PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 49, NO. 1–2, PP. 43–57 (2005)

PATH PLANNING ALGORITHM, BASED ON USER-DEFINED MAXIMAL LOCALIZATION ERROR

István NAGY

Budapest Polytechnic
Bánki Donát Faculty of Mechanical Engineering
Department of Mechanical and System Engineering
H–1081 Budapest, Népszínház u. 8.
e-mail: nagy.istvan@bgk.bmf.hu

Received: April 11, 2005

Abstract

A model-based path planning algorithm will be presented in this paper. The whole model, like the algorithm, is divided into 2 parts. The 1st part begins with a map-building process over an unknown environment, which is based on the construction of the so-called *Potential Field* (PF) of the environment. In this part, the above mentioned PF will be created by three autonomous robots, equipped with US sensors, which will cooperate and update the *global potential map* on the remote host [12]. The 2nd part begins with the calculation of the environment's 2D mathematical model. The calculation is realized through the *thresholding* of the global potential map. Furthermore, a landmark arrangement will be defined on this model. The *Artificial Error Field* (AEF), which covers the entire workspace, will be calculated and the result will depend on the sensory system of the mobile robot/robots and on the landmark arrangement. Actually, the three-dimensional *AEF* contains the *localization errors* corresponding to each 'x, y' position of the work space's free space. In respect of the *user-defined* maximal localization error ($\varepsilon_{\max}$), some *navigation paths (NP)* can be generated. These paths serve as the base for the calculation of the possible routes in form of directed and *weighted graph-map*. The route with *minimal complexity* between the *start* and the *docking* positions on this graph-map will be selected. Each point of the mobile robot's exact trajectory must fit in the selected navigation path (NP). This maintains the allowed position error below the defined limit. The shape of the trajectory is calculated by the use of cubic B-splines.

Keywords: Artificial Error Field (*AEF*), B-spline, Localization Error, Mobile Robot, Navigation Path (*NP*), Potential Field (*PF*), Workspace (*WS*).

1. Introduction

Models play a very important role in the process of learning from practice. Models of the controlled systems can be used for refining the commands on the basis of error analysis. Better models lead to faster correction of command errors, requiring less practice to achieve a given level of performance. The benefits of accurate modelling are improved performances in all aspects of control. This paper shows only one of the several possibilities to create a path planning algorithm, which starts out, – from the aspect of agents (mobile robots) –, from the totally unknown environment, accomplishing a B-spline curve (which is representing the most accurate trajectory), as a final trajectory.

The inspiration for this modelling was drawn from real life. How does a human being react when dropped to a totally unknown environment? First of all he looks around, in technical terms: *making a bitmap, through the visual sensors, from the environment*. After this, or parallel with this action, some significant points of the environment are designated (or memorized). These are the natural markers of the environment: *displacement of the markers*. From the "bitmap" the possible routes can be specified: wherever the robot can go through without grazing, a route could be defined. The dimensions of the obstacles in the model are given with their physical dimensions, plus some safety zone around them. The *'localization'* is realized with the aid of the 'natural markers'. The only difference *between the above mentioned algorithm and the one mentioned below, in section 4*, is the realization of the '*AEF*'. In the above-mentioned algorithm the localization error is analysed only at critical positions (close to the obstacles), while in section 4.*C*, the localization error is analysed on the whole work space.

In this paper the indoor environment is studied and the probability of the mobile robot occurrence will be calculated in each position, as accurately as possible.

On the other hand, the difference from the other algorithms is that this algorithm is built up on the user-defined maximal localization error ($\varepsilon_{\max}$), that can be generally determined for the entire workspace (*WS*) or merely for the critical segments of the WS. Additional difference can be recognized in the selection of the final optimal trajectory between the *START* and the *GOAL* positions: firstly, in the *weighting* of edges and nodes of the prepared *graph's map*, secondly, in the *final trajectory*, which is generated through the local minima of the calculated AEF.

2. Previous Works

The view of optimization and robot navigation has been studied in some previous works, but in what follows, only the most significant items will be mentioned.

- M. BETKE in [1], has studied the problems of piecemeal learning of an unknown environment. The robot must return to its starting point after each exploration. For the sake of a more precise localization of the mobile robot, according to Betke, it must perform alike.

In the present case the localization is performed in respect of the artificial markers. The motion and the path of the mobile robot are calculated in respect of these markers.

- The '*bug algorithm*' is one of the path planning methods which is closer to my research. The bug algorithm is guaranteed to get the goal location if it is accessible [9]. Note: The length of this trajectory can be arbitrarily worse than the length of the optimal trajectory. According to the bug algorithm and the renovated '*dist-bug algorithm*' the robot always returns to the *SG (Start-Goal)* line after circumnavigating the obstacles.

In this path planning method the *SG* line is not necessary, because reaching the goal and trajectory optimization are ensured by the 'global planning' that is based on the graph-like map of the environment.

- J. SOMLO and J. PODURAJEV in [10] classify the time optimal control problems into three categories: Motion on a constrained path between two endpoints; Motion in a free *WS* between two endpoints; Motion in a free *WS* containing obstacles. They suppose that the geometry of the path is already known, and divide it into two parts: the cruising part and the transient part. On the cruising part the motion is performed with the '*working speed*', and during the transient part this working speed value is reached.

In the present paper, the final trajectory – cubic B-spline – is generated, and the motion speed is determined continuously along the entire path.

- Significant work in the field of dynamical trajectory optimization is Cs. GÜRTLER's diploma work [13], which was further developed in [11].
- Other research works aim at solving the problem of trajectory optimization. As in the final trajectory selection, in optimization, either in a known or unknown environment, the main role was assigned merely to the path-length, the complexity of the paths was not taken into consideration. These works are summarized in *Table 1*.

Table 1. Summarizing some previous works

Model	Properties	Results
GRAPH	Cow-path problem, w paths, origin s, goal t is on one path	Optimal deterministic algorithm, [2].
	If $w = 2$ (chain graph)	Optimal spiral search, Competitive ratio: 9
	Layered graph, width 2	Optimal algorithm with competitive ratio: 9, [3].
	Grid graph, Distance $d(s, t) = n$	$9n - 2$ steps, [2].

The selection of the dynamically optimized path – in the developed model – is built up on the weighted graph, where the edges and the nodes of the graph are weighted differently. More detailed explanation of this problem can be found in [4].

- The works of Kavraky and Bessiere cannot be omitted, dealing with the two phases' path planning algorithms. These algorithms are:

– PPP (Probabilistic Path Planner) method – L. Kavraky,'96.
– ACA (Adriane's Clew Algorithm) – A. Bessiere,'94.

The two phases are the Global (*GPP*) and the Local (*LPP*) path planning algorithms.

FRAICHARD and MERMOND [5], solved the problem of final trajectory detection again in two phases, being the learning and the searching phase [5].

In this research these two phases are in fusion, therefore this algorithm was divided into seven main parts (A, B, C, …), which will be shown in the following through a specific example. However, some definitions must be clarified first (see below, Section 3).

- The path planning methods NF1, NF2 and VFF also have to be mentioned, which are very close to the potential field method. A number of significant problems have been identified which are inherent to potential field path planning methods and independent of the particular implementation (the local minima). One of the most severe problems of this kind is the tendency of the robot to oscillate in narrow corridors [15].

In present paper the potential field is only used for potential field map building. The path planning method is based upon the AEF and the user-defined localization error.

3. Basic Definitions

The description of the navigational environment based on the well-known method of the so-called *configuration obstacles* was at first introduced by LOZANO–PEREZ [14].

Let us assume the following orders and basic relations:

$^{(i)}\mathrm{WS}$ – the i-th workspace in the system.
$v^{(i)}\mathrm{WS}_{(j)}$ – j-th vertex of the i-th WS.
$b^{(i)}\mathrm{WS}_{(j)}$ – j-th edge (boundary) of the i-th WS.

$$b^{(i)}\mathrm{WS}_{(j)} = \left|v^{(i)}\mathrm{WS}_{(j+1)} - v^{(i)}\mathrm{WS}_{(j)}\right| \tag{1}$$

similarly:

the obstacles are marked with B.

$v^{(i)}B^{(m)}_{(n)}$ – n-th vertex of m-th obstacle in i-th WS.

$$b^{(i)}B^{(m)}_{(n)} = \left|v^{(i)}B^{(m)}_{(n)} - v^{(i)}B^{(m)}_{(n+1)}\right| \tag{2}$$

$^{(i)}\mathrm{FS}$ – free space of the i-th WS.
$^{(i)}\mathrm{AFS}$ – reduced (aligned) free space.

For the more detailed FS and AFS explanations see [14].

$$
\begin{aligned}
{}^{(i)}\mathrm{AFS} &= {}^{(i)}\mathrm{WS} - \sum \mathrm{int}\,({}^{(i)}B^{(m)}_{(n)} + R) \\
{}^{(i)}\mathrm{AFS} &\subset {}^{(i)}\mathrm{FS},
\end{aligned}
\tag{3}
$$

where R is radius of the encircled mobile robot.

${}^{(i)}\mathrm{Err}_{(x,y)}$ – position error of i-th WS in (x, y) location (AEF).
${}^{(i)}\varepsilon_{\max}$ – allowed maximal localization error (user defined) of i-th WS.
${}^{(i)}\mathrm{NP}_{(o)}$ – o-th navigation path of i-th WS.

$$
\begin{aligned}
{}^{(i)}\mathrm{NP} &\xleftarrow{x,y} {}^{(i)}\mathrm{Err}_{(x,y)} \cap {}^{(i)}\varepsilon_{\max}; \\
{}^{(i)}\mathrm{NP} &\subset {}^{(i)}\mathrm{AFS}
\end{aligned}
\tag{4}
$$

similarly:

${}^{(i)}\mathrm{RL}_{(p)}$ – p-th reduced navigation path. (piecemeal linearized NP, or rhumb-lines RL.)

$$
{}^{(i)}\mathrm{RL} \subseteq {}^{(i)}\mathrm{NP}; \tag{5}
$$

The more detailed NP and RL generations can be seen below in section 4.D.

$\mathrm{Mb}^{(i)}\mathrm{WS}^{(j)}_{(k)}$ – k-th marker located at the j-th boundary of the i-th WS.
$\mathrm{Mv}^{(i)}\mathrm{WS}^{(k)}$ – k-th marker located at the k-th vertex of the i-th WS.
$\mathrm{M}^{(i)}\mathrm{b}^{(m)}B^{(j)}_{(k)}$ – k-th marker located at the j-th boundary (edge) of the m-th obstacle in the i-th WS.
$\mathrm{Mv}^{(i)}B^{(m)}_{(k)}$ – k-th marker located at the k-th vertex of m-th obstacle in i-th WS.

For this reason, for the marker located at the middle of the j-th edge of the m-th obstacle is valid:

$$
\mathrm{M}^{(i)}b^{(m)}B^{(j)}_{(\mathrm{middle})} = \left| \frac{v^{(i)}B^{(m)}_{(k+1)} - v^{(i)}B^{(m)}_{(k)}}{2} \right|; \tag{6}
$$

Similarly, for the marker located at the middle of the *j-th* boundary of the WS:

$$
\mathrm{M}b^{(i)}\mathrm{WS}^{j}_{\mathrm{middle}} = \left| \frac{v^{(i)}\mathrm{WS}_{(j+1)} - v^{(i)}\mathrm{WS}_{(j)}}{2} \right| \tag{7}
$$

${}^{(i)}\mathrm{MR}^{(p)}$ – p-th mobile robot in i-th WS, ('p' concerns the multi-agent systems).
${}^{(i)}\mathrm{V}^{(p)}\mathrm{MR}^{(kM)}_{(rx,ry)}$ – visibility (V) between the p-th mobile robot (MR) and k-th marker (M) in the i-th WS.

These equations are very important in determining the visibility of the markers by the mobile robot(s).

Visibility (is a Boolean operator) to the k-th marker ($M_{(mx,my)}$), from the *(rx,ry)* location of the p-th mobile robot in i-th WS.

$$ {}^{(i)}\mathrm{V}^{(p)}\mathrm{MR}^{(kM)}_{(x,y)} = \begin{cases} \text{TRUE}, & \text{if} : (T) \\ \text{FALSE}, & \text{if} : (F) \end{cases} \tag{8} $$

$$ \begin{cases} (T) : \left| {}^{(i)}\mathrm{M}^{(k)}_{(mx,my)} - {}^{(i)}\mathrm{MR}^{(p)}_{(rx,ry)} \right| \cap \left\{ \sum_{m,n} (b^{(i)} B^{(m)}_{(n)}) \cup (v^{(i)} B^{(m)}_{(n)}) \right\} = 0; \\ (F) : \left| {}^{(i)}\mathrm{M}^{(k)}_{(mx,my)} - {}^{(i)}\mathrm{MR}^{(p)}_{(rx,ry)} \right| \cap \left\{ \sum_{m,n} (b^{(i)} B^{(m)}_{(n)}) \cup (v^{(i)} B^{(m)}_{(n)}) \right\} \neq 0, \end{cases} $$

where :

${}^{(i)}\mathrm{M}^{(k)}$ – is the k-th marker of the i-th WS.

$\max{}^{(i)}\mathrm{M}$ – is the maximal number of markers on the i-th workspace, what is the sum of markers, located on the vertexes and boundaries of the obstacles, and the workspace.

$$ k = \langle 1, \max {}^{(i)}\mathrm{M} \rangle; $$

$$ \max {}^{(i)}\mathrm{M} = \sum_{i,j,k} \mathrm{M} b^{(i)} \mathrm{WS}^{(j)}_{(k)} + \sum_{i,k} \mathrm{M} v {}^{(i)}\mathrm{WS}_{(k)} + \sum_{i,j,k,m} \mathrm{M}^{(i)} b^{(m)} B^{(j)}_{(k)} + \sum_{i,k,m} \mathrm{M} v {}^{(i)} B^{(m)}_{(k)}; \tag{9} $$

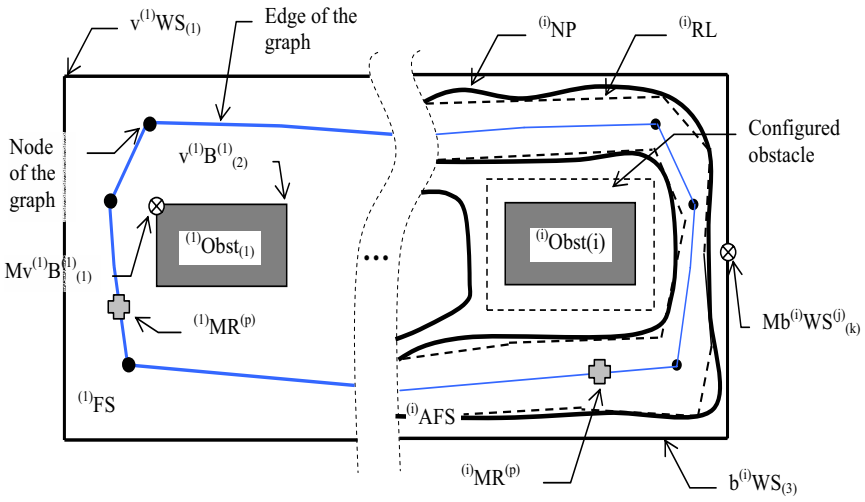


Fig. 1. Basic definitions on 1st and ith WS

3.1. New Functions

- *Localization error*: Error function, calculated in each x, y position of the entire WS (except obstacles).
- *AEF* (Artificial Error Field): is a 3D error field where the magnitude of the localization errors is represented on the 'z' axes.
- *NP* (Navigation's Paths): each x, y localization of the WS, where the *Eq.* (4) is valid.
- *RL* (Rhumb Lines): piecemeal linearized navigation paths.

4. The Algorithm

The algorithm will be shown through a real example realized in a *MATLAB 5.1* environment. In this present model the potential field (*PF*) building was prepared by three agents, but the *AEF* only with a single one. The multi agent *AEF* building is under development, but the possibility of using this algorithm in multi agent systems (*MAS*) will be shown, too.

A. Exploring and map building – in this part of the algorithm the sensory system of the robot (or agent – in *MAS*) plays a significant role. The present sensory model contains 8 ultrasonic sensors placed around, and one 'laser eye' on the top of the robot (See *Fig. 2*).
Each ultrasonic sensor can detect other agents (which are distinguished from the obstacles), and the obstacles, which are measured in the sector enclosed by angle β.

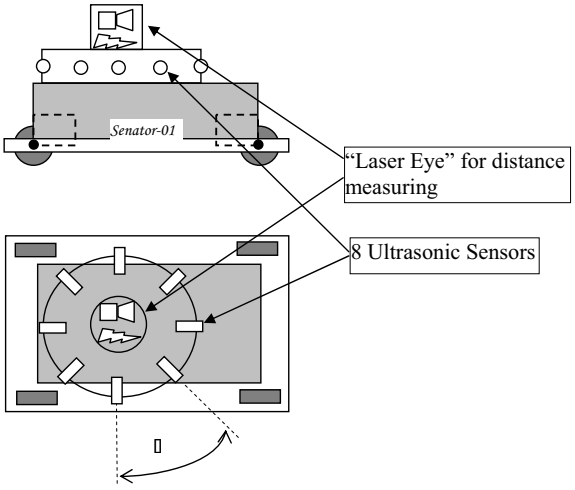


Fig. 2. Sensory system of the mobile robot

$$Ns = \frac{2\pi}{\beta}; \tag{10}$$

where Ns is the number of the equally spaced sensors around the agent. The agents are communicating with each other, and transmitting the data of positions and the direction of the next motion [6], [12].
The 'laser-eye' sensor is used for distance measuring, to promote the localization of the mobile robot from the given displaced markers, see [7]. The exact positions of the markers are already known.
The result of this part of the algorithm is the PF in form of a *.bmp* file. The PF can be expressed as follows [6]:

$$\begin{aligned} U_{\mathrm{ART}}(\vec{x}) &= U_{\mathrm{GOAL}}(\vec{x}) + U_{\mathrm{OBS}}(\vec{x}); \\ U_{\mathrm{GOAL}}(\vec{x}) &= -\frac{1}{2} kp(\vec{x} - \vec{x}_{\mathrm{GOAL}})^2; \\ U_{\mathrm{OBS}}(\vec{x}) &= \begin{cases} \frac{1}{2}\eta \left(\frac{1}{\vec{x}} - \frac{1}{l_0}\right)^2; & \text{if } \vec{x} \le l_0; \\ 0 & \text{if } \vec{x} > l_0; \end{cases} \\ \vec{F}_{\mathrm{ART}} &= -\nabla[U_{\mathrm{ART}}(\vec{x})], \end{aligned} \tag{11}$$

where: kp – is a positive gain, η is a constant and l_0 is a distance threshold, beyond which no repulsive force will be received by the robot, and x is the state vector, describing the position and the orientation of the robot. The resulting U_{ART} is constructed from components associated with the goal U_{GOAL}, and from obstacles U_{OBS}. The potential field (PF) is represented with the magnitude of the minus gradient of the U_{ART}. The algorithm of sensing and self-organizing would exceed the dimensions of this paper, for more detailed explanation see [6], [12]. The resulting *.bmp* file is shown in *Fig. 3*.

B. *Model building* – from the *.bmp* file, the edges of the obstacle and the boundaries of the WS are detected. By keeping the threshold limit on 'z' axis, the 2D mathematical model of the environment is obtained from the 3D *PF map*, see *Fig. 4*. The obstacles are completed to a polygonal form. Further, the whole WS is represented in a matrix, where the free spaces are marked with '1', the obstacles with '2', and the agents in *MAS* with '3'. The agents are point represented. The physical dimensions of the robot (see the above mentioned '*R*') will take effect in planning all the possible routes on the whole environment.

   B.1. The '*first attempt*' of the marker's displacement is based on the model of the environment, namely the markers are placed at the vertexes/vertices of the obstacles and/or at the vertexes/vertices of the *WS*. In *Fig. 4* the mathematical model of the *WS* and the basic displacement (1st attempt) of the markers are presented. The markers are located in the middle of the boundaries (vertices) of the *WS*, and at the vertexes of the obstacles.

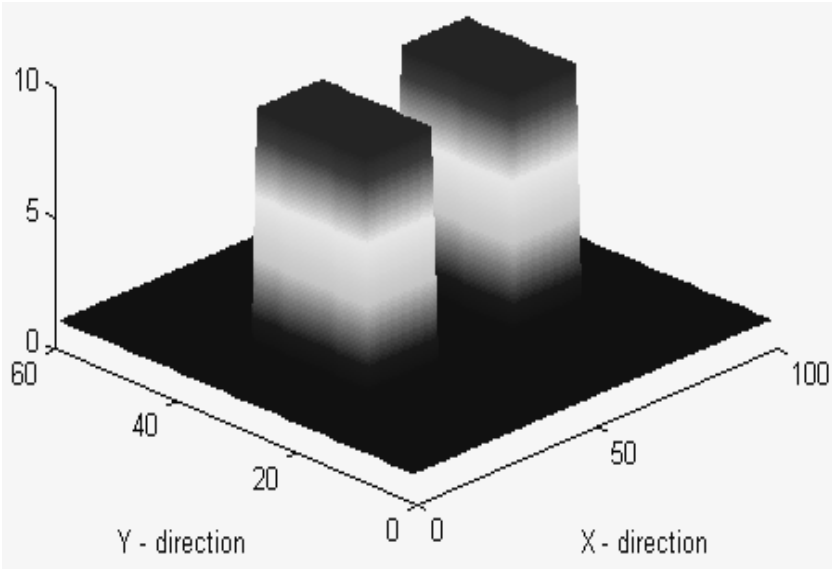


Fig. 3. Potential Field (PF) map of the Entire Workspace

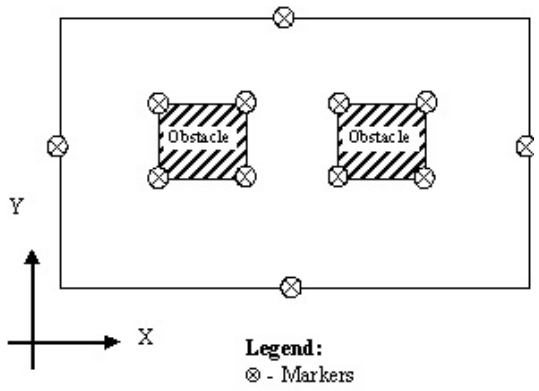


Fig. 4. The Mathematical Model of the PF map and the 1st Attempt of the Marker's Displacement

C. *AEF building* – the model of the sensory system of the robot is given with its relative and/or absolute errors. In this present case, the *AEF* is built up through the model of the 'laser eye' sensory system [7], where the point-represented mobile robot checks its distance from all visible markers at each x, y position of the *WS* (except the obstacles) and calculates the *localization error* function. The mathematical interpretation of the marker's visibility

is presented in (8). So, we get a 3D *AEF*, where the 'z' axes represent the magnitude of the error in positions x, y. The localization error function is calculated from the extent of the *error area*. The error area is calculated from the intersection of the segments, where the segments are given with the relative and/or absolute errors of the sensory system. The intersection of the segments is reduced to a parallelogram, and the size of this fault area is given with the following equation:

$$\begin{aligned} a &= \frac{m_1}{\sin\alpha_1}; \\ b &= \frac{m_2}{\sin\alpha_1}; \\ A &= b \cdot m_1 = \frac{m_1 \cdot m_2}{\sin\alpha_1}, \end{aligned} \tag{12}$$

where m_1 and m_2 are the heights belonging to the a and b sides of the parallelogram, and $m_1 = 2 \cdot \rho_1$ is valid, simultaneously, $m_2 = 2 \cdot \rho_2$.
Further, d_1 and d_2 are the distances measured from M_1 and M_2 markers with given relative/absolute[1] errors ($\rho_{(i)}$). The estimated robot position is $R_{(x,y)}$. For more exact explanation of this problem, and the conditions of reduction see [8] and *Fig. 5*.

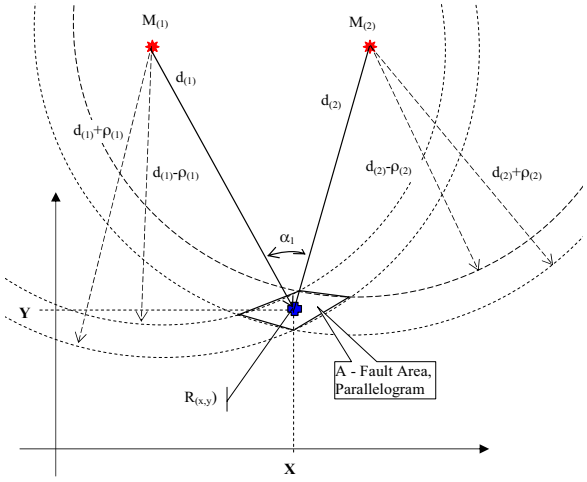


Fig. 5. Modelling of the Fault Area

The three-dimensional *AEF* of the *WS* (the mathematical model of which is given in *Fig. 4*) can be seen in *Fig. 6*.

[1]Relative/absolute: Our 'Laser eye' sensory system has 1 [mm] absolute error in 15 [m], and accordingly the relative one. The model of the 'Laser eye' sensory system was built on this fact.

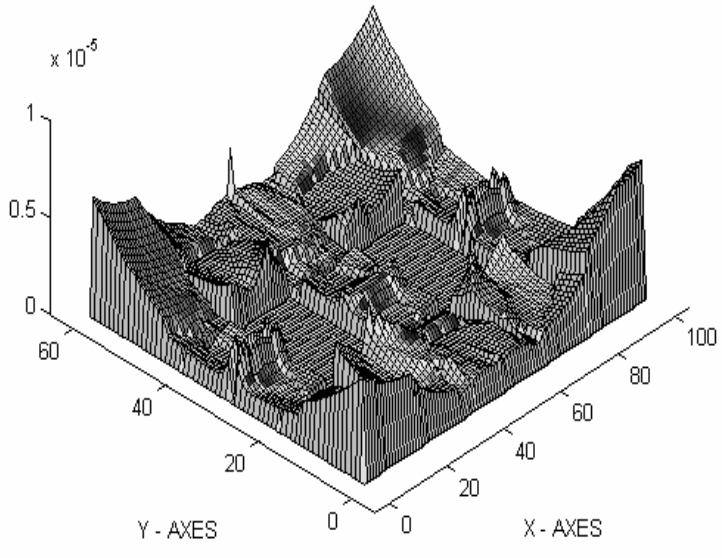


Fig. 6. The 3D *AEF* of the mathematical model given in *Fig. 4*

D. *Creating the navigation paths* (NP) – The intersections of the *AEF* and the user-defined maximal position error (ε_{max}) are projected onto the x, y plane of the *WS*. By connecting the appropriate projected points we get an area (this area is called navigation path – NP) in the *WS* where the following equation is valid:

$$\forall x, y \in NP : \mathrm{Err}_{(x,y)} < \varepsilon_{\max}, \tag{13}$$

where $\mathrm{Err}_{(x,y)}$ is the measured localization error in position x, y.
The reduced navigation path (RL) is given by the piecemeal linearization of the *NP*, see *Fig. 7*.

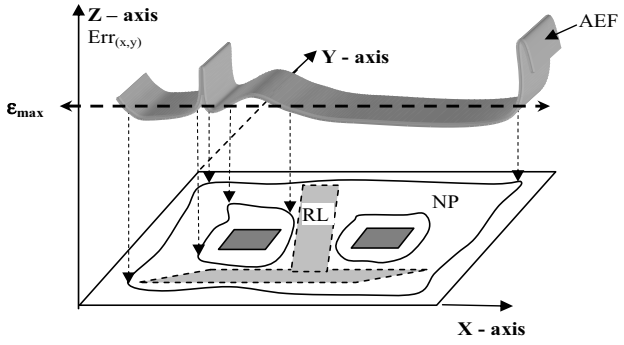


Fig. 7. Projection onto x, y Plane and the Rhumb lines (*RL*)

For the mathematical explanation of the *NP* and the *RL* see *Eq.* (4). Here, the physical dimensions of the robot (R) have to be taken into account. If the width of the *RL* is less than $2R$,the algorithm goes back to the "$B.1.''$ point and begins the '*second attempt*' of the markers' displacement.

E. *The navigation graph's map* – every graph consists of edges and nodes. In our case the edges are the centerlines of the *rhumb* lines and the nodes are the cross points of these centerlines. There are two types of nodes: *cross points* (see *Fig. 8a*), and *segment's end points* (see *Fig. 8b*). The edges and nodes are weighted differently. A very simplified example is when the edges are weighted according to the length and the nodes are weighted according to the angles enclosed between two centerlines. This weighting takes the dynamical features of the robot into consideration. For more detailed weighting, see [4] and *Fig. 8a, 8b*. In the *MAS* the edges and the nodes are weighted in the view of traffic density too.

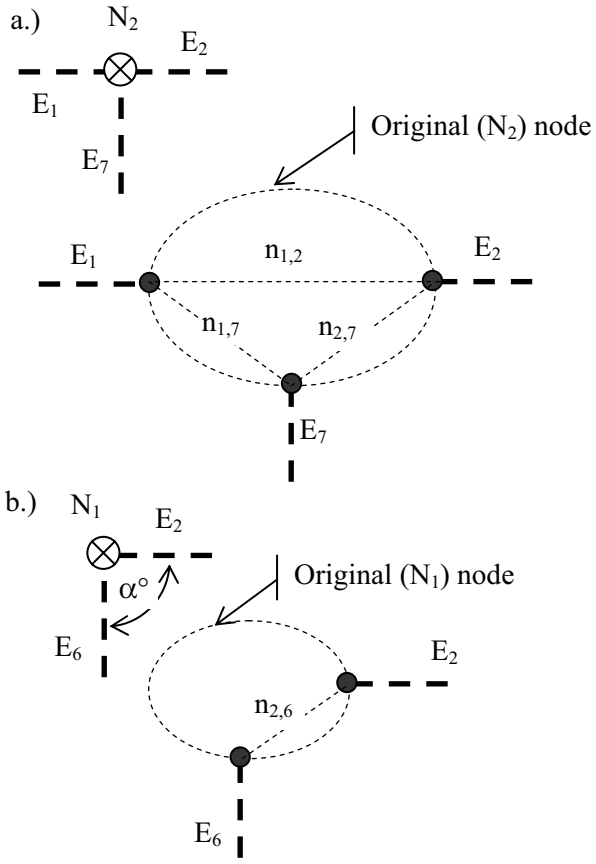


Fig. 8. Weighting of the Edges and the Nodes

F. *Final route selection* – For of route selection, firstly the goal position has to be designated. After this, the final route is selected according to the weighting of the edges and the nodes. Depending on the weighting the minimal (or maximal) weighted route will be selected.

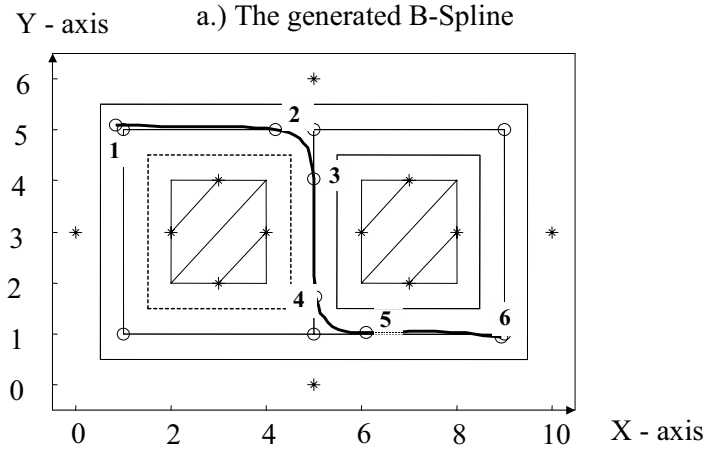
G. *Generating the final smooth trajectory* – The final trajectory is a B-spline curve, generated in the *RL* or *NP* with the following process:

- The local minima of the *AEF* over the *NP* are determined. These points can be the practicable points of the B-spline where the curve is passing through. If the points are relatively close to each other, some of them can be neglected. On the other hand, if they are relatively rare, we can add some extra points. The addition and exclusion are controlled by further rules, e.g. if a point has to be added, it is usually placed at the centerline of the *RL*. On the other hand, if some points have to be neglected (in the case if the local minima are dense), the points which are farther from the centerlines, will be excluded. Further, the knot points of the curve are calculated from these '*passing through*' points, based on the formulas in [4]. If the points of the generated B-spline are overflowing the boundary of *NP*, the rules of addition/exclusion of the 'passing through' points can be changed. If this procedure does not seem to be efficient, the algorithm has to return to the point B.1 to generate the 2nd, 3rd, …etc. attempts of the marker's displacement. Finally, we can generate the localization errors in each point of the final trajectory. See *Fig. 9*, where the points (1..6) in *Fig. 9a* are the '*passing through*' points of the curve. Among these, point '1' is the *START* and point '6' is the *GOAL* position of the mobile robot.

5. Conclusion

Most of the existing path planning algorithms in robotics have not considered the dynamical properties of the platform. The use of the weighted graph's map (mainly the weighted nodes) and the use of smooth trajectories, instead of polygonal ones, is a promising approach to trajectory optimization.

A complete path planning algorithm was shown in this paper. The algorithm starts with the map building of the completely unknown environment, and finishes with the dynamically smoothed final trajectory. The complexity of the algorithm is apparently high, but some parts of the algorithm could be developed separately and, at the end, we can assemble and harmonize these parts. In my specific case the algorithm has been divided into 2 parts. The 1st was a global PF map building process, based on the multi-agent platform, and the 2nd was the AEF calculation and the path planning. The whole second part is purely mathematical, and was built up on the mathematical model of the environment and marker's displacement (*Fig. 4*). Possible inaccuracy can arise at the beginning of the 2nd part, with determining the threshold limit. To eliminate this error, we have to establish a coefficient, based



b.) The Localization Errors in Each Points of the B-spline (Final Trajectory)

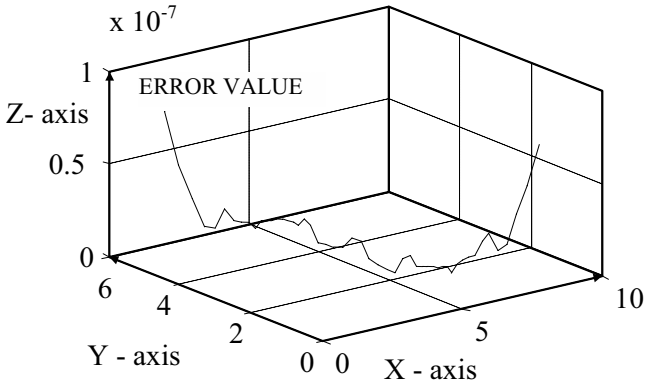


Fig. 9. Spline generation and the localization error of the final trajectory

on the threshold limit, and the mathematical model of the environment has to be calculated through this.

Another important point is the convergence of the algorithm. In the case of a point-represented robot the convergence is ensured, because by the re-arrangement of the markers (2^{nd}, 3^{rd} attempt) and/or increasing the number of markers, the

given limit of the user-defined maximal position error can be definitely kept. In the case of a robot with real dimensions, the possible routes are given with the distances between the obstacle ↔ obstacle, and/or the obstacle ↔ boundaries of the work space.

References

[1] BETKE, M., *Learning and Vision Algorithms for Robot Navigation*, PhD-thesis, M.I.T. Dept. of EE&CS., USA, 1992.

[2] BAEZA-YATES, R. A. – CULBERSON, J. C. – RAWLINS, G. J. E., Searching in the Plane, *Information and Computation*, **106** (2) (1993), pp. 234–252.

[3] PAPADIMITRIOU, CH. H. – YANAKAKIS, M., Shortest Paths without a Map, *Theoretical Computer Science*, **84** (1991), pp. 127–150.

[4] NAGY, I. – VAJTA, L., Local Trajectory Optimization Based on Dynamical Properties of Mobile Platform, *Proc., IEEE Int. Conf. INES'01*, Helsinki, Finland, 2001.

[5] FRAICHARD, TH. – MERMOND, R., Path Planning with Kinematics and Uncertainty Constraints, *Intelligent Autonomous Systems*, ISBN 5-86911-220-6, Ufa, 1998.

[6] LIU, J. – WU, J., *Multi-Agent Robotic Systems*, CRC-press, ISBN 084932288, USA, 2001, pp. 186–189.

[7] NAGY, I., Marker-Based Mobile Robot Positioning in Respect of Displacement of the Markers, *Proc., IEEE Int. Conf., INES'02*, Opatija, Croatia, 2002.

[8] NAGY, I. – VAJTA, L., Trajectory Planning Based on Position Error Analysis and Fault Area Modelling, *Proc. Int. Conf. ICAR'01*, Budapest, Hungary, 2001.

[9] LUMELSKY, V. – STEPANOV, A., Path-Planning Strategies for a Point Mobile Automation Moving amidst Unknown Obstacles of Arbitrary Shape, *Algorithmica*, **2** (4) (1987), pp. 403–440.

[10] SOMLO, J. – PODURAJEV, J., Optimal Cruising Trajectory Planning for Robots, *Proc. of the IASTED Int. Conf. Robotics and Manufacturing*, Oxford, England, 1993.

[11] GÜRTLER, CS. – NAGY, I. – VAJTA, L., Trajectory Planning for Mobile Robots Based on Dynamical Models, *Proc. IEEE. Int. Conf. INES 97*, Budapest, Hungary 1997, pp. 171–174.

[12] NAGY, I., Genetic Algorithms Applied for Potential Field Building in Multi-Agent Robotic System, *Proc., Int. Conf. on Comp. Cybernetics*, ICCC '03, Siófok, Hungary, 2003.

[13] GÜRTLER, CS., *Examination of Autonomous, Sensor-Controlled Mobile Robot's Navigational System*, BME, Diploma work in Hungarian, Budapest, 1996.

[14] BRADY, M. – HOLLERBACH, J. M. – JOHNSON, T. L. – LOZANO-PEREZ, T. – MASON, M. T., *Robot Motion: Planning and Control*, MIT Press, Cambridge, M.A., London, 1982.

[15] BORENSTEIN, J. – KOREN, Y., *The Virtual Force Field (VFF) and the Vector Field Histogram (VFH) Methods – Fast Obstacle Avoidance for Mobile Robots*, DOE project, Univ. of Michigan, USA, 1993.