Agent Navigation in Virtual Soccer: Comparative Analysis of Algorithms

Michail Panteleyev[1 [0000−0002−6077−760X]] and Anushervon Salimov[1 [0000−0003−1592−7012]]

Saint Petersburg Electrotechnical University "LETI", Department of Computer Science and Engneering, ul. Professora Popova 5, 197376 St. Petersburg, Russian Federation
{mpanteleyev,safmail96}@gmail.com

Abstract. The article is devoted to the analysis of various methods of solving navigation problems by an intelligent agent in the environment of virtual football Robocup Soccer on the basis of noisy data coming from a visual sensor. Two groups of methods for calculating the absolute coordinates of objects based on sensory data on flags and lines are considered: trigonometric methods and methods based on the use of the Kalman filter and particle filter. A software tool developed for experimental research and allowing arbitrary variation of the conditions for solving the navigation problem is briefly described. Experimental results of the comparative analysis of the speed and accuracy of algorithms implementing various methods are presented. The obtained results allow the agent to solve the navigation problem using anytime-algorithms, exchanging the solution time for the quality (accuracy) of the result. Taking into account the obtained results, the directions of further research on the implementation of the assessment of the tactical situation in virtual soccer are determined.

Keywords: intelligent agent · multi-agent system · virtual soccer · RoboCup Soccer · localization · anytime algorithms · particle filter.

1 Introduction

The creation of intelligent agents (IA) and based on them multi-agent systems (MAS) is currently the main direction of the development of artificial intelligence [1, 2]. IA is understood as autonomous systems that perceive the environment and implement purposeful behavior in this environment. An important feature of IA is their ability to act in groups, including in the face of active opposition from other groups of agents. In recent years, virtual soccer has been used as

a reference platform for practicing various approaches to building real-time IA (RIA) and tactics of group behavior of MAS in conditions of group opposition [3-5]. An ambitious task has been set by the community of experts in the field of IA and MAS - by 2050 to create a team of autonomous soccer players who can beat the team of world champions. One of the important tasks solved by the IA is navigation (determination of one's own location) based on information received from visual sensors. To solve this problem, probabilistic approaches are widely used, the most famous of which are the Kalman filter and particle filter [6, 7]. The selection of the most effective methods and algorithms for solving the navigation problem, taking into account the specific features of the specific operational environment of the IA, requires their experimental comparative analysis.

1.1 The features of the navigation task in the soccer environment

Virtual soccer environment (VS) RoboCup Soccer Server allows real-time simulation of the game of football teams consisting of 11 Autonomous IA. Each player is implemented as an independent program that connects to the server via a UDP socket. The server provides a virtual field on which it simulates the actions of players in accordance with the commands coming from them. In each step of the simulation, the player receives sensory information from the server and sends back commands about his own actions. The agent has three sensors: visual, auditory and body sensor. The agent receives the main information from the visual sensor, which it can control by setting the width and direction of the view. At any given time, the agent sees only a part of the field. The frequency of receiving visual information from the server depends on the agent's chosen sensor mode. Figure 1 shows a general view of the field with flags placed on it (see Fig. 1).

Flags are static objects with a priori known coordinates and, as a result, can be used as navigation bindings along with lines. The visual sensor provides information about all objects in the field of view: the ball (b), players (p), flags (f), lines (l) and goals (g). Flag specifiers determine their position on the field. For example, (f c) – flag in the center of the field; (f p l b) - flag marking the bottom corner (b – bottom) of the penalty area (p – penalty area) of the left half of the field (l – left). An example of a frame of visual information coming from the server is shown below:

(see 0 ((f c) 32.4 0 0 0) ((f r t) 45.3 -4) ((f c t) 53.3 40) ((f p l t) 39.7 35) ((f p l c) 36.4 23) ((f t 0) 62.3 -8) ((f t r 10) 59 5) ((f t r 20) 64.7 9) ((f t r 30) 63.3 17) ((f t r 40) 74 19) ((f t r 50) 82.6 28) ((f t l 10) 65.3 -17) ((f t l 20) 67.7 -26) ((f t l 30) 65 -32) ((f t l 40) 74.5 -44) ((f r t 20) 77.6 42) ((f r t 30) 85.6 34) ((b) 24.3 0 0 0) ((p "TeamName" 10) 43.1 32) ((l t) 63.4 66))

The agent receives visual information about each observed object in the polar coordinate system, the center of which is itself (i.e. azimuth and distance to the object). Using this relative information and a priori knowledge of the coordinates of static objects, he can calculate his own absolute coordinates and then the absolute coordinates of other players. At different points of time (simulation cycles) in the field of view of the agent, depending on its position and direction

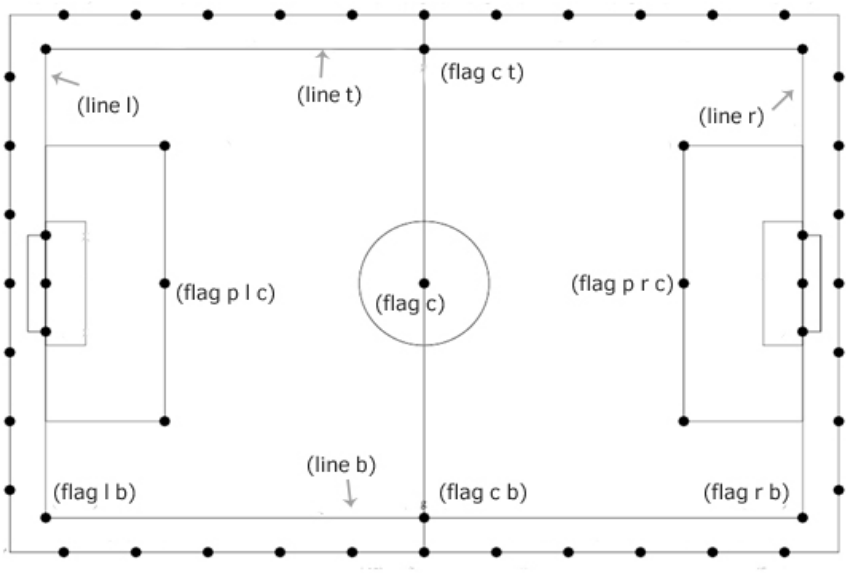

Fig. 1. General view of the field with flags placed on it.

of view are different navigation bindings (flags and lines). At the same time, the number of such bindings and the distance to them vary widely. A significant factor in solving the navigation problem is the inaccuracy (noise) of the sensory information received from the server. In particular, the noise imposed on the distance to the object is calculated by the server using the following formula:

$$QDistance = Quantize(exp(Quantize(ln(Distance), StepValue)), 0.1) \quad (1)$$

where QDistance is the distance perceived by the agent to the observed object; distance is the true distance to the object; quantizeStep is the quantization step.Thus, for an object located at a distance of 1 m, the error in determining the distance can be 10 cm, and at a distance of 10 m-up to 1 m.
The problem of determining the absolute coordinates of the players (themselves and others) on the observed information can be solved by different methods and algorithms, depending on the time spent and the accuracy of the solution.
In [2, 8-10], an approach to the construction of IA was developed, according to which the agent in problem situations dynamically determines the available stock of time and adapts the process of thinking about the solution to it. Within the frame-work of this approach, algorithms for solving particular problems are constructed as anytime-alghoritms(AA), in which the quality of the results (in particular, accuracy) depends on the time allocated to this algorithm. In the framework of this approach, the solution of the navigation problem is considered from the standpoint of the AA. Comparative experimental analysis of various

methods of solving the navigation problem is necessary to construct the AA profiles that fix the dependence of the accuracy of the result on the time allocated to the algorithm.

2 Navigation methods and algorithms

Based on the analysis of the literature [6, 7] the following methods of solving the navigation problem in the VS environment were identified:

1. on the nearest flag and the long line
2. by the two nearest flags
3. based on Kalman filter
4. based on Kalman filter

2.1 Navigation methods based on trigonometric models

Navigation on the nearest flag and back lines. In this case, the nearest flag and the farthest line are determined first among all visible objects. The player's direction to the line is used to find the angle of rotation of the agent's head. Let lb be the bisector of the player's viewing angle passing through his absolute coordinates. If the line l is in the player's field of vision, the visual message he receives will contain the distance d to the intersection point lb with l and the angle α between lb and l (see Fig. 2). In this case, the angle α will be equal to the rotation angle lb until it coincides with l, which has a positive value when rotating clockwise and a negative value otherwise. In this case, the angle α will be equal to the angle of rotation lb until it coincides with l, which has a positive value when rotating clockwise and a negative value otherwise. To get the angle of rotation of the player's head, calculate the angle β between lb and the lp line perpendicular to l. the angle β is calculated using the following formula:

$$\beta = -sign(\alpha) \cdot (90 - |\alpha|) \tag{2}$$

After calculating the angle θ, you can proceed to processing information about the flag. Sensory information about the f flag includes the direction f_φ is pointing at it relative to the player's head rotation angle and the distance fr to it. Since the agent knows the absolute coordinates (f_x, f_y) of all flags a priori, it can calculate its absolute coordinates (p_x, p_y) based on the relative sensory information. To do this, you must first perform a rotation by the angle θ of the player's head rotation in the polar coordinate system. The absolute coordinates of the player (p_x, p_y) can be obtained by converting polar coordinates to Cartesian coordinates:

$$(p_x, p_y) = (f_x, f_y) - \pi \cdot (f_r, f_\varphi + \theta) \tag{3}$$

a) b)

Fig. 2. Navigation on the nearest flag and back lines

Navigation on the two nearest flags and the farthest line. In this algorithm, the two closest flags among all visible flags are first determined. Knowing the absolute coordinates of the flag and the distance to it sets the circle of possible positions of the agent with the center at the location of the flag and the radius determined by the distance to it. Similarly, defines the range of possible positions of the agent and the second flag. The position of the agent is obviously determined by the intersection of these circles (see Fig. 2).
To determine a point unambiguously, you must calculate the distance d between the flags f and g using their absolute coordinates:

$$d = \sqrt{(g_x - f_x)^2 + (g_y - f_y)^2} \tag{4}$$

Distance d, as seen in Fig. 3, consists of segments [f; p'] and [p'; g]. The figure shows: a-the distance from the flag f to the point p'; b-the distance from the point p' to the flag g; h-the distance from the point p to the point p'. For triangles gpp' and fpp', you can write:

$$\{\, f_r^2 = a^2 + h^2 g_r^2 = b^2 + h^2 \tag{5}$$

Given the ratio b = (d – a), you can calculate the value of a:

$$a = \frac{f_r^2 - g_r^2 + d^2}{2d} \tag{6}$$

Then the absolute coordinates of the point p' will be defined as follows:

$$(p'_x, p'_y) = (f_x + a \cdot cos(\alpha), f_y + a \cdot sin(\alpha)) \tag{7}$$

The values cos(α) and sin(α) are determined from the relations:

$$sin(\alpha) = \frac{\Delta y}{d}, cos(\alpha) = \frac{\Delta x}{d} \tag{8}$$

Based on this, the absolute coordinates of player p will be equal:

$$(p_x, p_y) = (p'_x - h \cdot Sign \cdot sin(\alpha), p'_y + h \cdot Sign \cdot cos(\alpha)) \tag{9}$$

The true location of the player can be determined using the Sign sign: if the difference is positive (g_φ – f_φ), Sign = +1, otherwise Sign = -1.

2.2 Navigation methods based on the filters

When determining the absolute coordinates of the agent on the football field, trigonometric methods use visual information obtained only in the current measure. At the same time, information about a certain object obtained in different sensory measures can be considered as successive observations of a dynamic process.
In approaches that use filtering ideas, agent states (coordinates) are considered as a managed Markov process with hidden States x_t and a time step t.
Let's denote x_0 – the initial state of the player, $P(x_0)$ - the initial distribution at time t = 0. Then the player dynamics implemented by the server can be considered as a stochastic transition model $P(x_{t+1}|x_t, a_t)$, in which the agent changes its state x_t to the state x_{t+1} by its actions at performed at time t. We will assume that the y_t sensor data sent by the server in each clock cycle is conditionally independent of the x_t. Then the navigation problem can be reduced to an estimate of a posteriori probability density $P(x_t|y_t)$ in the state space X, describing the current state of the agent at time t. by the Bayes rule, you can write:

$$P(x_{t+1}|y_{t+1}) \propto P(x_{t+1}|y_{t+1})P(x_{t+1}) \tag{10}$$

where the a priori probability density function $P(x_{t+1}, y_{t+1})$ corresponds to a posteriori function from the last time step. For $P(x_{t+1})$, you can write:

$$P(x_{t+1}) = \int P(x_{t+1}|x_t)P(x_t|y_t)dx_t \tag{11}$$

This formula uses the Markov assumption that the current values are independent of previous time steps.
Equations allow us to construct an iterative Bayesian filtering scheme, in which the integral must be calculated according to the expression in order to analytically determine a posteriori probability. The result of the calculation must be multiplied by the value of the probability density $P(y_{t+1}|x_{t+1})$, and then normalize the probability density $P(x_{t+1}|y_{t+1})$. A posteriori probability can be calculated analytically if the observation and transition models are linear Gaussian. Since the Robocup Soccer simulation environment does not meet this requirement, approximations must be used.
An effective method for calculating a posteriori distribution in Bayesian filtering is the particle filter. It is based on a discrete approximation of the continuous function of the posterior density using a set of X_t^i particles with corresponding weights π_t^i, where i = 1, ..., N. the Empirical a posteriori estimate has the form:

$$P(x_t|y_t) = \sum_{i=1}^{N} \pi_t^i \delta(x_t - x_t^i) \tag{12}$$

Using the formula above, the integral for calculating the expression can be replaced with summation:

$$P(x_{t+1}) = \sum_{i=1}^{N} \pi_t^i P(x_{t+1}|x_t^i) \tag{13}$$

After replacing all integrals with sums and all continuous density functions with discrete expressions, the normalization of the fixed a posteriori function is reduced to the normalization of discrete values per unit of the sum.

$$P(x_{t+1}|y_{t+1}) \propto P(x_{t+1}|y_{t+1}) \sum_{i=1}^{N} \pi_t^i P(x_{t+1}|x_t^i) \tag{14}$$

Navigation based on the Kalman filter. The Kalman filter allows you to get an estimate of the state vector of an object (in this case, the player's coordinates) based on a series of noisy measurements. Solving the navigation problem in the VS using the Kalman filter enlarged includes the following steps:

1. Analysis and analysis of the received visual information in order to obtain a list of flags visible to the player and their relative coordinates.
2. Cyclic processing of various pairs of flags. In each cycle, two visible flags are selected and the absolute coordinates of the agent are calculated by these flags.
3. Calculation of the variance of the sensor error. (The quality of the variance estimation determines the quality of the Kalman filter).

$$\sigma_{i+1}^2 = \frac{(r_{max} - r_{min})^2}{2} \tag{15}$$

4. Updating the value of the Kalman gain taking into account the obtained dispersion. The coefficient value should provide the maximum proximity of the calculated op-timal values of the absolute coordinates to their true values.

$$K = \frac{\hat{\sigma}_i^2}{\hat{\sigma}_i^2 + \hat{\sigma}_{i+1}^2} \tag{16}$$

5. Correction using the Kalman coefficient of the estimated value of the absolute co-ordinates of the agent in this iteration.

$$\hat{x}_{i+1} = \hat{x}_i + K \cdot (y_i - \hat{x}_i) \tag{17}$$

Particle filter navigation. A particle filter is a method for determining the absolute coordinates of an agent, in accordance with which many hypotheses (particles) about their current values are created to estimate coordinates.
The algorithm for determining absolute coordinates based on a particle filter includes five steps: initialization, forecast, correction (calculation of weighting coefficients and resampling), and state estimation. Particle filter initialization is

performed at the mo-ment of receiving the first sensor frame and is reduced to randomly generating parti-cles around the obtained point. As a result, in the vicinity of the point, N particles with initially equal weights are generated. The position of each particle is determined on the basis of information about the trajectory and distance received from the visual sensor. At the forecasting stage, the range of possible values of the distance to the agent is taken into account. If the distance between the particle and the agent is out-side this range of values, the particle is removed from the set. Thus, after the forecast-ing stage, taking into account information on the range of values from the initial set of N particles, K particles remain. The correction procedure is performed each time new sensory information is received and includes two steps: determining value ranges and resampling.

Before calculating the range of values, some particles can be removed using Kalman filtering. Each time new information is received, the upper and lower bounds of the distances between the particles and the agent are determined. Particles that are out of range based on the prediction results are removed from the total set of particles.

This allows you to discard false hypotheses about the possible position of the object and, thus, reduce computational costs and improve the accuracy of the result. After a few steps of the correction procedure, most particles that correspond to erroneous hypotheses may have weights close to zero. Such particles hardly contribute to the final estimation of the state vector, but they spend computational resources. The resampling procedure allows you to re-allocate computing resources by discarding low-weight particles and duplicating high-weight particles.

At the last stage of the state assessment, the absolute coordinates of the agent are calculated as a weighted sum of the states of all particles.

3 Approach and results of experimental evaluation of navigation algorithms

For experimental research and comparative analysis of the effectiveness of various navigation algorithms, a tool program in the Java language has been developed that allows:

1. randomly position the agent on the field and set the direction and mode of operation of its visual sensor;
2. solve the navigation problem by various methods and evaluate the accuracy of the solution and the time spent

Since the number of navigational anchors (flags) falling into the agent's field of vision depends significantly on the viewing angle, the experiments were carried out sep-arately for different viewing angles: narrow, normal, and wide. The results of the study with a normal viewing angle, when the agent sees few flags, are presented in Fig. 4.

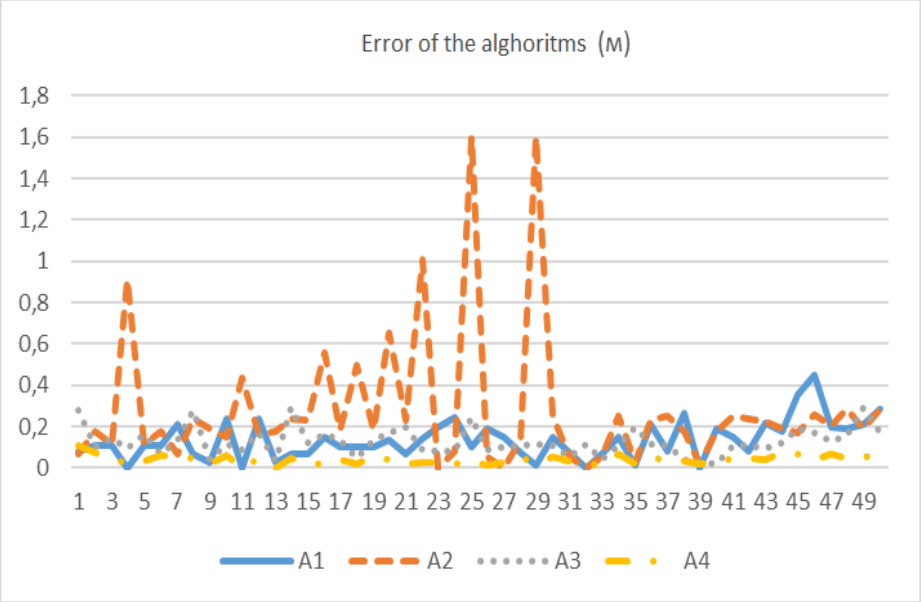

Fig. 3. Error of the algorithms

The errors of the algorithms were calculated at different 50 absolute coordinates. The graphs in the figure correspond to the following algorithms for calculating the abso-lute coordinates of the agent: A1 - by the nearest flag and the far line; A2 - by the two nearest flags; A3-using the Kalman filter; A4-using the particle filter.
As can be seen from the above dependencies, the greatest error is given by the algorithm for determining the coordinates of the two nearest flags. The most accurate method is the particle filter method, since it uses information about all visible flags, taking into account the sequence of observations.
The results of the experimental estimation of the algorithm implementation time are shown in Fig. 5. The experiments were carried out on a PC with the following charac-teristics: PC 2 cores, 4 logic processors, frequency 2.2 GHz

As the experiment showed, the algorithm using the particle filter works 7-15 times slower than other algorithms, due to the large number of processing operations. The averaged values of the error in determining the absolute coordinates of the agent and the time spent for different algorithms at a narrow viewing angle are presented in summary table 1.

The analysis of the obtained results shows that in the aggregate of two characteristics - accuracy and time – the algorithm of determining the absolute coordinates of the agent by the nearest flag and the far line is preferable. This time-consuming algorithm is slightly inferior to the fastest algorithm for determining the coordinates of the two nearest flags and, at the same time, gives an

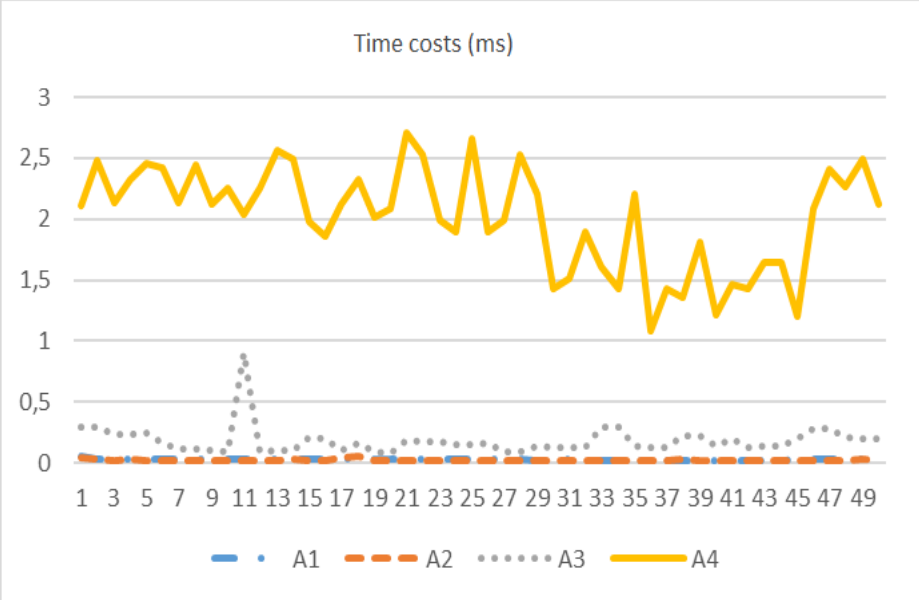

Fig. 4. Algorithm's time complexity

Table 1. Comparison of algorithms

Algorithm for determining the absolute coordinates of the agent	Average error (m)	Time (ms)
On the nearest flag and the long line	0,2482	0,0274
By the two nearest flags	1,0154	0,0222
Based on Kalman filter	0,2901	0,1826
Based on particle filter	0,0974	2,0154

error comparable to the algorithm based on the Kalman filter.

4 Summary

According to the criterion of accuracy of the solution of the navigation problem in the VS, the algorithm based on the particle filter is the best. However, it requires the maximum time (4 ms on average). Trigonometric algorithms for determining absolute coordinates are fast, but have a significantly lower accuracy, significantly dependent on the number of visible flags and the selected quality of the visual sensor. In addi-tion, these methods can give abnormally high errors at close azimuths of the direction to the nearest flags. The presence of various algorithms for solving the navigation task, wherein the accu-racy of the result, allows us to consider them from the standpoint of AA and adapt the deliberation process agent decisions based on dynamic (situational) changing time limits.

Thus, the directions of further research are: - creation of the refined profiles of AA of the decision of a navigational problem and their use in architecture of RIA for the environment of VS; - use of the calculated absolute coordinates of the players to determine the tactical arrangements of the teams during the game.

References

1. Russell S., Norvig p. Artificial intelligence: a modern approach, 2nd ed. Moscow: Izdat. Williams House, 2006. - 1408 p
2. Panteleyev M. G., Puzankov D. V. Intelligent agents and multi-agent systems: a monograph.:Publishing house SPbGETU "LETI", 2015. - 215 p.
3. Official website of the tournament: [Electronic resource] http://www.robocup.org Last accessed 10 Dec 2019
4. Robocup Soccer Server: https://github.com/rcsoccersim/rcssserver/ Last accessed 10 Dec 2019
5. Thrun S., Wolfram B., Fox D. probabilistic robotics (intelligent robotics and Autonomous agents)/ / The MIT Press, 2005. – 672 P.
6. Panteleyev M. G. the Concept of constructing intelligent real-time agents based on the model of advanced iterative planning / / proceedings of the 13th NAC. Conf. AI with international participation CII-2012. T 3. - Belgorod: Publishing house of BSTU. - 2012. - 25-33.
7. Panteleyev M. G. Formal model of advanced iterative planning of actions of intelligent real-time agents. proceedings of the 14th national Academy of Sciences. Conf. at the AI con-ference. on artificial intelligence with international participation CII-2014. T 1. - Kazan: pub-lishing house RIC "school". - 2014. 323-333.
8. Panteleyev M. G. extended iterative action planning for intelligent real-time agents / / Proceed-ings Computer Science, 2019, Vol. 150, PP.