From Exploring to Optimal Path Planning: Considering Error of Navigation in Multi-Agent Mobile Robot Domain

István Nagy

Óbuda University, Bánki Donát Faculty of Mechanical and Safety Engineering
Institute of Mechatronics and Vehicle Engineering
Népszínház u. 8, H-1081 Budapest, Hungary
nagy.istvan@bgk.uni-obuda.hu

Abstract: In this paper a complex path planning model is presented. The model building starts by mapping the totally unknown environment, then the geometrical map of the "work space" is built-up. Afterwards, based on the determined error of navigation, the reduced free space of the environment is created. The path planning process is begins with this reduced environment. At first it determines some cruises (corridors) for the multi-agent traffic, and then comes the builds up the graph-map of the cruises. Based on the weighting of the graphs, the algorithm selects the time-optimal, dynamically optimal, and collision-free path, from among the possible ones, connecting the starting and docking positions. The final path is created with fitting some B-Spline curves to the selected graph-like one in consideration of movements of the other agents.

Keywords: mobile robot; multi-agent; path planning; map building

1 Introduction

Multi-agent mobile robot systems play a significant role in robotic systems. Collaborative robots started their career with the beginning of Artificial Intelligence (AI). Among the first collaborative robots' projects the different competitions in robotic soccer can be mentioned [1], [2] possibly having a peak in an international joint project called "*RoboCup Coach Competition*" [3], [4].

While the robot soccer leagues gives an exciting task to the researchers working in AI, there exist several other fields of research where the multi-agent systems can be useful. The "teamwork" plays a larger role, not only in human work, but in the robotics world, too.

Nowadays, "*swarmbots*" is a very often-used expression (see e.g [5], [15]). These are collaborative robots, which can autonomously execute some predefined task or series of tasks. The control of the agents can be centralized or decentralized. In case of decentralized control, the communication between the agents is more frequent, because the agents have to know about the activities of other agents and have to plan their own actions based on the behaviour of other ones. In the case of centralized control, the agents are communicating with a central control unit that has more or less accurate information about the positions and activities of the agents. The swarm of "*mini-robots*" could be perfectly used for exploring unknown environments where human beings are unable to get in. Each agent should be equipped with sensors and an antenna and would be able to map the environment. But here is the question of with regards to hardware vs. software. What we need is a small and "smart" agent with high mobility. In contrast with this, here are the physical dimensions of sensors, accumulators, motors, and locomotion. Based on these parameters, the minimal physical dimension of the agent can be estimated, which, in our opinion, has not been typical for the "*swarm-technology*" as of yet. On the other hand, nowadays, a lot of control strategies exist in swarm-bot theories. Can we tell that the theory preceded technology in this case? For this reason, we decided to use the expressions *multi-agent system* (MAS) and *self-organization* instead of swarm-bot or swarm-technology in this paper.

In this paper, the so-called 'centralized control theory' is improved. A multi-agent mobile robot system is used for exploring an unknown environment by using a new approach. The agents are equipped with 16 ultrasonic (US) sensors equally placed around the body. The process takes the difference between the short and long range activation of the sensors. The equipment of the agents, the ultrasonic measuring system used in this paper is similar to that of the system introduced in [6]. The main difference between the two approaches is in the map building process. The number of agents and the work space (WS) are also different. While in the previously mentioned paper the motion strategy was implemented with the help of some genetic algorithm (GA) based module, in this project the motion strategy of the agents is calculated using a *remote host* computer. Moreover, in this paper the process is not finished with the creation of the map, but continues with path planning based on the navigation's error defined in the multi-agent domain.

2 The Multi-Agent System Used for Potential Field-based Map Building

As it can be seen in Figure 1 the multi-agent system (MAS) used for potential field (PF) based map building is not a pure distributed or centralized system. The independent agents can communicate with each other, but the sensed data is sent to the *remote host* computer, i.e., the sensed data is shared with all of the agents.

As result, a global potential map of the environment can be created on the remote host computer, and since the agents send their current position to the remote host, the motion strategy is computed by the remote host, as well. The theory of this map building process was initiated by Liu and Wu in [7]. In this paper, based on their theoretical results, this approach is improved, extended, and implemented in *MATLAB* environment. The novelties presented in this paper include the extension of the map building with the path planning process, as well. The movement strategy is derived from the distance measurement results taking into consideration the current locations of the agent vs. obstacles and non-visited locations. We assume that that the agents, based on the distance and orientation details of their movements, are capable to determine their actual position. The behaviour of the agent (the direction of the next step) is selected after the processing of these (measured distance) parameters.

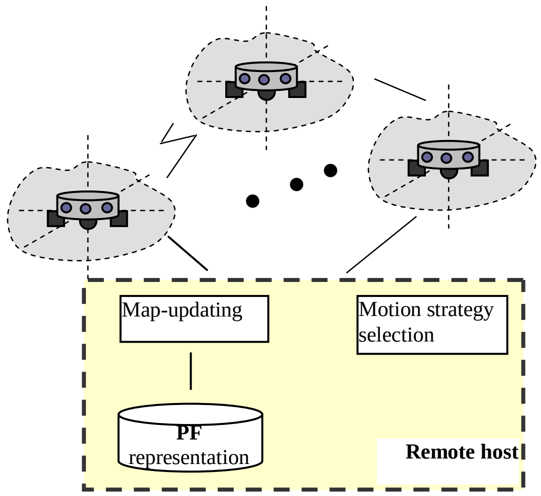


Figure 1.
The structure of the multi-agent system

The process of map building in pseudo codes can be written as follows:

```
Until non-visited location$\neq 0$
    Begin
        move to new location {$\mathbf{P_0}(x_0,y_0)$}
        take 16 measurements {$\mathcal{L}_0=[\mathcal{D}_1^0,..,\mathcal{D}_{16}^0]$}
        send location and measured data to the remote host
        associate neighbours, {$\mathbf{P}_j(x_j,y_j)$}
        calculations
        update map
        next motion planning
    End
```

2.1 The Coordinate System of the Agents and the Distance Calculations

Generally, the kinematics of a single agent, in vectorial form, can be expressed with the following equation, in the *x,y* coordinate plane:

$$\mathbf{P}(j) = \mathbf{F}(\mathbf{P}(0), \mathbf{u}(0)) + \mathbf{v}(0); \tag{1}$$

where ***F(P(0)),*** ***u(0)*** denotes the (non-linear) state transition function and ***v(0)*** is the noise source assumed during the measuring of the actual position. The model applied here ignores the measurement noises, i.e., ***v(0)=0***. The control input is $\boldsymbol{u(0)}=[T(0),\Delta\theta(0)]^T$, where *T(0)* stands for the distance between points ***P(0)*** and ***P(j)*** and *Δθ(0)* marks the change of the orientation of the agent. Considering these conditions the new location of the agent, in scalar form, can be expressed by

$$\left[\frac{x_j}{y_j}\right] = \left[\frac{x_0}{y_0}\right] + T(0)\left[\frac{\cos(\Delta\theta_0)}{\sin(\Delta\theta_0)}\right] \tag{2}$$

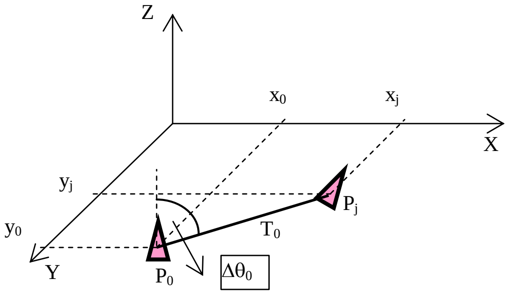


Figure 2.
The coordinate system of the agent and the distance calculation

2.1.1 Proximity Measurements and Distance Association in a Neighbouring Region

The agents measure the distances of their initial position (P0) and the surrounding obstacles with their 16 US sensors and the data is stored in a vector $\mathcal{L}^0 \cong [\mathcal{D}_1^0,..,\mathcal{D}_{16}^0]$. Based on these distance measurements the locations of the obstacle-free areas can be evaluated too. It is known that the measuring gives an accurate result if the direction of sensing is parallel with the plane of the movement (usually this is set) and also perpendicular to the plain of the sensed obstacle. Unfortunately, in many of the cases this latter cannot be ensured. For that reason, the distance measured in the next step (distance measured from ***P(j)*** position) will be evaluated based on:

$$\hat{D}_i^j = D_i^0 - T_0 \cos\beta; \quad i = [1,..,16] \tag{3}$$

where β is the angle between the sensing directions of positions ***P(0)*** and ***P(j)***. Set out from this, the estimated proximity values in location ***P***(*j*) are stored in vector $\hat{L}_j \cong [\hat{D}_1^j,..,\hat{D}_{16}^j]$. Let D_i^j denote the true distance of the obstacle from ***P(j)***. In this case, the difference between the measured and true distances equals the error in direction i : $\varepsilon_i = | D_i^j - \hat{D}_i^j |$. For a better approximation of estimated distance, a weighting function has been introduced for the elements of vector $\hat{L}_j$. This weighting function is a function from the agent's position to the location ***P(j)***. The weighting must be equal to 1, when the robot is exactly in ***P(j)*** position.

$$w_j = \exp(-\eta T_0^2) \tag{4}$$

where η is some positive gain factor and T_0 is the distance, see Figure 2.

2.2 Self-Organizing and Potential Field Calculation

The potential field is calculated based on well-known repulsive forces, which are derived from the measured distances. It means that the base for the PF calculation depends on vector $\hat{L}_j$ and its components, with the following condition:

$$[\hat{D}_1^j,..,\hat{D}_{16}^j] \geq 0 \tag{5}$$

The potential field, in location ***P(j)***, is calculated from the data of each sensors (*i*) of each robot (*r*) in each step (*k*). The estimation of the amplitude of the potential is:

$$\hat{u}_j^k \cong \sum_{i=1}^{16} \exp(-\lambda \hat{D}_i^j) \tag{6}$$

where λ is a positive gain. If at some locations condition (5) is not satisfied, the process discards them or assumes an obstacle. At step k the set of calculated potential amplitudes, $\Omega_j^k \cong [\hat{u}_j^{k1},..,\hat{u}_j^{kr}]$, is calculated based on:

$$\Omega_j^k = \Omega_j^{k-1} \cup Q \tag{7}$$

where:

$$Q = \begin{cases} \hat{u}_j^{kr}; & if \quad \hat{L}_j \ satisfies \ (5) \\ 0; & otherwise \end{cases} \tag{8}$$

Because of confidence of the estimated results, a weighting vector $W_j^k \cong [w_j^{k1},..,w_j^{kr}]$ is associated to $\boldsymbol{\Omega}_j^k$ and the acceptable PF value is calculated as:

$$u_j^k = \begin{cases} \hat{u}_j^{kr}; & if \quad \forall i \in [1,r], \quad w_j^{ki} = 1; \\ \sum_{i=1}^{r} \hat{u}_j^{kr}.\overline{w}_j^{ki}; & otherwise \end{cases} \tag{9}$$

where $\overline{w}_j^{ki}$ is a normalized weight component of $\boldsymbol{W}_j^k$ and calculated as follows:

$$\overline{w}_j^{ki} = \frac{w_j^{ki}}{\sum_{n=1}^{r} w_j^{kn}}; \tag{10}$$

2.3 The Motion Selection Process

The agents can apply one of the three variants of the motion defined in [7]. In each of the three variants the next movement is calculated based on:

1. motion direction (ϕ)
2. motion step (d_s)

The location of the agent in step ($k+1$) can be written:

$$P_0^{k+1} = P_0^k + d_s.e^{j\phi}; \tag{11}$$

2.3.1 Motion: Directional-1

This motion is derived from the standard deviation (*std*) of the PF map for all sensing sectors (16 sectors) within a given maximum step length (d_m) at steps *k-1*, and *k*. Let Vector $\boldsymbol{\Delta}$ store the standard deviation of the differences between the PF values at step *k* and *k-1* in each of the (16) sensing sectors. For component *i* it takes

$$\Delta_i = std(\{l_{ij} \mid l_{ij} = u_{ij}^k - u_{ij}^{k-1}, \forall j \in \varepsilon_i, i = 1,2,..,16\}); \tag{12}$$

Let vector $\boldsymbol{\Lambda}$ store the standard deviation of the PF values at step *k* for *all locations* in the same sensing sector:

$$\Lambda_i = std(\{v_{ij} \mid v_{ij} = u_{ij}^k, \forall j \in \varepsilon_v, \quad \mid j - i \mid \leq 1\}); \tag{13}$$

In equations (12), (13) ε_i, ε_v denote sensing sectors *i* and *v*, respectively. Finally, the robot selects its movement direction (ϕ_i) in sensing sector *i* so that it satisfies:

$$\begin{gathered} \phi_i \mid_{\Delta_i} = \max(\Delta_1,..,\Delta_{16}) \\ \forall j, P_j \notin \tau_i, \end{gathered} \tag{14}$$

where τ_i denotes sensing sector *i* and P_j stands for location *j*. After determining the movement sector, the agent will choose the exact location within the selected sector $\boldsymbol{P}_0^{k+1}(x_0,y_0)$ and the PF value around this location: $(x_0, y_0) \mid_{\Lambda_i(x_0,y_0)} = \max(\Lambda_1,..,\Lambda_{16})$. (15)

2.3.2 Motion: Directional-2

This strategy is almost the same as the strategy detailed in sub-subsection 2.3.1, except that the exact location of agent k+1, $\boldsymbol{P}_0^{k+1}(x_0,y0)$, within the selected sector, is chosen based on the minimum criteria of vector $\boldsymbol{\Lambda}$, that is:

$$(x_0, y_0)|_{\Lambda_i(x_0,y_0)} = \min(\Lambda_1,..,\Lambda_{16}); \tag{16}$$

2.3.3 Random Motion Strategy

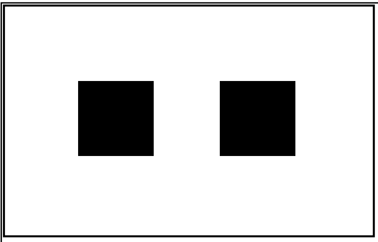
According to this strategy the agent selects its motion randomly. The direction of movement is selected randomly among the sensing sectors ([1,..,16]). The movement's step length is also randomly selected in the range of (*1, max. movement step*) ([1,..,d_m]).

$$\begin{aligned} \phi i &= rand([1..16]); \\ ds &= rand([1..dm]); \end{aligned} \tag{17}$$

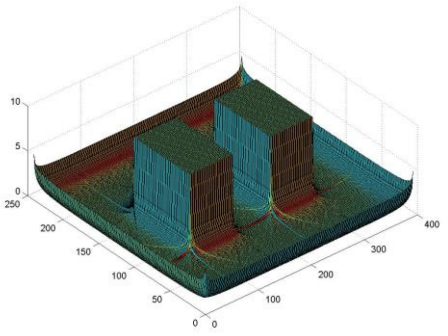
2.4 Case Study 1 - PF created with 6 Robots

In the following, the effectiveness of the PF creation is illustrated by an example. In this simulation, six robots explore an unknown environment. The environment is represented by a *bmp* image, which is used only for validation of the results.

The potential field is created in MATLAB SW environment.



a.)
The original Work Space



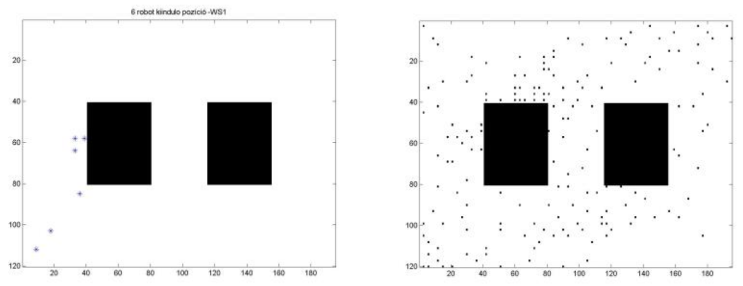
b.)
The created Potential Field

Figure 3.
The original WS in 2D and the created PF in 3D

The program executed 40 cycles (steps, maximal running step=40), with 6 robots. Each robot was equipped with 16 ultrasonic sensors, meaning 16 sensing sectors/agent. During the simulation, the program used motion strategy *Directional-1*. The coefficients η and λ were chosen in interest of displaying appropriately. The program parameters are summarized in Table 1. Figure 4 contains the starting positions of the agents (see Figure 4a) and all the visited locations of the agents during the 40 cycles (see Figure 4b).

Table 1
The parameters used in process

Specification	**Value**
Number of Agents	6
Number of sensors/agent	16
Maximum running step	40
Maximum movement step (d_m)	8
Coefficient (η)	1/600
Coefficient (λ)	1



a.)
The initial positions of the agents

b.)
The locations visited by the agents

Figure 4.
The initial positions of the agents and the visited locations by the agents

3 The Path Planning Process

The path planning process is initialized with the created 3D PF of the environment. Firstly, this entire 3D potential field has to be projected down to two dimensions. In ideal case, the 2D PF and the original workspace (WS) (see Figure 3b) should be identical, but unfortunately it is a very rare case because of the unavoidable error of the potential field creation. There are several sources causing error, but the most significant of them can be derived from the ultrasonic proximity measurement (see section 2.1.1: $\varepsilon_i = | D_i^j - \hat{D}_i^j |$). The second error factor also depends on the distance measurement. The difference is that in this case the estimated local potential field value (u_j^k) is calculated vs. the true potential field value ($\bar{u}_j$). Here K stands for the total number of visited locations on the WS and k corresponds to the actual step.

$$\varepsilon^k \cong \sqrt{\frac{1}{K}\sum_{j=1}^{K}(u_j^k - \bar{u}_j)^2}\ ; \tag{18}$$

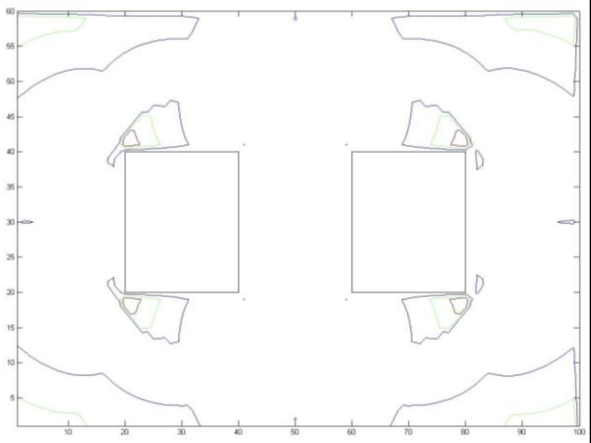
In view of these errors, the *free space* [8] of the WS can be created. (Just for interest: these errors can be seen in Figure 3b, where the local PF value is increasing in direction of axis z. It occurs near to the bounds of the WS and at the obstacles.)

3.1 The Creation of the Area of Navigation

The area of navigation (AN) is a part of the work space, where the mobile robot (agent) can freely move within the given conditions. Usually the given conditions are related to the accuracy of the positioning of the agent, because this gives the basic parameter for the navigation [8]. If the relative/absolute error of the device responsible for the positioning is known, the error of the position can be determined on the entire WS. In the present case, this device is the ultrasonic sensor, with the previously mentioned proximity-error. A more detailed analysis of the error would be exceeding the objectives of the paper, especially because it is neglected. Although, the major part of the errors, in case of similar measuring systems (SONAR), are caused by (1) the false echoes from the corners, (2) the non-perpendicular measuring angles, and (3) the outer noises. Certainly, other types of measuring systems, like the LIDAR, LADAR or RADAR systems have different absolute/relative distance errors. In accordance with this, the created area of navigation of the WS will look like the original WS reduced by the "error zones" at the corners. The created AN with different relative errors can be seen in Figure 5.

An interesting question is whether the area of navigation can or cannot be created on a multi-agent domain? We find that:

- If in the MAS the agents are homogeneous, with the same construction and measuring system, it can be created, at least theoretically,
- If in the MAS the agents are inhomogeneous (with different construction and measuring system) then each type of agents should have their own navigation area.



Fiure 5.
The navigation's area with different relative errors

3.2 The Path Planning

The path planning process starts on the created area of navigation, but it is not sure that the whole area of navigation is accessible for each agent.

Definition: Area of navigation (AN) is accessible for mobile robot (agent) ***R***, if: for ($\forall x, y \in AN$) holds that any "*x*" can be connected with any other "*y*" (where x≠y) in such a way that each point of the connecting line (or curve) is an element of the AN. Moreover, if we are working with a non point-represented robot, the following condition has to be fulfilled: $d(\overline{\mathcal{NA}}^{(i)}, \overline{\mathcal{NA}}^{(i+1)}) > 2r$, where "*r*" is the radius of the circle around the mobile robot and *d* is the distance between two neighbouring *non*-area of navigations ($\overline{\mathcal{NA}}^{(i)}, \overline{\mathcal{NA}}^{(i+1)}$) (usually obstacles).

3.2.1 Path Planning in Single Agent Model

In the case of a single agent, the situation seems to be clear. A single AN is given with accessible territories. The path planning process begins with determining the starting (*S*) and docking (*G*) positions. The only question is how to optimize the path. In this case the optimization process has two phases:

1 Searching for the path with the lowest weight on the weighted and oriented graph-like path. This corresponds to the optimization of the dynamics (smoothness) of the path.

2 Generating a B-spline curve to the selected graph-like path. This corresponds to some kind of time-optimization of the path.

Besides of these, it is possible to generate the so-called "fast" and/or the "safe" path between positions *S* and *G*. The difference between them is that the *safe* path lies on the *centreline* (Voronoi diagram) of the AN, and the *fast* path is generated based on the *visibility graph method* between *C* and *G*.

3.2.1.a Graph-like Path Generation

In the case of a single agent, firstly the Voronoi diagram of the AN is created, and then by the "linearization" of this curve the topology of the AN is obtained [9]. Then the weighting of the graph can be divided into 2 parts:

1. Weighting of the edges: here, the weighting can be multi-parameterized. Namely, the edges are not weighted based on length only ($w_1 S_{(i)} L$), but, mainly in multi-agent domain, can also be weighted based on the traffic density ($w_{(n)} S_{(i)} TrD$). In this case the weighting of an edge is:

$$Ws_{(i)} = w_1 S_{(i)} L + \ldots + w_{(n)} S_{(i)} TrD; \tag{19}$$

2. Weighting of the nodes: with the decomposing of the graph-nodes, each cross-point of the graph can obtain multiple weightings. With this procedure, we can consider the complexity of the planned path, that is, the relative incidence and sharpness of the turns. The weighting of the "*n*" signed sub-segment of node *J*, in knowledge of angle α is (see also Fig. 6.)

$$Wn_{(J)} = w_n |180 - \alpha^0_{(i,j)}|; \tag{20}$$

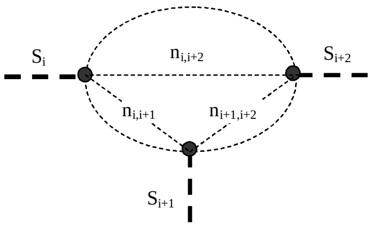


Figure 6.
Weighting of the graph nodes

Generally, the final cost function (*Cf*) is calculated as the sum of the weightings of the edges and nodes:

$$Cf = \sum_i Ws_{(i)} + \sum_J Wn_{(J)}; \tag{21}$$

3.2.1.b Spline Generation

Several methods exist to generate a spline over a polygon (graph-like path). In this particular case, the conditions for the curve generation are the following:

- $S, G \in b(t) \in AN$; - the curve has to contain the starting and docking position and has to be on AN.

- The curve must pass the graph-nodes of the selected route
- The curve must satisfy C^2 continuity
- For each point of the curve, $\forall (x,y \in b(t)) \in AN$ has to be held

Suppose that the *configuration points* of the curve $[p_0,\dots,p_n]$ are given with their respective time values at the graph-nodes of the selected route. The task is to find the function $p_i(t)$ in such a way that it satisfies C^2 continuity, i.e., in the form of a 3-degree polynomial. This can be achieved through the equation system written in (22), from which the derivatives $[\,p'_0,\dots,p'_n]$ can be determined and substituted to define the segments and consequently the composite function *p(t)* (see (23)). For mathematical proof, see [9].

$$\begin{bmatrix} 1 & 0 & .. & & & & \\ T_1 & 2(T_0+T_1) & T_0 & 0 & .. & & \\ 0 & T_2 & 2(T_1+T_2) & T_1 & 0 & .. & \\ . & & & & & & \\ ... & 0 & T_{n-1} & 2(T_{n-1}+T_{n-2}) & T_{n-2} & & \\ ... & & & ... & .0 & 1 & \end{bmatrix} \cdot \begin{bmatrix} p'_0 \\ p'_1 \\ p'_2 \\ . \\ p'_{n-1} \\ p'_n \end{bmatrix} = \tag{22}$$

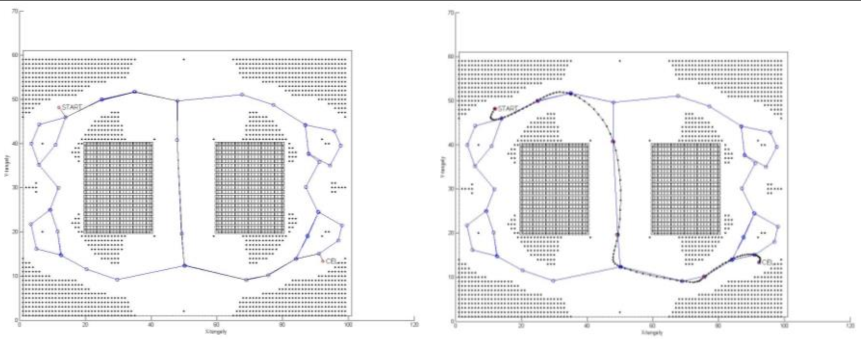
$$\begin{bmatrix} v_{start} \\ 3[T_0/T_1(p_2-p_1)+T_1/T_0(p_1-p_0)] \\ 3[T_1/T_2(p_3-p_2)+T_2/T_1(p_2-p_1)] \\ . \\ 3[T_{n-2}/T_{n-1}(p_n-p_{n-1})+T_{n-1}/T_{n-2}(p_{n-1}-p_{n-2})] \\ v_{end} \end{bmatrix}$$

$$p(t_j) = \sum_{i=-1}^{n+1} p_i N_i^d(t_j) = p_j \tag{23}$$

3.3 Case Study 2 - Path Planning in Single Agent's Domain

In the following, simple simulation results are presented for the path planning in single agent's domain. The first one illustrates the safe path generation, while the other is an example of fast path creation.

3.3.1 The “Safe Path” Generation



a.)
The linearized centreline

b.)
The generated B-spline on the selected polygon

Figure 7.
The graph-like map and the generated curve of the safety path

As it can be seen in Figure 7, in this case the safest path is generated on the centreline of the AN. The complexity of the path is calculated through the Cost function with the following form:

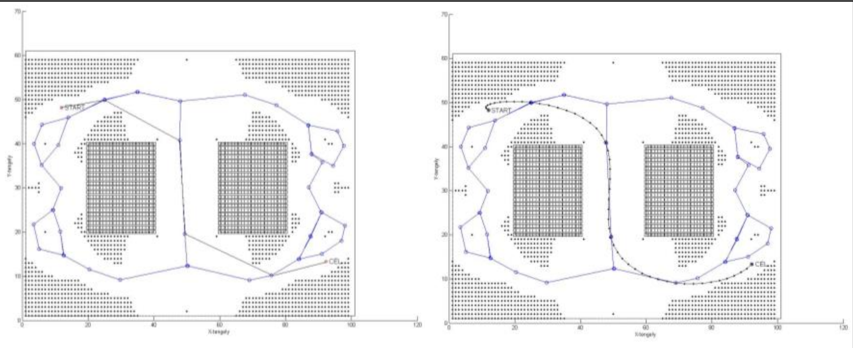
$$Cf_{safety} = \underbrace{\min\left(w_{(g)} SSG_{(g)}\right) + \min\left(w_{(h)} CCG_{(h)}\right)}_{\text{the shortest segments to the "centerline"}} + \min\left(\sum_i Ws_{(i)} + \sum_J Wn_{(J)}\right)\Bigg|_{\substack{SG(g)\to SG(\min)\\ CG(h)\to CG(\min)}} + \min(Cf_{g\times h\times k}) \tag{24}$$

The parameters and results of the calculations are summarized in Table 2.

Table 2
The calculated results of the “safety” path

Parameters (supposed)		Calculated results	
Velocity	3 [m/s]	Nr. of nodes	12
Acceleration	1 [m/s]	Execution’s time	65,322 [s]
Deceleration	1 [m/s]	Length	120,97 [m]
Orientation’s changing	1 [s]	Complexity	835,07

3.3.2 The “Fast Path” Generation



a.)
The linearized centreline

b.)
The generated B-spline on the selected polygon

Figure 8.
The graph-like map and the generated curve of the fast path

The cost function of the fast path can be as:

$$Cf_{fast} = \min(Cf_{g\times h\times k}) + \underbrace{w_{(g)}SSG_{(\min)} + w_{(h)}CCG_{(\min)}}_{\text{Weighting of the segments between the visible nodes}} \tag{25}$$

The parameters and results of the calculations are summarized in Table 3.

Table 3
The calculated results of the “fast” path

Parameters (supposed)		Calculated results	
Velocity	3 [m/s]	Nr. of nodes	6
Acceleration	1 [m/s]	Execution’s time	61,483 [s]
Deceleration	1 [m/s]	Length	110,35 [m]
Orientation’s changing	1 [s]	Complexity	654,79

3.3 Path Planning and Optimization in Multi-Agent Domain

There is a large amount of problems in path planning in a multi-agent domain. The first, as mentioned before, is the problem of homogeneity of the platform. Let us suppose that the area of navigation contains a lot of spaces for opening several multi-lane paths.

In the case of agents with equal dimensions, the lanes can be identically set with the same width to make the path planning easier. The traffic control can follow two strategies: 1) selected agents can wander only on the selected lanes, 2) agents can change the lanes if the selected lane is free of other agents. This strategy needs some centralized control, because the agent has no information about the occupancy of the other lanes. In our opinion, the lanes should be divided into more segments and the occupancy of the segments could be monitored in the so called “segment’s occupancy table” belonging to the central traffic control.

In case of agents with different dimensions and different localization’s systems, the situation is more complicated. The ideal case would be dynamically changing the lane-width in the different segments, what is hardly realistic, but not impossible (theoretically). The second (more realistic) possibility is to fit the lane-width to the biggest agent in the system. In this way, we are back at the previous, static lane-width system.

Outside of this, the path optimization in the multi-agent domain should be solved in the same way as in the single agent system, i.e., by the topological graph of the environment, with the only difference that the topological path will have more sub-branching or, in dynamically changing segmentation, the topological graph will change dynamically as well.

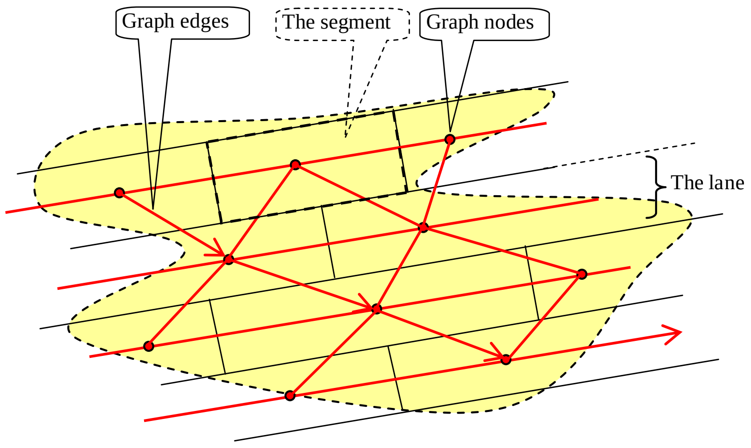


Figure 9.
An extracted part of the multi-lane work space

The regulations regarding the weighting of the graph, which were stated in Sub-subsection 3.2.1.a, are valid in this case too, except that in the multi-agent case we have to think in the space-time continuum due to the occupation of different segments in different time intervals. This time factor increases the computation and the dimensionality of the problem.

Conclusions

In the research field of mobile-robot systems, a large amount of papers have been published related to intelligent navigation or parking systems [10]. Among the most significant approaches in parking navigation is the use of a hybrid navigation structure with elements of computational intelligence (CI) [11]. The first papers dealt with the single agent’s systems. Later on multi agent systems (MAS) have come into the focus, where the agents were defined in a different way. A good example for the extended agent’s definition can be found in [12], where the basic classification of the agents is extended by the functional-computational aspect. The next direction of research in MAS concerns the cooperation and/or the self-organization of the agents. This is possibly one of the most recent fields of research regarding multi-agent systems. In [13] the agent’s cooperation is evaluated based on the agent’s performance. The self-organization can be found not only in multi-agent systems but in manufacturing systems too [14]. After or parallel with agents cooperation, swarm technology, inspired by the behaviour of ants and bees has started to spread [15]. However, in this paper the use of the expression *swarm-technology* is avoided, replaced by such expressions like cooperation, collaboration, or self-organization, which seem more suited to the situation.

This paper introduces a new method for potential field (PF) building on partly centralized multi-agent’s domain. The ‘partly centralized’ expression is because the PF is built-up on a host computer, however the agents share their actual positions with each other (and certainly also with the remote host computer). The presented strategy is analysed, evaluated, and illustrated through a case study.

Furthermore a new path planning process is proposed in this paper (see section 3: The path planning process). Firstly, the theorems of optimal path planning are introduced in single agent’s domain, then analysed and evaluated through another case study. Afterwards, the author shows that the new formulas developed and used in single agent’s system can be validated, with the burden of higher computational demands, in MAS, too.

Acknowledgement

This work was sponsored by the Hungarian National Scientific Fund (OTKA 105846). The project was realised through the assistance of the European Union, with the co-financing of the European Social Fund, TÁMOP-4.2.1.B-11/2/KMR-2011-0001.

References

[1] M. K. Sahota, A. K. Mackworth: Can Situated Robots Play Soccer? (Mar. 04. 1999);
URL: http://www.cs.ubc.ca/spider/mack/links/papers/ai94sm/ai94sm.ps

[2] M. K. Sahota, A. K. Mackworth, R. A. Barman, S. J. Kingdon: Real-Time Control of Soccer-Playing Robots Using Off-Board Vision, in: IEEE International Conference on Systems, Man, and Cybernetics, 1995, pp. 3690-3693

[3] H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, M. Asada: The Robocup Synthetic Agent Challenge 97, in: International Joint Conference on Artificial Intelligence (IJCAI97), 1997

[4] J. A. Iglesias, A. Ledezma, A. Sanchis: Caos Coach 2006 Simulation Team: An opponent Modelling Approach, Computing and Informatics, Vol., pp. 1-23, V 2008-jan-17

[5] A. E. Yilmaz: Swarm Behaviour of the Electromagnetics Community as regards Using Swarm Intelligence in their Research Studies, Acta Polytechnica Hungarica, ISSN 1785-8860, Vol. 7, No. 2, pp. 81-93, 2010

[6] I. Nagy: Behaviour Study of a Multi-agent Mobile Robot System during Potential Field Building, Acta Polytechnica Hungarica, Vol. 6, No. 4, pp. 111-136, 2009

[7] J. Liu, J. Wu: Multi-Agent Robotic Systems, CRC Press LLC, ISBN 0-8493-2288-X, 2001

[8] T. Lozano-Perez: Spatial Planning: a Configuration Space Approach, in IEEE Trans. on Computers 32/1983

[9] Nagy I.: Pozicionáló pontosságon alapuló közel idő-optimális pályatervezési eljárás mobilrobotokhoz (Localization Error Based Near Time-optimal Path Planning Process, for Mobile Robots), PhD Thesis, pp 61-84, Budapest University of Technology and Economics, Hungary, 2010, In Hungarian

[10] Gy. Mester: Intelligent Mobile Robot Motion Control in Unstructured Environments, Acta Polytechnica Hungarica, Journal of Applied Sciences, Vol. 7, Issue No. 4, pp. 153-165, Budapest, Hungary, 2010

[11] 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

[12] J. Kelemen: Agents from Functional-Computational Perspective, Acta Polytechnica Hungarica, ISSN 1785-8860, Vol. 3, No. 4, pp. 37-54, 2006

[13] 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

[14] J. Somló, I. J. Rudas: General Solution for the Self-Organizing, Distributed, Real-Time Scheduling of FMS- Automatic Lot-Streaming Using Hybrid Dynamical Systems, Acta Polytechnica Hungarica, ISSN 1785-8860, Vol. 8, No. 6, pp. 155-180, 2011

[15] J. Dréo, P. Siarry: A New Ant Colony Algorithm Using the Heterarchical Concept Aimed at Optimization of Multiminima Continuous Functions, in Proceedings of the Third International Workshop on Ant Algorithms (ANTS'2002) Vol. 2463 of LNCS, M. Dorigo, G. Di Caro and M. Sampels, ed.s, Berlin: Springer-Verlag, pp. 216-221, 2002