

Behaviour Study of a Multi-Agent Mobile Robot System during Potential Field Building

István Nagy

Institute of Mechatronics and Vehicle Techniques
Bánki Donát Faculty of Mechanical and Safety Engineering
Budapest Tech
Népszínház u. 8, 1081 Budapest, Hungary
nagy.istvan@bkgk.bmf.hu

Abstract: In this paper a multi-agent based mobile robot simulation system will be presented where the behaviour of the system is studied with different number of agents (1, 3,6) and also with different number of ultrasonic range sensors on agents (8 or 16 US sensors on individual agents). The task of the autonomous agents is to create the potential field (PF) of an unknown environment. The classic problems of PF building, like oscillation and trapping, are not the focus of the article, but instead, the article is concerned with the agents' self-organizing ability where self-organizing is controlled by a genetic algorithm (GA). The GA is equipped with two fitness functions where one "maintains" the distances between certain agents (spat distr), while another "watches" the area coverage (area cover). In fact, the paper can be divided into three main parts. The first part describes the ultrasonic sensing and range measuring with systematic errors, the potential field (PF) building and the moving strategies. The second part contains description of the GA, the operation of the GA, the structure of the system, the fitness functions and a general system-error determination. In the final third part, the obtained results are analyzed and presented in the appendices.

Keywords: Genetic algorithm (GA), Mobile Robot, Multi-agent, Potential Field

1 Aims and Motivation

Nowadays, in mobile robot research a huge amount of literature is available about path planning and course controlling based on a potential field. These articles are mostly about eliminating or preventing the classic problems arising in potential field building, such as trapping and oscillation. With the evolution of this area of knowledge, newer and newer methods are appearing for handling these mentioned problems, but these methods usually concern single agents. A good example can

be seen in [1], where the authors are eliminating the oscillation problem by a VFB¹ guiding model, in which at path planning the VFB is realized by a neuro-fuzzy model producing an oscillation free path between the starting and docking positions. The developed algorithm was tested in a virtual training environment named “COSMOS”, and the results can be found in the mentioned article. In relation to this, another example can be mentioned where the classic parking problem is realized by a hybrid navigation structure, with the elements of computational intelligence [2]. The hybrid structure has three components: *harmonic PF* (calculation of the path in an initial – static – environment); *neural network* (trying to control the robot to pass through the orientation marks that the path is composed of); *fuzzy controller* (obstacle avoidance and trying to find the next orientation marks again). For simulation results see [2].

My primary aim is to create a functional simulation system that will be able to create the potential field of an unknown environment on a multi-agent platform. While developing it I will not devote to the classic problems of potential field building (trapping, oscillation), but I rather wish to control the group behaviour of agents, believing that the previously mentioned basic problems can also be eliminated by this. My secondary aim is, in case of a successful system, to accomplish its analysis (see conclusion) and (probably in another article) to increase the efficiency of the algorithm by tuning the system or the GA parameters. Later in the future I would like to apply this algorithm for multi-agent systems with different sensors (e.g. visual sensors) as well.

2 Introduction

This paper actually is a continuation of the conference paper [3], and this is why the basic definitions and determinations published previously are mentioned here only in a shortened form.

Distributed problem solving at multi-agent mobile robot systems has its origin in the late 1980s [4], [5], however, since 2000, the field of cooperative mobile agents has shown dramatic development. It is reasonable to ask: Why should we use multi-agent mobile robot systems? Answering it, let me compare several advantages of multi-agent systems, as contrasted with single-agent ones.

- More efficiency (faster and more accurate).

Keeping to the main topic of the paper, in multi-agent systems – by exchanging the main information between one another –, the individual agents are capable to localize themselves faster and more accurately.

¹ *Vector Field Based guiding model*

- More fault-tolerant.

Namely, if in a single-agent system the agent breaks down, the task will not be executed, while in a multi-agent system, though depending on its intelligence, the execution of the task is continued.

Generally speaking, a multi-robot system has the following remarkable properties:

- Larger range of task domains (flexibility)
- Fault-tolerance
- Greater efficiency, robustness

In the development of multi robot systems, primary merits can be attributed to M. J. Mataric (MIT, USA) whose scientific achievements include researching and developing strategies of behaviour-based mobile robot systems [6], [7], [8]. Each of these studies contains relevant statements and definitions in the field of individual or group behaviour of mobile robots. The individual agent is very well defined by Tecuci in [9] –“the agent is an autonomously active entity with certain possibilities to sense its environment and act in it in order to achieve certain states of this environment in which certain previously specified goals are achieved”. Later, by the development of this field of science different types of agents were defined, and this can be observed very well in [10], where the basic classification of agents is extended and apart from this the agents are classified from a functional-computational perspective. After the definition of single agents, we can now focus on multi-agent systems and mainly on the cooperation between individual agents. In [11], the authors try to draw the agents into an agent coalition for the sake of a more efficient task execution. Firstly, the agent coalition is formed, the individual agents are rated and some value is assigned to them. Then, based on the agent’s value, the agent will join the coalition if the coalition brings to the agent at least the same or better results than when it works independently. Another important contribution has been made by Fukuda and Iritani, who tried to widen the possibilities of the cooperation between separated agents in multi agent mobile robot systems [12].

The simulation system, described in this paper has a modular structure. There is a separate module for the sensory system of the agents (which is the mathematical model of the ultrasonic range detector), another module contains the GA, responsible for near-optimal behaviour selection, and the next separated module is responsible for displaying results and assessments.

Since the simulation system has been prepared in a MATLAB environment, it is inevitable to make the mathematical model of the system. The workspace is digitally decomposed (grid construction), and the agents are point-represented in this model. The visited areas are renumbered during the process of map-building, in order to avoid duplicity of the map occurring in the same area. The potential field building and calculation is based on the principle of the well-known

repulsive forces. A simplified map building process by one agent is represented on Figure 1. The agent moves to the new position, assigns the position to the grid construction of the model, performs distance measurements, evaluates the potential field value (broadcasting the parameters of its own new position to the other agents), and then plans the next move. The potential field values are stored on the host remote server, where the global potential field map of the whole WS will be updated. In the advanced systems (will be represented in this paper), in order to avoid collisions, the moving mechanism is controlled by the GA.

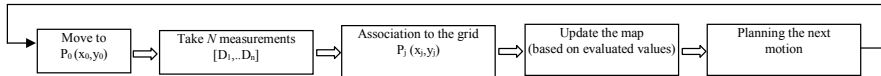


Figure 1

A simple map building process

3 Sensing

The individual agents are equipped with 8 or 16 ultrasonic sensors for distance measurements. The sensors are equally spaced on a ring around the body of the robot (see Figure 2a) so that the sensors form a regular octagon (or a polygon with 16 points) on the circle of the agent. In this case the sensing sector of each sensor can be calculated with the form:

$$\beta = \frac{2\pi}{N}; \quad (1)$$

where, N is the number of sensors. The sensors can also choose either *long-* or *short-range* sensing. The long-range sensing (*LRS*) perceives the obstacle or other agents in the given sector (β) in infinite² distances. The short-range sensing (*SRS*) is determined in a circle with radius R_0 . Occupation of the segments by other agents or obstacles, is represented with a binary word, and will have importance in choosing the next behavior or the moving mechanism.

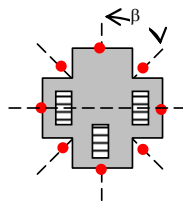


Figure 2a

The agent, and the sensors around, located by angle β

² infinite=beyond the given radius R_0

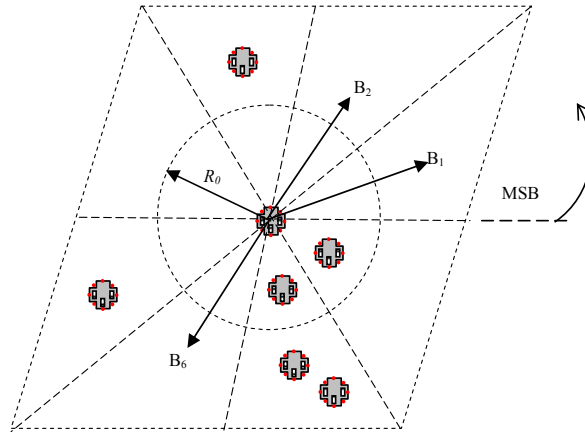


Figure 2b

The Long, and Short range sensing

Let us see the illustration given on Figure 2b, where the binary words of short-respectively long-range sensing are:

LRS: 00101010

SRS: 00000011

Mathematically can be written:

$$b_j = \begin{cases} 1, & \exists P_i \in \varepsilon_j; \\ 0, & \forall P_i \notin \varepsilon_j \end{cases}; \quad (2)$$

where, b_j is the value, given by the j^{th} sensor, and P_i is the position of i^{th} agent in sector ε_j . In case of short-range sensing the sensing sector (ε_j) is valid only in the given R_0 radius.

$$\underline{\varepsilon}_j = \{P_i \mid |P_i - P_0| < R_0\}; \quad (3)$$

where, P_i is the position of i^{th} agent, and P_0 is the position of reference agent [13]. The surrounding environment of the reference robot is represented with the binary words *LRS* and *SRS*. We can say that two binary words are equivalent if the number of 1s and the position of 1s in relation to one another are identical (e. g. the words $\zeta_1=01100000$ és $\zeta_2=00000110$ are equivalent). It is observable that with shifting to left or right, or with circular operations we can get several equivalent words. Let us name these equivalent bits *stimulus* and label (ζ). The stimulus contains the description of the environment of the mobile robot [13].

In the perception model, the starting positions of the agents are already known (see Appendix 3). In an ideal case, the (d) distance is calculated from the time of

flight (t) and the spreading of speed of sound (v), in case of ultrasonic range measurements [14].

$$d = \frac{1}{2} v.t; \quad (4)$$

In this model the ideal case is considered, that is after checking the sensing segment's occupation, the distance calculations in x and y directions has been provided. The distance measurement model can be seen on Figure 3.

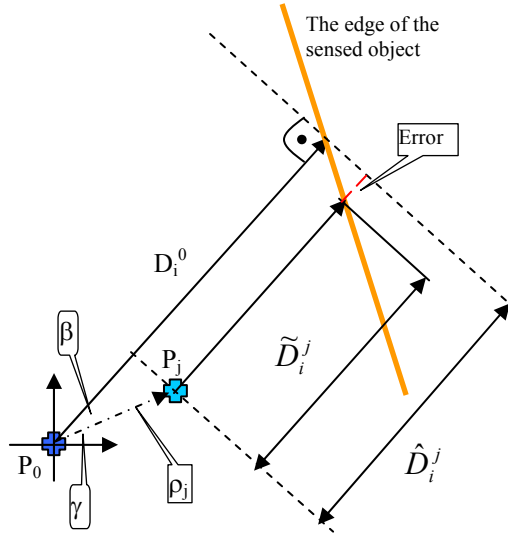


Figure 3

The mathematical model of *US* sensing

As a result of *WS* rasterizing (grid construction), an exact result of distance measurement is almost impossible. Unfortunately this is inconceivable in real environment, since we have to take into consideration the error (δ_D), see Figure 3.

$$\delta_D = |\tilde{D}_i^j - \hat{D}_i^j|; \quad (5)$$

where, \tilde{D}_i^j is the real distance, and \hat{D}_i^j is the evaluated one.

$$\hat{D}_i^j = D_i^0 - \rho_j \cdot \cos \beta, \quad i = 1..8, \text{ resp. } (1..16); \quad (6)$$

The distances measured with N ultrasonic sensors are stored in the L measuring-vector. The measuring-vector belonging to the P_0 location is: $L_0 \equiv [D_1^0, D_2^0, \dots, D_8^0]$. Besides, the evaluated distances in the model (after the grid association) belonging to the P_j position are stored in the distance-vector: $\hat{L}_j = [\hat{D}_1^j, \hat{D}_2^j, \dots, \hat{D}_8^j]$. The errors depend on the complexity of the environment and certainly on the map grid

width (see Figure 3). With reference to the members of L_j vector, for the sake of better evaluation results, the weighting vector (w_i) has been introduced. Regarding the distance between the robot location and P_j raster position, the weighting can be written as:

$$w_j = e^{-\eta \rho_j^2}; \quad (7)$$

where η is a positive constant, and ρ_j is the distance between P_0 - P_j locations (See Figure 3). Namely $w_j=1$ if the agent is exactly in the position P_j (in this case $P_j=P_0$).

4 The Potential Field

The creation of the artificial potential field (APF) has been done by the well-known repulsive force method [1], [13].

$$\begin{aligned} U_{ART}(x) &= U_{GOAL}(x) + U_{OBS}(x) \\ U_{GOAL}(x) &= -\frac{1}{2}k_p(x - x_{GOAL})^2 \\ U_{OBS}(x) &= \begin{cases} \frac{1}{2}\eta\left(\frac{1}{x} - \frac{1}{l_0}\right)^2; & \text{if } x \leq l_0; \\ 0 & \text{if } x > l_0; \end{cases} \end{aligned} \quad (8)$$

$$\vec{F}_{ART} = -\nabla[U_{ART}(x)];$$

where, U_{ART} is the APF, U_{GOAL} is the potential field spreading from the goal, U_{OBS} is the potential field of the obstacle, k_p is a positive gain, l_0 is a threshold limit beyond which are no repulsive forces, and η is a positive constant.

The potential field building

In case of validity of the next condition:

$$\forall i \in [1, N], \quad \hat{D}_i^j \geq 0; \quad (9)$$

the potential field is calculated from the \hat{L}_j vector. This condition is valid for the visibility of P_j position simultaneously. If the above mentioned condition is not valid, it means that the agent is on the obstacle, or is part of the obstacle. The evaluated value of the potential field at the location P_j in step “ t ” (if the above mentioned condition is valid), is:

$$\hat{U}_j^t \cong \sum_{i=1}^N e^{-\lambda \hat{D}_i^t}; \quad (10)$$

where λ is a positive coefficient. The potential field values, belonging to P_j position, measured by k^{th} mobile robot, at time “ t ”, are stored in set Ω .

$$\Omega_j^t = \{\hat{U}_j^{t1}, \hat{U}_j^{t2}, \dots, \hat{U}_j^{tk}\}; \quad (11)$$

In case if a member of Ω_j^t set equals zero, then is valid:

$$\begin{aligned} \Omega_j^t &= \Omega_j^{t-1} \cup \Theta; \\ \Theta &= \begin{cases} \hat{U}_j^{tk}, & \text{if for } \hat{L}_j - \text{is valid (9);} \\ 0, & \text{otherwise;} \end{cases} \end{aligned} \quad (12)$$

To the Ω_j^t set, is associated the following confidence weight vector:

$$W_j^t = \{w_j^{t1}, w_j^{t2}, \dots, w_j^{tk}\}; \quad (13)$$

where the normalized weight component of W_j^t is:

$$w_j^{-ti} = \frac{w_j^{ti}}{\sum_{n=1}^k w_j^{tn}}; \quad (14)$$

Finally an acceptable potential field value can be readily calculated as follows:

$$U_j^t = \begin{cases} \hat{U}_j^u, & \text{if } \exists i \in [1, k], w_j^u = 1; \\ \text{else: } \sum_{i=1}^k \hat{U}_j^u \cdot w_j^{-ti}; \end{cases} \quad (15)$$

5 Motion Mechanism

After the execution of sensing, measurements, and estimating the value of the potential field in position P_0 , the agent has to move to the next position to continue its measurements. This move can be applied as based on three motion selection [13]:

Directional1 – here the standard deviation of potential field is calculated in all (N) sensing sectors within the given maximum movement step (dm) at time “ t ” and “ $t-1$ ”. Moreover, the move in time “ $t+1$ ” will be calculated according to the motion direction (ϕ) and motion step (d_s). For the P_0 location of the robot at time “ $t+1$ ” can be written:

$$P_0^{t+1} = P_0^t + d_s \cdot e^{j\phi}; \quad (16)$$

Let us store the difference of the standard deviations of potential fields at the same sensing vector, at time “ t ” and “ $t-1$ ” in vector Δ . Then the i^{th} component of this vector is:

$$\Delta_i = \text{std}(\{l_{ij} \mid l_{ij} = U_{ij}^t - U_{ij}^{t-1}, \forall j \in \varepsilon_i, i = 1, 2, \dots, N\}); \quad (17)$$

Besides, let array Λ be the standard deviation of potential field for all locations in the same sensing sector at time “ t ”. For the i^{th} component of this vector can be written:

$$\Lambda_i = \text{std}(\{v_{ij} \mid v_{ij} = U_{ij}^t, \forall j \in \varepsilon_v, |j - i| \leq 1\}); \quad (18)$$

where ε_v is defined similarly like ε_i - see above. After that, the motion direction in sector i :

$$\begin{aligned} \phi_i \mid \Delta_i &= \max(\Delta_1, \Delta_2, \dots, \Delta_N); \\ \forall j, P_j &\notin \varepsilon_i; \end{aligned} \quad (19)$$

where P_j is the location, and ε_i is the sensing sector. Namely, the agent will select its direction of movement (ϕ) in “ t ” sector, based on the condition (19). The exact position, $P_0^{t+1}(x_0, y_0)$, within the selected sector, should satisfy the following condition:

$$(x_0, y_0) \mid \Lambda_i(x_0, y_0) = \max(\Lambda_1, \Lambda_2, \dots); \quad (20)$$

Directional2 – The strategy is almost the same as previously (see *Directional1*), the only difference being in selecting the exact position within the selected sector. The exact position selection is based on the minimum value of vector Λ .

$$(x_0, y_0) \mid \Lambda_i(x_0, y_0) = \min(\Lambda_1, \Lambda_2, \dots); \quad (21)$$

Limited random – The agent selects its motion direction and step size randomly, within the given limits.

$$\phi_i = \text{rand}([1..N]); \quad (22)$$

$$d_s = \text{rand}([1..d_m]); \quad (23)$$

The next question should be about how to make the potential field building process more effective. The answer lies in the appropriate behaviour mechanism.

6 Behaviour Selection Mechanism

Usually, in behaviour-based task execution at mobile robots, the next behaviour is very much influenced by the environment (as we know, the environment of the mobile robot is represented by stimuli; see above). In a general case, it exists as a set of behaviours out of which the best behaviour will be chosen by the algorithm, based on the environment's appraised measurements.

Behaviour in single agent environment – Let the primitive behaviour correspond to the direction of the 8 (or 16) ultrasonic sensors (see Figure 2b, $\{B_1, B_2, \dots, B_8\}$). In other words, the elementary motions are summarized in vector B , $B=[B_1..B_N]$. The values of this vector can be $B_i \in \{-1, 1\}$ in such a way that: $B_i=1$, if the agent is capable of executing the required move, else $B_i=-1$. The behaviour and the weighting vectors are:

$$B = \begin{bmatrix} B_1 \\ B_2 \\ \cdot \\ \cdot \\ B_N \end{bmatrix}; W = \begin{bmatrix} w_1 \\ w_2 \\ \cdot \\ \cdot \\ w_N \end{bmatrix}; \quad (24)$$

where, the values of weighting: $W_i=-1$, if $B_i=-1$, and in other cases the weighting is:

$$\sum_{i=1}^N W_i |_{w_i \neq -1} = 1; \quad (25)$$

The simplified behaviour selection process can be summed up as follows: Based on the sensing vector (see *LRS*, *SRS* mentioned above), the appropriate stimulus is given which triggers a condition for the behaviour selection mechanism. Next, as the output of the embedded learning mechanism (See Figure 4; dashed line), the near optimal behaviour will be selected. Let us have a few more words about the simplified learning mechanism. Any response to the sensing vector of the agent (what is nothing else than a stimulus, and the response to the stimulus, which is the behaviour) is represented by the varying weight. The learning step is the following: if the agent has selected the motion direction, then the components of the W vector will be updated. As a result of a series of updating, the outcome will be some more significant directions.

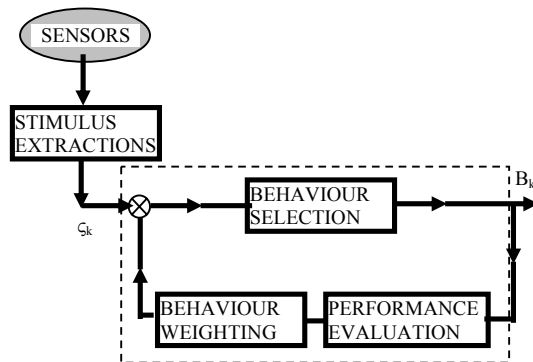


Figure 4

The simplified behaviour selection

The process: a stimulus (ζ_k) is chosen, to which the appropriate behaviour (B_k) belongs, next the motion is executed, and then the operation is appraised by weighting (W). The next stimulus-behaviour pair selection is based on this weighting vector, which produces a more effective motion mechanism.

Behaviour in a multi-agent environment – Let the agent to the stimulus (ζ_k) select the behaviour (B_k), at time „ t ”. After it, the robot learns, based on its local performance criteria. In the case if a common basis for behaviour selection is used, the agents can share their learned knowledge. The behaviour weight vector will be updated, based on the following:

$$W_{\zeta^k}^{t+1} = \text{normal}(\text{shape}(W_{\zeta^k}^t + \Delta W)); \quad (26)$$

where operator “*normal*” is normalizing the weight vector, and ΔW is an increment vector. The performance of operator “*shape*” is illustrated on Figure 5, where the updated weight vector passes through *function1*, and conditionally *function2* [15].

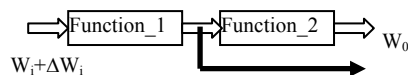


Figure 5

The “shape” operator

The *function1*:

$$w_0 = \begin{cases} 0, & \text{if } w_i < 0 \\ w_i, & \text{if } 0 \leq w_i \leq 1; \\ 1, & \text{if } w_i > 1 \end{cases} \quad (27)$$

The *function2*:

$$w_0 = \frac{\alpha}{1 + e^{-w_i}} - \psi; \quad (28)$$

where coefficients α and ψ influence the shape of the function. Then, the j^{th} component of the ΔW weight-increment vector is:

$$\Delta w_j = \begin{cases} \delta |_{E(B_k)}, & \text{if } j = k \\ 0, & \text{if } j \neq k \end{cases}; \quad (29)$$

where $E(B_k)$ is an evaluation of behaviour B_k and $\delta \in [-1, 1]$. At time $t=0$, the i^{th} component of the behaviour weight vector is:

$$w_i^0 = \begin{cases} -1, & \text{if } B_i = -1 \\ \frac{1}{\beta}, & \text{otherwise} \end{cases}; \quad (30)$$

where β is the number of the feasible behaviours.

Behaviour selection mechanism – this mechanism assigns the behaviour weight to the corresponding behaviour (*sel*: $W \rightarrow B_k$). The behaviour selection mechanism can work in two ways:

a) selection based on the *probability of the behaviour weight vector distribution*:

$$B_{\text{sel}} = B_k |_{P(w_k)}; \quad (31)$$

b) selection based on maximum weight:

$$B_{\text{sel}} = B_k |_{w_k = \max(w_1, w_2, \dots, w_N)}; \quad (32)$$

After behaviour selection the agent moves along in the selected direction, with step size d_0 . Let us mark the position at time “ t ” $\rightarrow P^t$, and at time “ $t+1$ ” $\rightarrow P^{t+1}$. In this case this whole process (action) can be defined as [15]:

$$P_i^{t+1} = \text{action}(B_k, d_0, P_i^t); \quad (33)$$

The next step in behaviour based robotics was, mainly in environments where multi-agent robot groups occur, that for the selecting of near optimal behaviour genetic algorithms and/or neural networks were used. In this paper the near optimal behaviour is selected through a genetic algorithm which is working with two fitness functions. The essence of behaviour-control is that the agents are organized into robot groups, the efficiency of the individual agents is evaluated, and then based on this evaluation the next “action” is selected. In “action”, the direction selection is considered with two situations. The first is the *spatial distribution* of agents (when the distance between two agents i and j is less than

the given threshold distance: $d_{ij} \leq R_2$). The second is the *area coverage*, when $d_{ij} > R_2$.

Spatial distribution - For the reference robot i , and m – neighbouring robots, is valid:

$$i, \forall m \in [1, M], m \neq i, d_{im} \leq R_2;$$

$$\text{spat}_{m=1}^{m_d-1} \text{distr} \frac{e^{-j\gamma_m}}{d_{im}} \cong \xi_i e^{j\theta_i}; \quad (34)$$

Area coverage:

$$i, \forall m \in [1, M], m \neq i, d_{im} > R_2;$$

$$\text{area}_{n=1}^N \text{cov er} \frac{e^{\frac{j2\pi n}{N}}}{D_n^0} \cong \xi_i e^{j\theta_i} \quad (35)$$

where, d_{im} is the distance between robot i and m , then m_d is the number of group robots inside of R_2 threshold limit, γ_m is the relative angle of motion direction (see Figure 3, where m^{th} agent is moving to P_j location and $\theta_i \neq \rho_i$), and D_n^0 is the n^{th} component of L_0 vector.

Let the significant proximity direction a time t be θ_i^u (that is the direction of the i^{th} agent's motion is u , where u is one of the 8/16 sensing sectors: $u \in [1..N]$). There exists a probability vector ϖ_i , where the components express the efficiency of (34) and/or (35) if the motion was executed. This ϖ_i vector can be written as follows:

$$\varpi_i = [\phi_1, \phi_2, \dots, \phi_N];$$

where $\phi_k \in [0, 1]$ and $\sum_{k=1}^N \phi_k = 1$; . In case if the agent in the next motion (at time $t+1$) selects a different motion direction “ v ”, ($v \in [1..N]$), denotes it by θ_i^v , then the k^{th} component of the ϖ_i vector will be updated as follows:

$$\phi_k^{t+1} = \frac{\phi_k^t + \delta}{1 + \psi}; \quad (37)$$

where

$$\delta = \begin{cases} \psi, & \text{if } k = v - u + 1 \\ 0 & \text{otherwise} \end{cases};$$

where ψ is a positive coefficient. As a result of permanent updating some motion directions become more significant than others.

7 Genetic Algorithm

In this system the near optimal motion direction is selected through a GA. The simplified operation of the genetic algorithm works as follows:

- The fitness of each member in a GA population is calculated according to an evaluation function (*fitness functions*), which measures how well the individual performs.
- Individuals performing well are propagated in proportion to their fitness; on the other hand, the poorly performing members are reduced or eliminated completely.
- By exchanging the information between members it is possible to create new search points, by which the population explores the search space and converges in an optimal solution.

To find and represent these new search points, the GA uses its operators. Several operators are known, but the three most frequently used ones are: *reproduction* (selects the fittest members and copies them exactly; *crossover* (swapping some part of their representations.); *mutation* (prevents the loss of information that occurs as the population converges on the fittest individuals).

In every step the mobile robot checks its environment, then according to the vectors (34), (35) and the probability vector ϖ_i , (what is the result of the learning process), next motion direction is selected. In compliance with this probability vector, the GA *population* will be determined on the basis of this ϖ_i vector. The structure of this whole system can be seen on Figure. 6.

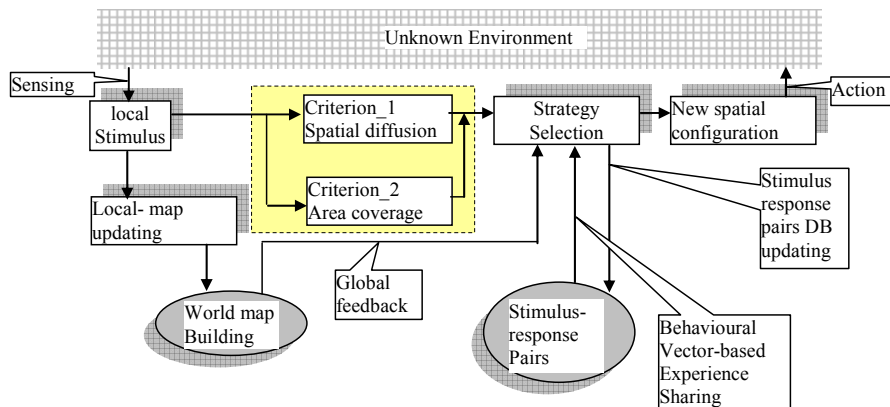


Figure 6

The architecture of the system

Inside the dashed lines the GA module can be seen. For *chromosome representation* let us define a 2D coordinate frame, centred at the current location of the agent, and square bounded, where the sides are determined by the maximal step size ($2 \cdot d_m + l$). In accordance with this, the local region for the agent's next movement is: $x', y' \in [-d_m, d_m]$. Moreover, let us suppose that $(2 \cdot d_m + l)$ corresponds to a binary string L , following from the fact that location within the local region can be represented by two binary values, with length L . For the behaviour evolution of a single agent, we can use a chromosome of length $2L$, and for a group with M robots it is $2LM$.

The fitness functions – In this system 3 fitness functions are used, out of which the 1st is the *general* fitness function (f_g), used for exploring less confident regions and for avoiding the repetition of other agents' work [13].

$$f_g = \prod_{i=1}^m \left\{ (1 - \max \{w_i^{tk}\}) \prod_{j=1}^{m_e} \sqrt[4]{d_{ij} - R_1} \right\}; \quad (38)$$

where w_i^{tk} is the confidence weight corresponding to the location of agent i , then m is the number of agents grouped together during one evolutionary movement step, m_e is the number of robots which do not belong to m , that is the inter-distance between two robots i and j is greater than R_1 ($d_{ij} > R_1$). The 2nd and 3rd fitness functions are special functions, and correspond to the criteria of multi-robot spatial diffusion and area coverage, see relations (34), (35).

$$f_1 = \prod_{i=1}^{m_d-1} \prod_{j=i+1}^{m_d} \sqrt{d_{ij} - R_2}; \quad (39)$$

$$f_2 = \frac{\sqrt{\Delta v}}{\prod_{i=1}^{m_c} \xi_i};$$

where, m_d is the number of robots with inter-distances $d_{ij} > R_2$, where m_c is the number of area-covering robots, Δv is the number of location visited by agents m_c and ξ_i is the proximity distance between robot i and other agents. The complete fitness function can be defined as follows:

$$F = \begin{cases} f_g \cdot f_1, & \text{for spatially diffusing robots} \\ f_g \cdot f_2, & \text{for area - covering robots} \end{cases}; \quad (40)$$

8 The System Error

At simulation systems the question of system errors is not avoidable. There are several possibilities to error definition. Usually at models related to mobile robots, we can define errors arising from: *a)* non-ideal mathematical models, *b)* discretization of the work space. Of course each of these errors is repairable. The 1st is repairable by the more exact mathematical models, and the 2nd one by scaling. In the present system, the error arising from discretization of the WS is formally defined as follows [13]:

$$\varepsilon' \cong \sqrt{\frac{1}{K} \sum_{j=1}^K (\tilde{U}_j^t - \hat{U}_j)^2}; \quad (41)$$

where K , denotes the total number of locations in the potential field map, and \tilde{U}_j^t, \hat{U}_j belongs to the estimated and true potential field values at position $P_j(x_j, y_j)$.

Conclusion

It is a simulation system for potential field building process in the multi-agent domain that has been described in this paper. The aim is to provide an opportunity for studying the behaviour vectors of agents, for the sake of selecting the near-optimal behaviour.

The aims stated in the first section (aims & motivation) have been fulfilled. A working multi-agent based simulation model has been created and the features mentioned in the second section (introduction), namely “more efficiency”, have “more or less” been realised as well. Let us look at one of the most important elements of the list: “faster and more accurate”. An unambiguous answer is given in Appendix 5, where on Figure 12 it is clearly seen that in case of a single agent, the system was not able to create the potential field in 30 steps, while in case of 3 or 6 agents (Figures 13 and 14) it was accomplished successfully. Another conception of mine was that the basic problems of trapping and oscillation will be solved by a GA algorithm. The idea has proved to be successful too, as seen in the 5th interval on Figure 15b in Appendix 6, where in case of single agent the problem is clearly visible, while on Figures 16 and 17 (in case of 3 or 6 agents) this problem is not present. The reason why I used the words “more or less” above is because I expected slightly better results from the aspect of area coverage. In my view, in case of 6 agents it is possible to improve the area coverage by tuning the GA parameters and fitness functions.

Appendices

This paper contains 6 appendices. In Appendix 1 the geometrical 2D map of the WS and its exact potential field can be seen. Appendix 2 includes tables with system parameters, GA parameters and computer parameters, used in the simulation. In Appendix 3 the starting positions of mobile agents can be found, while Appendix 4 contains the table and graph of running times with different number of agents and sensors. In Appendix 5 the resulting PFs are seen, also built up by different number of agents and sensors on the agents, while in Appendix 6 the wandered trajectories are represented in 6 intervals.

Appendix 1

The Workspace and its exact potential field

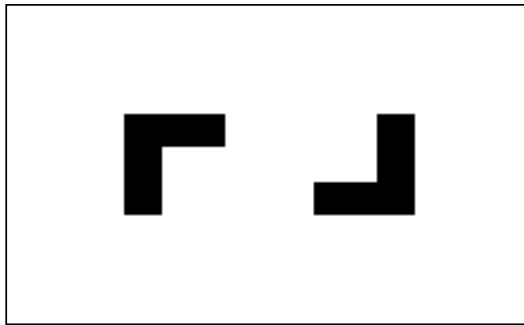


Figure 7

The *WS* in 2D with two obstacles

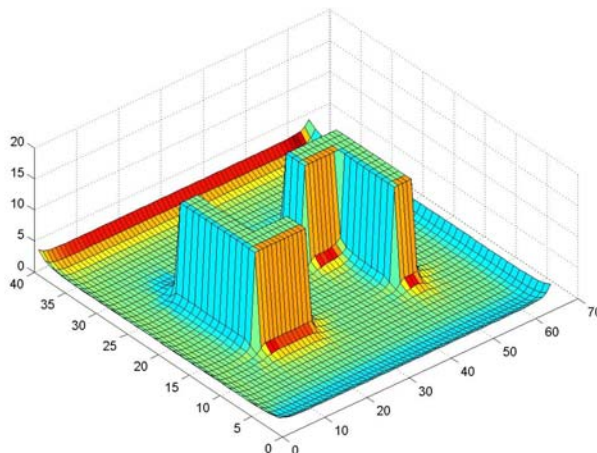


Figure 8

The exact *PF* of *WS*

Appendix 2

Tables of *parameters* used in the system and in the *GA*

Table 1
System parameters

Description	Unit	Value
The loaded <i>WS</i> size	unit	195 x 120
<i>WS</i> resolution (grid width)	grid	3
Normalised <i>WS</i>	$\frac{X}{GridWidth}; \frac{Y}{GridWidth}$	40 x 65
Maximum movement step (d_m)	location	7
Behaviour-vector increment		0,2
Threshold distance <i>R1</i>	grid	10
Threshold distance <i>R2</i>	grid	15

Table 2
Parameters used in *GA*

Description	Unit	Value
Robot description (in <i>GA</i>)	bit	8
Population size (<i>P</i>)		20/3045/65/90/120
Generations per step		8/12/18/26/36/48
Crossover probability (p_c)		0,6
Mutation probability (p_m)		0,1/0,05/0,005

Table 3
Computer parameters

Description	Unit	Value
Operation system		WIN-XP, prof.
Processor clock	GHz	1,60
RAM size	Mbyte	512

Appendix 3

The starting positions of the mobile robots

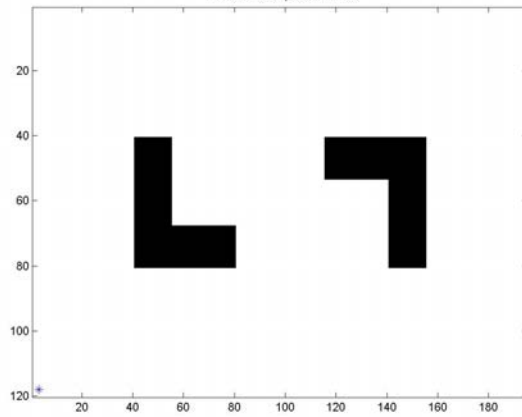


Figure 9
Starting position of 1 agent

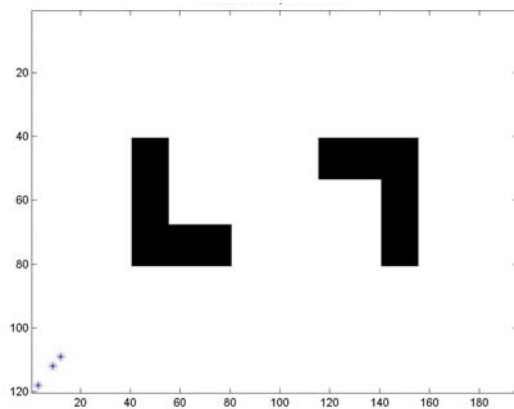


Figure 10
Starting positions of 3 agents

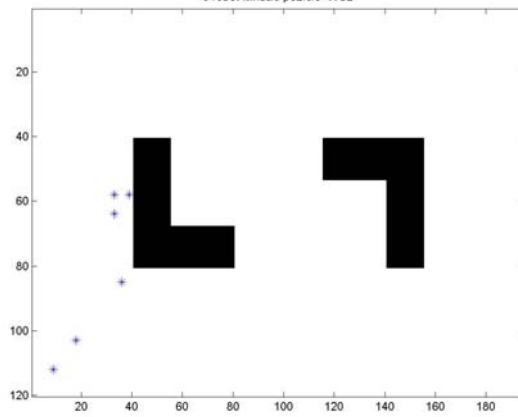



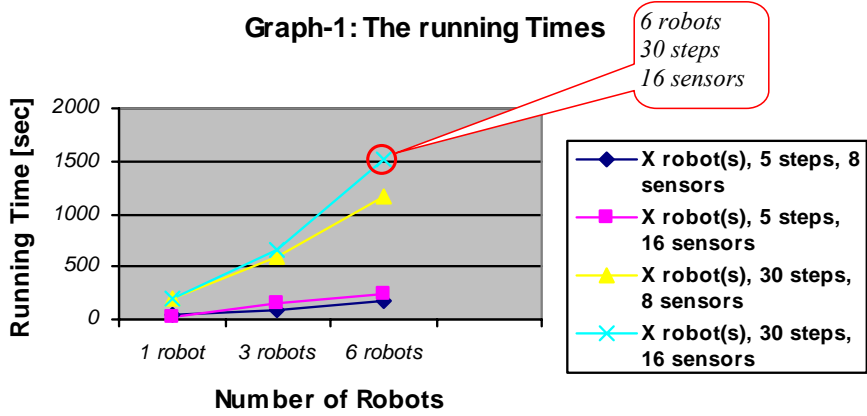
Figure 11
Starting positions of 6 agents

Appendix 4

Table of results

Table 4
The Running Times

	<i>Running Times</i> (in seconds)			
	5		30	
Number of sensors	8	16	8	16
Number of robots ↓				
1	37	31	208	195
3	77	150	595	657
6	177	240	1162	1521



Appendix 5

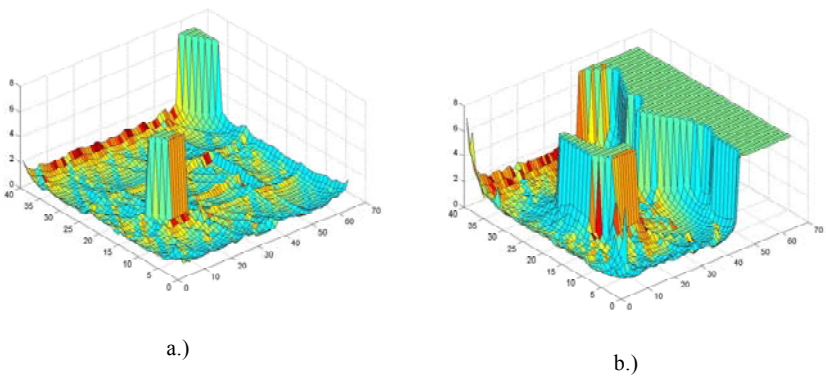


Figure 12
The resulting *PF* of *WS*, built up by *1* agent in *30* steps
a) 8 sensors, b) 16 sensors

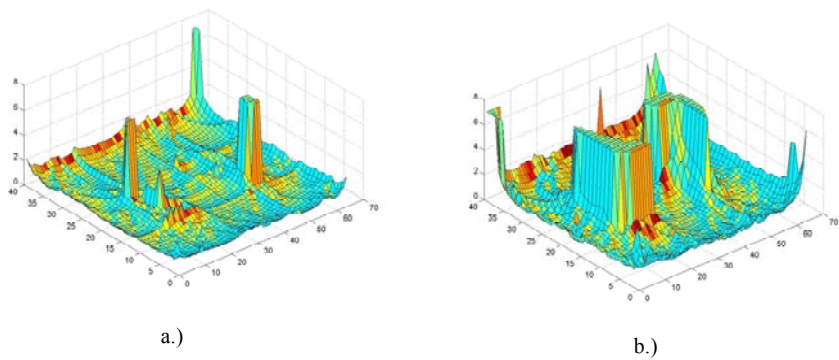


Figure 13
The resulting *PF* of *WS*, built up by *3* agents in *30* steps:
a) 8 sensors, b) 16 sensors

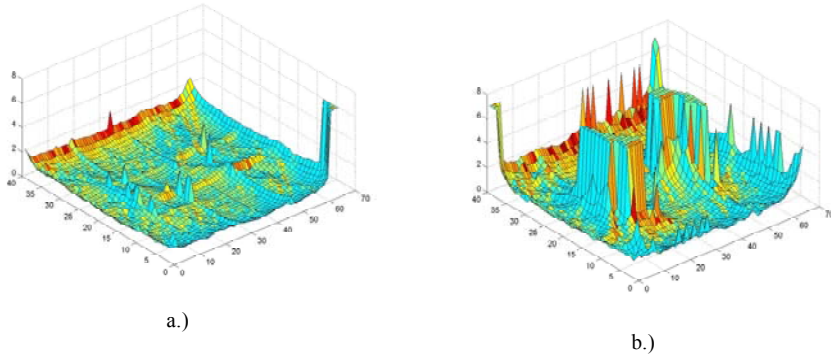


Figure 14
The resulting *PF* of *WS* built up with 6 agents in 30 steps:
a) 8 sensors, b) 16 sensors

Appendix 6

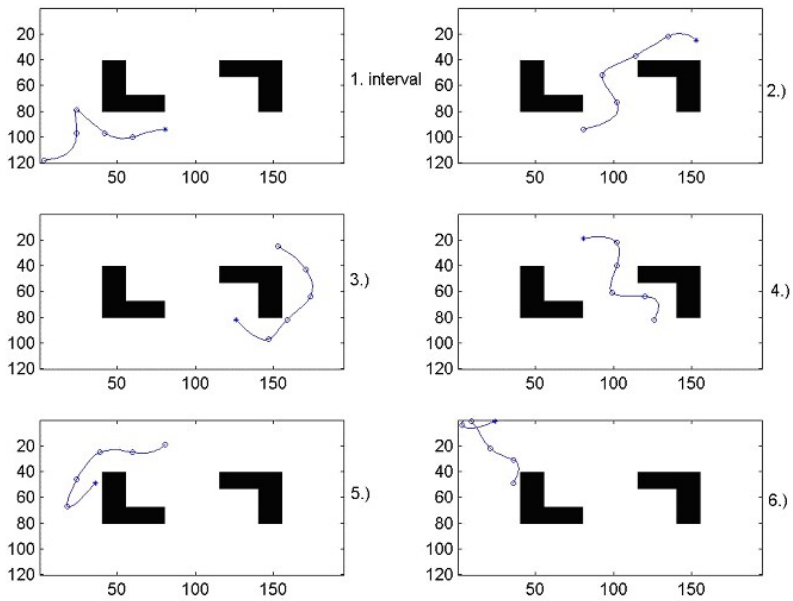


Figure 15a
Trajectories of 1 agent, in 6 intervals, MaxRunStep=30, 8 sensors

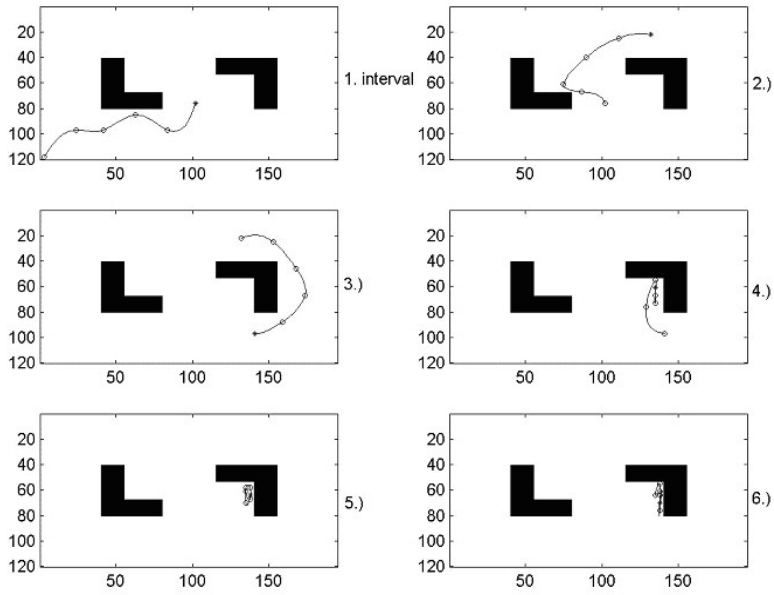


Figure 15b

Trajectories of 1 agent, in 6 intervals, MaxRunStep=30, 16 sensors

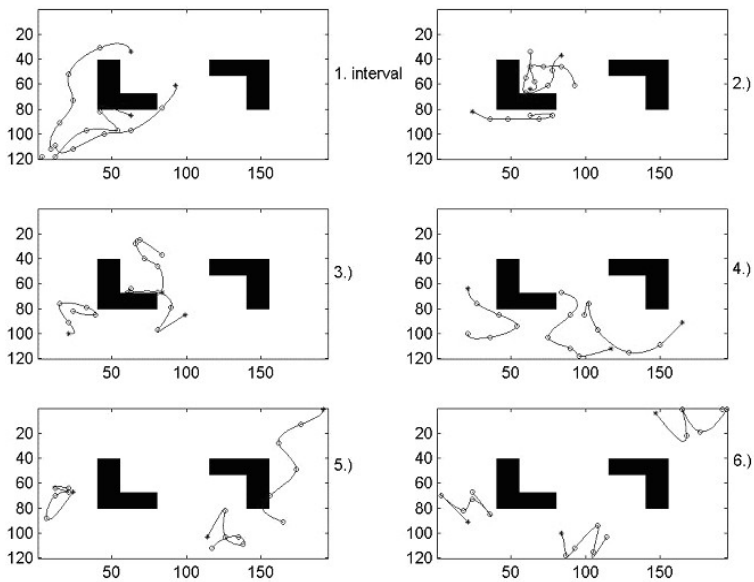


Figure 16a

Trajectories of 3 agents, in 6 intervals, MaxRunStep=30, 8 sensors

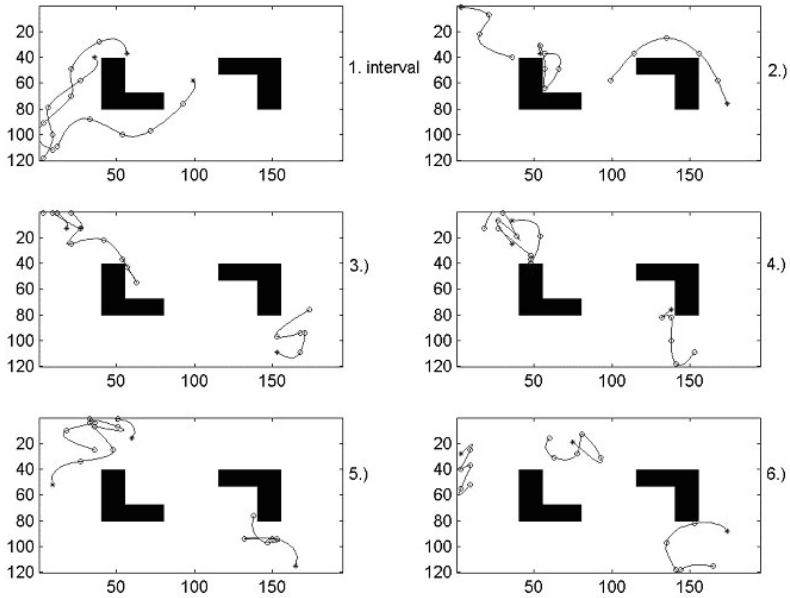


Figure 16b

Trajectories of 3 agents, in 6 intervals, MaxRunStep=30, 16 sensors

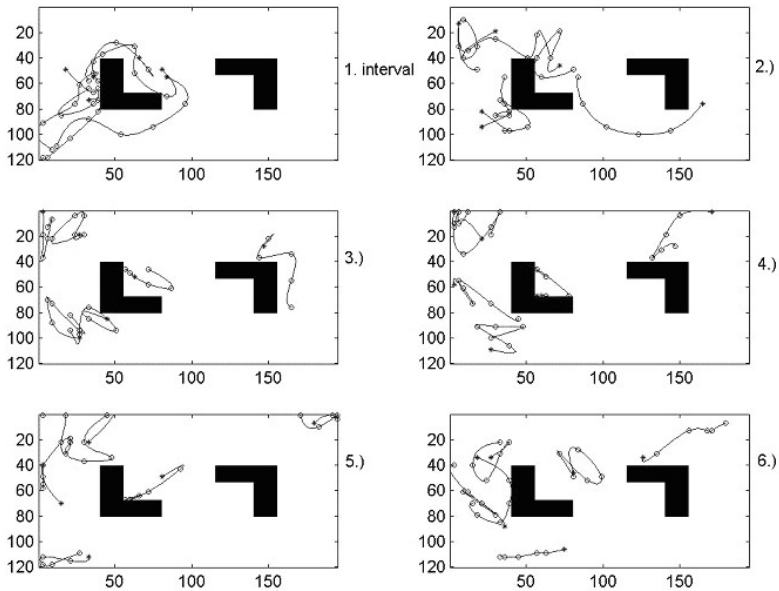


Figure 17a

Trajectories of 6 agents, in 6 intervals, MaxRunStep=30, 8 sensors

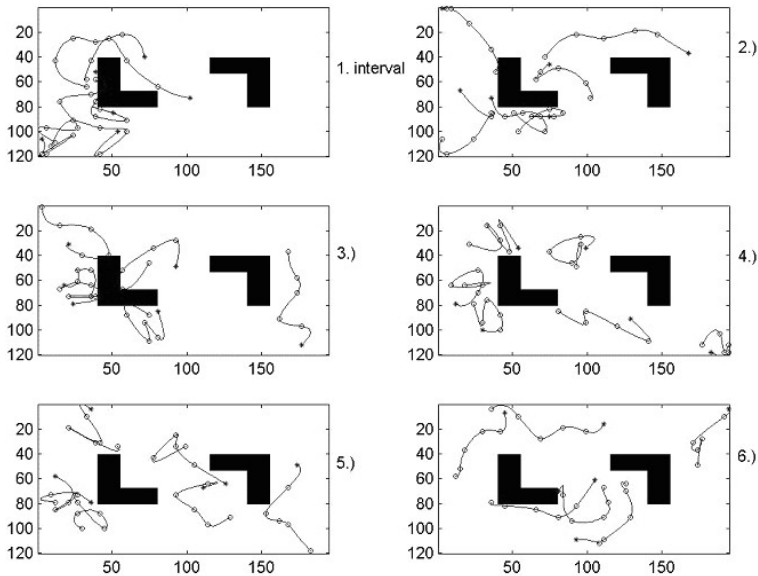


Figure 17b

Trajectories of 6 agents, in 6 intervals, MaxRunStep=30, 16 sensors

References

- [1] S. Mizik, P. Baranyi, P. Korondi, M. Sugiyama: *Virtual Training of Vector Function-based Guiding Styles*; Transactions on Automatic Control and Computer Science, ISSN 1224/600X Vol. 46(60) No. 1, pp. 81-86, 2001
- [2] J. Vaščák: *Navigation of Mobile Robots Using Potential Fields and Computational Intelligence Means*; Acta Polytechnica Hungarica, ISSN 1785-8860, Vol. 4, No. 1, pp. 63-74, 2007
- [3] I. Nagy, A. L. Bencsik: *A Simulation System for Behaviour-based Potential Field Building in Multi-Agent Mobile Robot System*; Proc. of the 3rd IAESTED International Conference on Computational Intelligence, pp. 7-12, ISBN: 978-0-88986-672-0, Canada, 2007
- [4] T. Fukuda, S. Nakagawa: *A Dynamically Reconfigurable Robotic System*; In Proc. of the International Conference on Industrial Electronics, Control and Instrumentation, pp. 588-595, Cambridge, MA, 1987
- [5] H. Asama, A. Matsumoto, Y. Ishida: *Design of an Autonomous and Distributed Robot System: ACTRESS*; In proc. of the IEEE/RSJ, International Workshop on Intelligent Robots and Systems, pp. 283-290, Tsukuba, 1989

- [6] D. Goldberg, M. J. Mataric: *Coordinating Mobile Robot Group Behaviour Using a Model of Interaction Dynamics*; Proc. of the 3rd Int. Conf. on Autonomous Agents, pp. 100-107, Seattle, 1999
- [7] P. Pirjanian, M. J. Mataric: *Multi-Robot Target Acquisition Using Multiple Objective Behaviour Coordination*; In proc. of the IEEE International Conference on Robotics and Automation, San Francisco, 2000
- [8] M. J. Mataric: *Behaviour-based Robotics as a Tool for Synthesis of Artificial Behaviour and Analysis of Natural Behaviour*; Trends in Cognitive Science, 2(3), pp. 82-87, 1998
- [9] Gh. Tecuci: *Building Intelligent Agents*; Academic Press, San Diego, Cal., 1998
- [10] J. Kelemen: *Agents from Functional-Computational Perspective*; Acta Polytechnica Hungarica, ISSN 1785-8860, Vol. 3, No. 4, pp. 37-54, 2006
- [11] B. Frankovič, T-T. Dang, I. Budinská: *Agents' Coalitions Based on a Dynamic Programming Approach*; Acta Polytechnica Hungarica, ISSN 1785-8860, Vol. 5, No. 2, pp. 5-21, 2008
- [12] T. Fukuda, G. Iritani: *Construction Mechanism of Group Behaviour with Cooperation*; In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 535-542, Pittsburgh, 1995
- [13] J. Liu, J. Wu: *Multi-Agent Robotic Systems*; CRC Press LLC, ISBN 0-8493-2288-X, 2001
- [14] G. Dudek, M. Jenkin: *Computational Principles of Mobile Robotics*; Cambridge University Press, ISBN 0 521 56021 7, 2000
- [15] I. Nagy: *Genetic Algorithms Applied for Potential Field Building in Multi-Agent Robotic System*; In Proc. of the IEEE International Conference on Computational Cybernetics, ICC 2003, pp. 105-108, Siófok, Hungary, 2003