Simatch: A Simulation System for Highly Dynamic Confrontations between Multi-Robot Systems

Zhiqian Zhou,Weijia Yao, Junchong Ma, Huimin Lu, Junhao Xiao, Zhiqiang Zheng
Department of Automation
National University of Defense Technology
Changsha, P. R. China
setaria_viridis@126.com, weijia.yao.nudt@gmail.com, junchong_nubot@outlook.com,
lhmnew@nudt.edu.cn, junhao.xiao@ieee.org, zqzheng@nudt.edu.cn

Abstract—Simulation is becoming more and more important for robotics research, especially for multi-robot system research. Simatch, originating from the Robot World Cup (RoboCup) Middle Size League (MSL) match, is proposed to simulate the highly dynamic confrontation between multi-robot systems and validate the adversarial multi-robot strategies. It consists of three independent subsystems: the simulation subsystem setting up the simulated environment, the strategy subsystem realizing the multi-robot strategies and the scene subsystem describing the specific scene. Due to the independence of these subsystems, it is easy to employ Simatch for various applications. Simatch is validated by a series of tests, including the omnidirectional locomotion test, a simulated MSL match among ten robots and a simulated encirclement of sixteen robots. Since 2016, Simatch has been the official software for the MSL simulation match in the Chinese Robotics Competition. Besides, an MSL simulation project based on Simatch has been proposed to promote the development of MSL in the 2017 MSL international workshop. The proposed simulation system facilitates the development of multi-robot cooperation algorithms and related research.

Index Terms—Simatch, simulation system, highly dynamic, adversarial, multi-robot system, strategy.

I. Introduction

Middle Size League (MSL) [1] is one of the founding soccer leagues of Robot World Cup (RoboCup)[1]. In this league, two teams each comprising five autonomous robots play against each other in a soccer match according to the modified game rules based on the actual human soccer game. The novel feature of MSL is its highly dynamic and aggressive environment. Robot players with average weight nearly 40 kg can run at a speed over 5 m/s. It is a high-level testbed for multi-robot strategies. Since collision is common in this game and robots are quite complex and expensive, experiments on real robots are very costly and difficult. Therefore, we turn to build a simulation system to simulate the highly dynamic and aggressive match between multi-robot systems.

There are some available robot simulation systems. Ùbersim [2] simulates the dynamic and aggressive environment in RoboCup Small Size League (SSL). The main deficiency of Ùbersim is that the robot models can only be parameterized at compiling time. SimRobot, another simulator in RoboCup Simulation, focus on rigid body dynamics and the simulation of a variety of sensors and actuators [3]. However, it is not widely used in other domains. USARSim (Unified System for Automation and Robotics Simulation), combined with ROS (Robot Operating System), has been used in various applications since 2012 [4]. But it is only commercially available. In [5], a simulation system for MSL robots has been realized based on Gazebo, but the project has been out of maintenance.

Considering the fast implementation and perfect interface with real robot codes based on ROS, Simatch, a brand new simulation system based on Gazebo and ROS, is built to simulate an MSL match, or any other kinds of confrontation between multi-robot systems. This paper is an important extension of [6] and [7], which only introduce the implementation of the simulation environment and the realization of basic motions of simulation models, etc. In this paper, an extended simulation system consisting of several new components is developed to facilitate the research of multi-robot systems. It is notable that Simatch comes with many new features. It can not only be used to simulate the confrontation between multi-robot systems but also record all the data of the game as a data set. In addition, the game can be completely reproduced using the data-playback function. Moreover, Simatch is open source and highly flexible and extensible, it has been used for research on encirclement control [8], [9], multi-robot cooperation [10], task allocation [11] and so on.

This paper is organized as follows. In Section II, the overall architecture of Simacth is depicted. Section III-V describe the structure and function of three subsystems. Then, Section VI introduces two key interfaces combining these independent subsystems. Various validations are presented in Section VII. Section VIII concludes the paper and summarizes the future work.

II. Overall Architecture

Fig. 1 shows the general architecture of Simatch. Simatch includes two teams of robots, “Cyan Team” and “Magenta Team”, and three independent subsystems, the simulation subsystem, the strategy subsystem and the

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

scene subsystem. These subsystems are highly independent. A brief introduction for each subsystem is described in the following text.

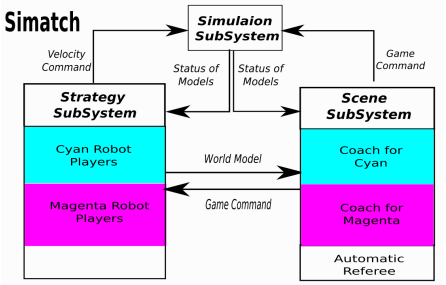


Fig. 1: The general architecture of Simatch.

The simulation subsystem is the basis of Simatch. It models the simulated environment with various simulation models based on Gazebo and realizes basic motions of simulation models. By modifying simulation models, it is possible to simulate various robots and environments. It also provides exact statuses of models for other subsystems, which is the basis of a multi-robot collaboration.

The strategy subsystem realizes multi-robot strategies. Each team of robots have their own strategies. Generally speaking, the strategy subsystem does not depend on a specific robot model or multi-robot system, which makes it possible to test multi-robot strategies for real robots on a simulated multi-robot system. By sending velocity commands, it controls simulated robots' motions. "World Model" is a virtual world built by a team of robots, including statuses of simulation models and strategies information of the team.

The scene subsystem is built to describe the specific scene and control the process of a game with game commands that divide a complicated or simple game into a series of stages. They help two teams of robots to agree on the process of the game, which is the key to realize the confrontation. The game command is built to describe different stages and it is also the only shared information for two teams of robots.

III. Simulation Subsystem

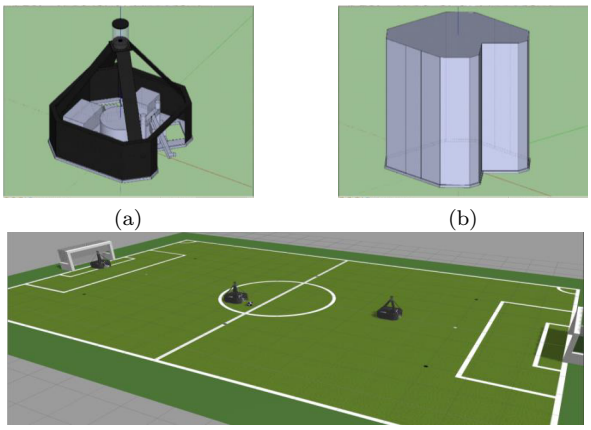
The simulation subsystem has two important functions. The first one is to set up a simulated environment, which consists of various simulation models and a simulation world. Another one is to realize basic motions of simulation models.

A. Simulation Models and the Simulation World

In [6], a typical environment for MSL matches is built. Its simulation models include the robot model, the soccer field model, the goal model and the soccer ball model. They are spawned as model plugins of Gazebo. The realization of basic motions, including omnidirectional locomotion, ball-dribbling and ball-kicking, is written in the corresponding model plugin. The simulation world determines lighting, simulation step size, simulation frequency and other simulation properties, which are shown in Table 1. The robot model and the simulation world are shown in Fig. 2.

TABLE I: Parameters for a simulation MSL match

Property Name	***Value***
physics engine	Open Dynamics Engine (ODE)
max step size	0.005
update rate	200
gravity	9.8
model plugins	ground_plane, soccer field, left goal, right goal



(a) (b)

(c)

Fig. 2: Mesh property (a) and collision property (b) of the robot model and the simulation world (c).

B. Realization of Omnidirectional Locomotion

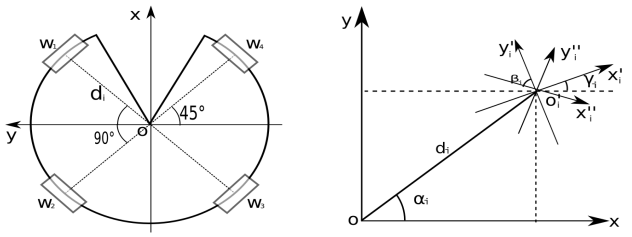
Compared with the traditional nonholonomic dual-drive wheeled robot, the omnidirectional mobile robot is able to synchronize steering and linear motion in any direction [12]. This advantage improves the flexibility of the robot greatly and enables the robot to realize faster target tracking and obstacle avoidance, which is extraordinarily important in a highly dynamic and aggressive environment. Therefore, nearly all teams employ omnidirectional motion system in real MSL matches. To improve the flexibility of simulated robots and enhance the antagonistic between simulated robots, the omnidirectional motion based on our real robot [13] is simulated.

(a) (b)

Fig. 3: Omnidirectional wheel and motion model of the wheel (a) and base frame of the real robot (b).

Our custom-designed omnidirectional wheel and its motion model are shown in Fig. 3(a) and the base frame with four omnidirectional wheels is illustrated in Fig. 3(b). The omnidirectional wheel consists of a motor-controlled wheel hub and sixteen passive rollers. Rollers rotate in

the vertical direction perpendicular to the hub rotation axis, which enables the wheel to move smoothly in any direction. The radius of omnidirectional wheels is denoted by r and the rotation speed of the ith wheel hub is represented by ω_i.



(a) (b)

Fig. 4: Layout of omnidirectional locomotion system (a) and coordinate frames (b).

Fig. 4(a) shows the layout of our omnidirectional locomotion system in two dimension plane. The gray rectangles represent four omnidirectional wheels with numbers from 1 to 4. d_i denotes the distance between the center of the ith wheel and the center of the robot. The robot coordinate frame is presented by xoy with its origin at the center of the robot.

Fig. 4(b) shows coordinate frames of each omnidirectional wheel. xoy is the robot coordinate frame and $x'_i o'_i y'_i$ is the ith wheel coordinate frame attached to the center of the ith ominidirectional wheel. $o'_i x'_i$ is the rotation center axis of motor hub and $o'_i y''_i$ is parallel to the rotation center axis of passive rollers and $o'_i x''_i$ is perpendicular to $o'_i y''_i$. α_i is the angle between ox and oo'_i. β_i is the deflection angle of the roller, representing the angle from $o'_i x'_i$ to $o'_i y''_i$. γ_i is the rotation angle between xoy and $x'_i o'_i y'_i$.

The speed of the robot is described by $(v_x, v_x, \omega)^T$ in xoy. We define $W = (w_1, w_2, w_3, w_4)^T$ to describe rotation speeds of four wheels.

Since four omnidirectional wheels share the same structure, β_i shares the same value, which is denoted by β. All d_i are decided by the layout of omnidirectional motion system and they share the same value d. Besides, $o'_i x'_i$ coincides with oo'_i in our real robots, then:

$$\gamma_i = \alpha_i, i = 1, 2, 3, 4. \tag{1}$$

According to the kinematic analysis of omnidirectional locomotion in [12], [14], the transformation from $(v_x, v_x, \omega)^T$ to W is described in (2) and (3) and parameters are shown in Table 2.

$$W = A \times \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}; \tag{2}$$

$$A = \frac{1}{r} \times \begin{bmatrix} \frac{-cos(\alpha_1+\beta)}{sin\beta} & \frac{-sin(\alpha_1+\beta)}{sin\beta} & -d \\ \frac{-cos(\alpha_2+\beta)}{sin\beta} & \frac{-sin(\alpha_2+\beta)}{sin\beta} & -d \\ \frac{-cos(\alpha_3+\beta)}{sin\beta} & \frac{-sin(\alpha_3+\beta)}{sin\beta} & -d \\ \frac{-cos(\alpha_4+\beta)}{sin\beta} & \frac{-sin(\alpha_4+\beta)}{sin\beta} & -d \end{bmatrix} \tag{3}$$

TABLE II: Parameters for the omnidirectional motion

symbol	α_1	α_2	α_3	α_4	β	d
value	$\frac{\pi}{4}$	$\frac{3\pi}{4}$	$-\frac{3\pi}{4}$	$-\frac{\pi}{4}$	$\frac{\pi}{2}$	0.203(m)

IV. Strategy Subsystem

The strategy subsystem is built to realize multi-robot strategies, including multi-robot collaboration, multi-robot path planning, task allocation and so on. Considering the interface with real robot codes, the framework of the strategy subsystem is built as a distributed multi-robot strategy system based on ROS. Its framework is shown in Fig. 5.

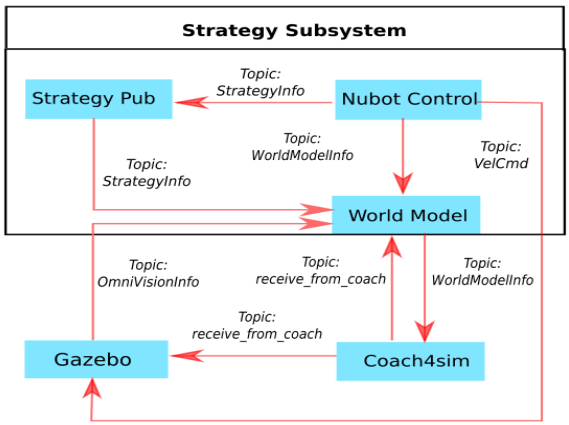


Fig. 5: The framework of the strategy subsystem.

The strategy subsystem is composed of several types of nodes. The cyan rectangles stand for nodes and the arrows represent information flow based on topics. Each node is a process for a specific task and all nodes are combined together into a graph. Its communication infrastructure mainly depends on the ROS middleware, including publish/subscribe anonymous message passing and request/response remote procedure calls[2]. The main nodes are:

Gazebo : The Gazebo node is created by the simulation subsystem. With the package named *gazebo_ros_pkgs*[3], the simulation subsystem publishes topic "OmniVisionInfo" and provides exact statuses of simulation models for other subsystems.

WorldModel : The Nubot Control node, the World Model node and the robot model make up a complete robot. As its name suggests, the World Model node is built mainly to set up a virtual world, storing the key information in the simulated environment, including statuses of models, the coach information and the strategies of its team.

NubotControl : The Nubot Control node is the core of the robot. Based on its "world", the Nubot Control node makes decisions and sends velocity commands. With

[2] http://www.ros.org/core-components/
[3] http://gazebosim.org/tutorials?tut=ros_installing&cat=connect_ros

standard interfaces with Gazebo, topic "VelCmd" can control the motion of the corresponding robot model. For every robot, the World Model node and the Nubot Control node are unique and distinct, which comprises the basis of the distributed strategy subsystem.

StrategyPub : The Strategy Pub node is built for communication between different robots. Because the strategy subsystem is distributed, every robot does not know its teammates' strategies, which are necessary to cooperate with its teammates. In the real world scenario, robots share their strategies using RTDB [15]. But it is neither feasible nor necessary for the simulation because all robots share the same IP (Internet Protocol) address. Therefore, the Strategy Pub node is built for the convenient and reliable communication via ROS messages. It subscribes to "StrategyInfo" and collects strategy information from its robots at first. Then, it fuses all strategies and publishes new "StrategyInfo".

Coach4sim : The Coach4sim node is an important component of the scene subsystem. By sending game commands, included in the topic "receive_from_coach", it controls the process of the game.

V. Scene Subsystem

As aforementioned, the scene subsystem is built to describe a specific scene. However, it is hard to understand the scene for the robot. Besides, to realize the confrontation, all robots agree on the process of the game. Therefore, we defines game commands, a series of commands, to divide the game into a series of stages, which is reasonable for that a real game is usually the repetition of simple stages. General speaking, the definition of different stages is consistent with rules of a specific scene. For example, a soccer match includes stages of game-start, game-stop, kick-off, corner-ball, penalty etc. The coach machine and the automatic referee are able to send game commands to robots and they are equipped with different functions.

A. The Coach Machine

The coach machine is built for each team. It aims not only to send game commands but also to realize the interaction between human and robot players. It is also a state visualization tool. It displays the various states of the robot, which can be used to diagnose whether the robot is in error, and assist in the improvement of the multi-robot collaborative strategies. The Coach Machine is built with three basic functions:

(1) Obtaining all information of its robot players.

(2) Sending game commands to robot players and controlling the process of soccer game.

(3) Visualize information to analyze and design better multi-robot cooperation strategies.

To ensure a simple and efficient interaction between human and robot players, a graphical user interface (GUI) is built (see Fig. 6). The GUI is divided into three areas: the command area, the display area and the status area.

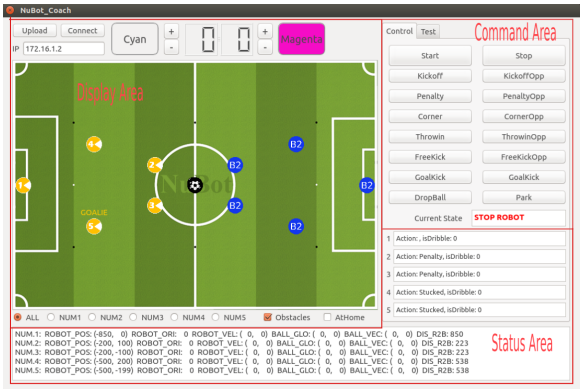


Fig. 6: The GUI of coach machine

By clicking these command buttons, corresponding game commands are sent to robot players. The main function of the display area is to visualize the positions of robot players, soccer ball and obstacles. In Fig 6, the yellow circles stand for robot players, and the blue circles stand for obstacles, including opponent robot players. The status area is used to exactly show robot players' statuses, including the position and orientation, the velocity, the action, the information of ball dribbling, etc.

B. The Automatic Referee

As stated earlier, the coach machine is built for each team and it only connects with its robot players, which makes it difficult to realize a confrontation. Therefore, the automatic referee is proposed to not only to send game commands but also to realize a confrontation conveniently. It is designed to judge automatically robot players' penalties and send game commands to all robot players.

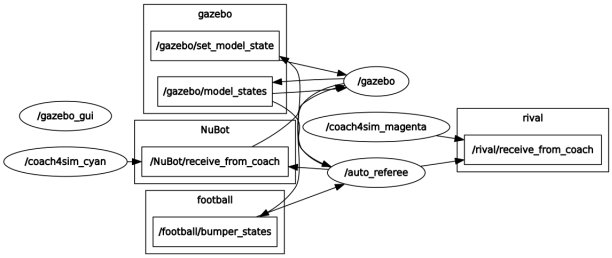


Fig. 7: A graph of the automatic referee.

A graph of the automatic referee is shown in Fig. 7. The automatic referee joins in the graph as the auto_referee node. By subscribing to topic "/gazebo/model_states" from /gazebo, it obtains exact statuses of all models. Then, it is able to judge if there is a foul or a goal according to the rules and sends game commands to all robots by publishing topic "receive_from_coach".

VI. Interfaces

As the core of Simatch, the strategy subsystem is built with two key interfaces, the interface with the simulation subsystem and the interface with the scene subsystem. Since these interfaces are based on the ROS topic/service publishing/subscription mechanism, it enables the application of the simulation system in different research fields.

A. Gazebo Interface

The Gazebo interface is mainly built on the basis of *gazebo_ros_pkgs*, which provides wrappers around the stand-alone Gazebo. In Fig. 8, topic "set_model_state" describes the desired position and velocity of models and topic "set_link_state" describes the desired position and velocity of links. Topic "link_states" and topic "model_states" consist of the exact status of simulation models. Besides, the package provides other application programming interfaces to apply the force and the torque on simulation models and links. These application programming interfaces make it practical to realize various motions of simulation models, even some motions are very complicated. Moreover, since application programming interfaces are standardized, Simatch is able to employ different models for different scenes. It supports not only most of featured models from Gazebo, but also self-built models from researchers such as our simulated soccer robot. For some specific scenes, it is allowed to build a brand new simulation models. On the ground of various simulation models, it is easy to modify the simulated world of Simatch for any specific scene.

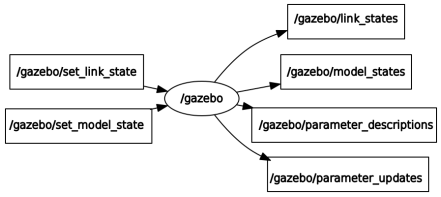


Fig. 8: A graph of the Gazebo.

B. Scene Interface

As shown in Fig. 1, the game command is the only information flow from the scene subsystem to the strategy subsystem. Therefore, the scene interface only includes topic "receive_from_coach", which consists of $MatchMode$ and $MatchType$. $MatchMode$ represents the current game command and $MatchType$ records the last valid game command, which are used to describe an exact stage of a game. Though the scene interface is very simple, it is enough to combine effectively the strategy subsystem and the scene subsystem. Obviously, the scene subsystem is still highly independent, which makes it feasible to design various scenes with different game commands.

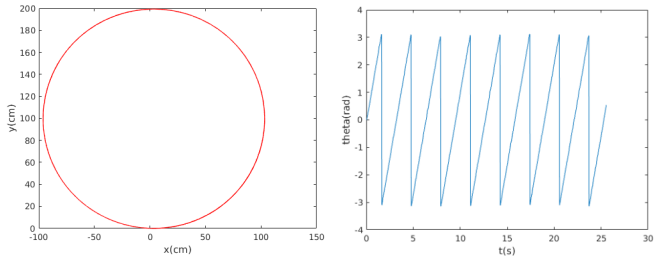
VII. Validations

A. Test of the Omnidirectional Locomotion

To validate the omnidirectional locomotion of the simulated model, a series of velocity commands are sent and the behavior of robots are observed. In this test, the given velocity commands make up the analytic function of a circle. The initial status of the robot is $(x, y, \theta)^T = (0, 0, 0)^T$ in the world coordinate frame. The unit of x and y is cm and the unit of θ is radian. The context of topic "VelCmd" is depicted in (4):

$$(v_x, v_y, \omega)^T = (200, 0, 2)^T \tag{4}$$

The test lasts about 25.56 s and the update step is about 0.015 s and results are shown in Fig. 9.



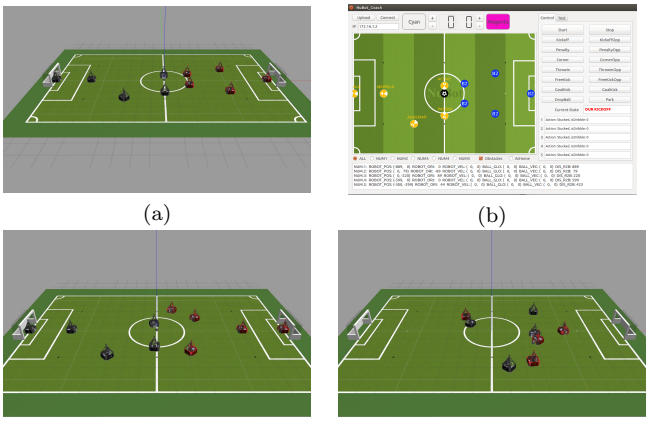
(a) (b)

Fig. 9: The trajectory (a) and the orientation (b) of the robot.

Two indicators are defined to measure the simulated robot. The first one is the position error, computed by the difference between the samples' distance to the given circle center and the given radius. Another one is the orientation error, representing the error between actual orientation and the orientation computed by ω. After statistical analysis, the average position error of 1704 samples is 2.2481 cm and the average orientation error is 0.4401 rad. The result proves that the omnidirectional locomotion model is realized.

B. Simulation of an MSL Match

Firstly, the communication between each side of robots is tested. In this step, though all robot model are spawned, just a team of robots run its codes and only one coach machine is employed. With the GUI of coach machine, we send game commands to cyan robots and observe statuses of them. The result is shown in Fig. 10(a) and Fig. 10(b).



(a) (b)

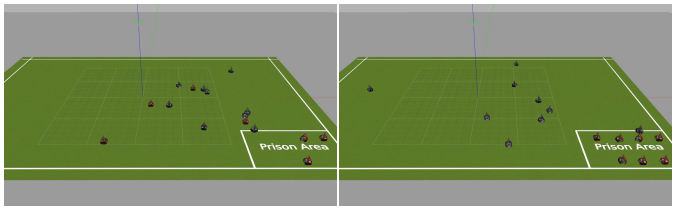
(c) (d)

Fig. 10: Simulation of an MSL match. (a) Simulation with an active cyan team; (b) Coach for the cyan team; (c) Cyan's kick off; (d) Robot players scrambling the ball.

Secondly, we run the automatic referee to control the game process. And then, the simulated match runs by itself and the automatic referee makes correct punishment to robots, which proves that the simulation of an MSL match is realized.

C. Simulation of an Encirclement

To verify that it is easy to apply Simatch to simulate other aggressive scenes, we use it to simulate an encirclement between sixteen robots, nine chasers (cyan robots) rounding up another seven targets (magenta robots). The result is shown in Fig. 11.



(a) (b)

Fig. 11: Simulation of an Encirclement. (a) Cyan robots round up magenta robots; (b) All magenta robots are driven into the prison area.

The simulation of an encirclement verifies that not only the number of robots can be adjusted easily but also the scene can be modified to different scenarios. Simatch can be applied to simulate various aggressive scenes between multi-robot systems.

VIII. Conclusion and Future Work

A. Conclusion

To test multi-robot collaboration strategies in MSL matches, we set up Simatch based on our previous work. The paper describes its general architecture and three subsystems. The simulation subsystem models the simulated environment and realizes basic motions of simulation models. To improve the flexibility of robots and enhance the antagonistic between robots, we simulate the omnidirectional locomotion based on the real robot. The strategy subsystem originates from the distributed robot codes. Actually, distributed multi-robot systems have been widely used in many domains. The scene subsystem is the key to simulate the confrontation between multi-robot systems. By sending game commands to all robots, it controls the process of the match. To combine these independent subsystems, two convenient interfaces are built based on the ROS topic/service publishing/subscription mechanism, which make Simatch flexible and extensible enough to be employed in different research fields. Then, a single robot omnidirectional locomotion test is carried out and proves that the omnidirectional locomotion is realized. Later, a simulated MSL match between 10 robot players is simulated to verify its ability to simulate the confrontation between multi-robot systems. Finally, we simulate the encirclement with sixteen robots, which verifies the flexibility and extensibility of Simatch.

To sum up, Simatch is able to simulate the highly dynamic confrontation between multi-robot systems, which makes tests of adversarial multi-robot strategies convenient and effective, and it promotes studies on adversarial multi-robot problems. Simatch has been applied to the MSL simulation match[4] in the Chinese Robotics Competition. Besides, a simulation project[5] based on Simatch has been proposed to promote the development of MSL in the 2017 MSL international workshop.

B. Future work

Our future work will focus on two aspects. At first, we will create more simulation models to simulate different kinematic models, such as drones, to cater for different application scenarios. Then, we will set up different simulation worlds and design different application scenarios to expand the simulation system to other research fields.

References

[1] R. Soetens, R. V. D. Molengraft, and B. Cunha, RoboCup MSL - History, Accomplishments, Current Status and Challenges Ahead. Springer International Publishing, 2015.

[2] B. Browning and E. Tryzelaar, "Übersim:a multi-robot simulator for robot soccer," in Proc. of the 2nd Int. joint Conf. on Autonomous agents and multiagent systems, 2003, pp. 948–949.

[3] T. Laue and S. Kai, SimRobot – a general physical robot simulator and its application in robocup. Springer-Verlag, 2006.

[4] S. Balakirsky and Z. Kootbally, "Usarsim/ros: A combined framework for robotic control and simulation," in ASME 2012 International Symposium on Flexible Automation, 2012, pp. 101–108.

[5] D. Beck, A. Ferrein, and G. Lakemeyer, A Simulation Environment for Middle-Size Robots with Multi-level Abstraction. Springer-Verlag, 2008.

[6] W. Yao, W. Dai, J. Xiao, H. Lu, and Z. Zheng, "A simulation system based on ros and gazebo for robocup middle size league," in IEEE International Conference on Robotics and Biomimetics, 2016, pp. 54–59.

[7] J. Xiao, D. Xiong, W. Yao, Q. Yu, H. Lu, and Z. Zheng, Building Software System and Simulation Environment for RoboCup MSL Soccer Robots Based on ROS and Gazebo. Springer International Publishing, 2017.

[8] W. Yao, Z. Zeng, X. Wang, H. Lu, and Z. Zheng, "Distributed encirclement control with arbitrary spacing for multiple anonymous mobile robots," in Chinese Control Conference, 2017.

[9] W. Yao, H. Lu, Z. Zeng, J. Xiao, and Z. Zheng, "Distributed static and dynamic circumnavigation control with arbitrary spacings for a heterogeneous multi-robot system," Journal of Intelligent & Robotic Systems, pp. 1–23, 2018.

[10] J. Ma, W. Yao, W. Dai, H. Lu, J. Xiao, and Z. Zheng, "Cooperative encirclement control for a group of targets by decentralized robots with collision avoidance," in The Chinese Control Conference, 2018.

[11] W. Dai, H. Lu, J. Xiao, and Z. Zheng, "Task allocation without communication based on incomplete information game theory for multi-robot systems," Journal of Intelligent & Robotic Systems, no. 1, pp. 1–16, 2018.

[12] C. Wang, X. Liu, X. Yang, F. Hu, A. Jiang, and C. Yang, "Trajectory tracking of an omni-directional wheeled mobile robot using a model predictive control strategy," Applied Sciences, vol. 8, no. 2, p. 231, 2018.

[13] D. Xiong, J. Xiao, H. Lu, Z. Zeng, Q. Yu, K. Huang, X. Yi, and Z. Zheng, "The design of an intelligent soccer-playing robot," Industrial Robot, vol. 43, no. 1, pp. 91–102, 2016.

[14] Z. Li, C. Yang, C.-Y. Su, J. Deng, and W. Zhang, "Vision-based model predictive control for steering of a nonholonomic mobile robot," IEEE Transactions on Control Systems Technology, vol. 24, no. 2, pp. 553–564, 2016.

[15] F. Santos, L. Almeida, P. Pedreiras, and L. S. Lopes, "A real-time distributed software infrastructure for cooperating mobile autonomous robots," in International Conference on Advanced Robotics, 2009, pp. 1–6.

[4] https://github.com/nubot-nudt/simatch

[5] https://github.com/RoboCup-MSL/MSL-Simulator/wiki