capazes de manter as partes mais importantes do campo bem cobertos durante uma partida, evitando um agrupamento de membros da equipe em uma determinada área.

As formações definem um conjunto de papéis que consistem em um tipo de jogador e uma posição de origem no campo. Dessa forma, cada agente possui um papel próprio atribuído a si, que tem de cumprir enquanto aquela formação é utilizada no jogo. Quando determinada situação de jogo ocorre, a formação poderá ser alterada, alterando também os papéis dos agentes em campo.

Além disso, as formações são chamadas durante o jogo de acordo com a situação corrente da partida. Ou seja, se a situação do time na partida for de ataque a formação poderá ser alterada para uma formação de ataque ou se for de defesa para uma formação de defesa.

A. Modificando Formações

Neste trabalho, abordamos a modificação e geração de uma nova formação, que será utilizada numa situação de defesa durante o jogo. Como já explicado anteriormente o agente se comporta de maneira diferente para cada tipo de situação de jogo, isso por conta da mudança de estratégia juntamente com a mudança da formação.

A ferramenta de edição de formação (fedit) permite carregar arquivos .conf, que possuem um padrão pré-definido e são utilizados para carregar as formações na simulação. Ao abrir um deles, nos é permitido alterar manualmente a posição (x, y) do jogador e da bola no campo. Além disso, utilizamos um esquema de triangulação para podermos dizer aonde nosso jogador está em relação a bola. O modelo de triangulação utilizado é a triangulação de Delaunay [2], conforme ilustrado na figura 2, que por sua vez atua como um delimitador de espaço de ocupação em relação a bola.

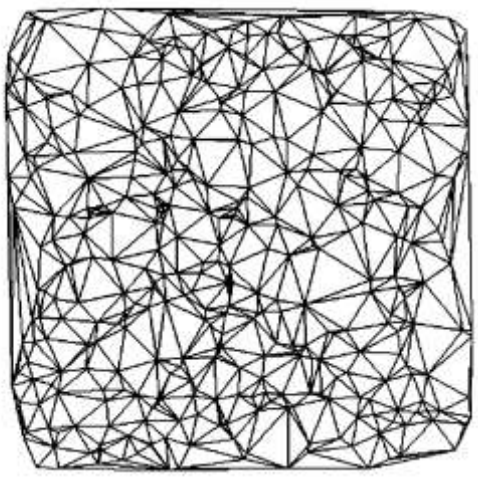

Figura 2: Triangulação de Delaunay com um conjunto de 400 pontos aleatórios no plano[2].

Além da função de delimitação de espaços, é necessário reconhecer os padrões destes espaços, para otimizar as posições de nossos jogadores. Para tal, utilizamos um método conhecido como diagrama de Voronoi que, a partir dos centroides da triangulação de Delaunay, definida na formação, determinamos a área de influência deste centro em relação aos outros. Desta forma podemos determinar um jogador dentro de uma área de influência em relação a bola.

B. Criando novas formações

Com isso podemos iniciar o processo de geração das malhas, ou seja, das triangulações sobre as quais aplicaremos um método de evolução, em que serão geradas automaticamente novas formações, permitindo-nos testá-las durante uma partida. Os melhores resultados são separados e, a partir destes, são gerados outros novos até chegarmos em uma formação otimizada.

Para utilizar os métodos Evolutivos foi criado um algoritmo que gera as triangulações no campo, alocando os vértices e as posições dos respectivos 11 jogadores em campo. Para gerar o Diagrama de Voronoi[3] foi utilizado da biblioteca OpenCV[4] que facilitou a criação do diagrama conforme está representado na figura 3, 4 e 5. Nas imagens, é possível observar o método em funcionamento. O diagrama ilustrado na imagem 3, representa a geração das triangulações, o da imagem 4, o diagrama de Voronoi gerado, e na imagem 5 temos a formação resultante que será aplicada na partida.

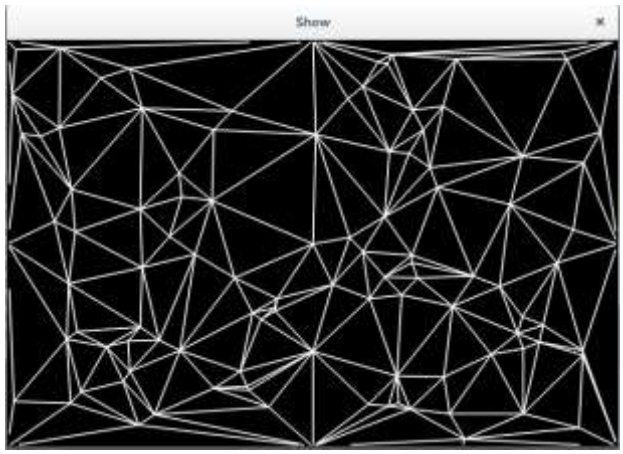

Figura 3: Diagrama que representa as triangulações.

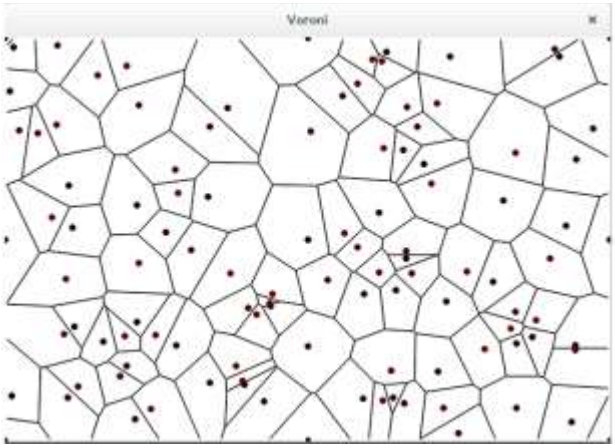

Figura 4: Diagrama de Voronoi juntamente com as triangulações.

Figura 5: Demonstração da utilização da formação gerada.

Em nosso problema, estamos aplicando um algoritmo denominado Evolução Diferencial [5] para a otimização das formações que serão utilizadas durante o jogo.

IV. Conclusão

Foi criado um algoritmo para geração e otimização de formações, que através de mecanismos de evolução diferencial tem apresentado uma melhora significativa nas formações de defesa durante a execução dos jogos de teste. Continuamos com os experimentos visando obter bons resultados na competição.

V. Agradecimentos

A Faculdade de Educação Tecnológica do Estado do Rio de Janeiro, FAETERJ Petrópolis e a Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro, FAPERJ, pelo apoio oferecido.

Referencias

[1] ROBOCUP. http://www.robocup.org/

[2] A. L. Moura, "*Uma proposta para triangulação de Delaunay 2D e localização planar de pontos em ocaml*" Tese (Doutorado) – Universidade Federal de Uberlândia, 2006, p 43.

[3] D. Reem, *"An algorithm for computing Voronoi diagrams of general generators in general normed spaces".* In Proceedings of the sixth International Symposium on Voronoi Diagrams in science and engineering, ISVD 2009, p. 144 – 152.

[4] OpenCV. http://docs.opencv.org/

[5] R. Storn, K. V. Price. *"Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces"*. Journal of Global Optimization, 1997, 11, p. 341-359