A Global Path Planning Algorithm, Based on Maximal Localization Error

István Nagy, Attila L. Bencsik

Budapest Polytechnic
Bánki Donát Faculty of Mechanical Engineering
Department of Mechanical and System Engineering
H-1081 Budapest, Népszínház u. 8.
nagy.istvan@bgk.bmf.hu; bencsik.attila@bgk.bmf.hu

Abstract: A model based path planning algorithm will be presented in this paper. The whole model, as the algorithm is divided for 2 parts. The 1st part begins with a map-building process over an unknown environment, which process is based on the construction of the so called Potential Field (PF) of the environment. In this part, the above mentioned PF will be create by three autonomous robots, equipped by US sensors, which will cooperate and update the global potential map on the remote host [12] . The 2nd part is beginning with the calculation of the environment's 2D mathematical model. The calculation is realized through the tresholding of the global potential map. A landmark arrangement will be defined on this model. In further, 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; on the landmark arrangement. Actually the three-dimensional AEF contains the localization errors corresponding to each "x, y" position of the mobile robot. In respect of the user defined maximal localization error (ε_{max}), some navigation paths (NP) can be generated. These paths are the bases to the calculation of a map for the possible routes in form of directed and weighted graph. We select the route with minimal complexity between the start and docking positions on this graph. Each point of the mobile robot's exact trajectory must fit within the selected navigation path. 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

The models play very important role in learning from practice. Models of the controlled systems can be used to refine the commands on the basis of analyzing the errors. Better models lead to faster correction of command

errors, requiring less practice to achieve a given level of performance. The benefits of accurate modeling are improved performances in all aspects of control. This paper will show only one of the several possibilities to create a path planning algorithm, which is started out, -in aspect of agents (mobile robots)- from the totally unknown environment and accomplishing with a B-spline curve (which is representing the most accurate trajectory), like a final trajectory.

The inspiration of this modeling was drawn on the real life. How does a man react, if it is dropped to a totally unknown environment? First off all it would look around, in technical explanation: *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 analyzed only at critical positions (close to the obstacles), while in section *4.C*, the localization error is analyzed on the whole environment.

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 the algorithm is built up on the user defined maximal localization error **(ε_{max}),** what 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 the edges and nodes of the prepared graph's map, secondly, in the *final trajectory*, which is generated through the local's minima of the calculated AEF.

2. Previous works

I have studied some previous works in view of optimization and robot navigation. I will mention only the most significant works.

- 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 the more precise localization of the mobile robot, according to Betke, it must perform as like.

In my case the localization is performed in respect of the natural/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 my path planning method the *SG* line is not necessary, because the goal reaching and trajectory optimization is insured by the "global planning" what is based on the graph-like map of the environment.

- J. Somlo and J. Podurajev in [10] the time optimal control problems are classified into three categories: Motion on constrained path between two endpoints; Motion in free *WS* between two endpoints; Motion in a free *WS* containing obstacles. They suppose that the geometry of the path is already known, and divided 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 this present paper the final trajectory –cubic B spline- is generated, and the motion speed is determined continuously along the entire path.

- Significant work made in field of dynamical trajectory optimization is Cs. Gürtler's diploma work [13], which was further developed in [11].

- Other researches are aimed to solve the problem of the trajectory optimization. As yet in the final trajectory selection, in optimization, either in known or unknown environment, the main role was pointed merely to the path-length. The complexity of the paths was not taken into consideration. These works are summarized in TAB.1.

Model	Properties	Results
G R A P H	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].

TAB.1. Summarizing some previous works

The selection of the dynamically optimized path of my 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].

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 we assume the following orders and basic relations:

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

$$b^{(i)}WS_{(j)} = \mid v^{(i)}WS_{(j+1)} - v^{(i)}WS_{(j)} \mid \qquad (1);$$

similarly:

The obstacles are signed with *B*.

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

$$b^{(i)}B^{(m)}_{(n)} = \mid v^{(i)}B^{(m)}_{(n)} - v^{(i)}B^{(m)}_{(n+1)} \mid \qquad (2);$$

$^{(i)}$FS – free space of the *i-th* WS.

$^{(i)}$AFS – reduced (aligned) free space.

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

$$^{(i)}AFS = {}^{(i)}WS - \sum \mathrm{int}(^{(i)}B^{(m)}_{(n)} + R) \qquad (3);$$
$$^{(i)}AFS \subset {}^{(i)}FS$$

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's path of *i-th* WS.

$$^{(i)}NP \xleftarrow{x,y} {}^{(i)}Err_{(x,y)} \cap {}^{(i)}\varepsilon_{\max} \qquad (4);$$
$$^{(i)}NP \subset {}^{(i)}AFS$$

similarly:

$^{(i)}RL_{(p)}$ – *p-th* reduced navigation's path. (piecemeal linearized *NP*, or rhumb-lines *RL*.)

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

The more detailed *NP* and *RL* generation can be seen bellow in section 4.D.

$Mb^{(i)}WS^{(j)}_{(k)}$ – *k-th* marker located at the *j-th* boundary of the *i-th* WS.

$Mv^{(i)}WS^{(k)}$ – *k-th* marker located at the *k-th* vertex of the *i-th* WS.

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

$Mv^{(i)}B^{(m)}_{(k)}$ – *k-th* marker located at the *k-th* vertex of *m-th* obstacle in *i-th* WS.

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

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

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

$$Mb^{(i)}WS^{j}_{middle} = \left| \frac{v^{(i)}WS_{(j+1)} - v^{(i)}WS_{(j)}}{2} \right| \qquad (7);$$

$^{(i)}MR^{(p)}$ – *p-th* mobile robot in *i-th* WS, ("*p*" is concerned to the multi agent-systems).

$^{(i)}V^{(p)}MR^{(kM)}_{(rx,ry)}$ – 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 determination of 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*.

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

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

where :

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

$\max^{(i)}M$ – 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, respectively workspace.

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

$$ \max {}^{(i)}M = \sum_{i,j,k} Mb^{(i)}WS^{(j)}_{(k)} + \sum_{i,k} Mv^{(i)}WS_{(k)} + $$

$$ + \sum_{i,j,k,m} M^{(i)}b^{(m)}B^{(j)}_{(k)} + \sum_{i,k,m} Mv^{(i)}B^{(m)}_{(k)} \qquad (9); $$

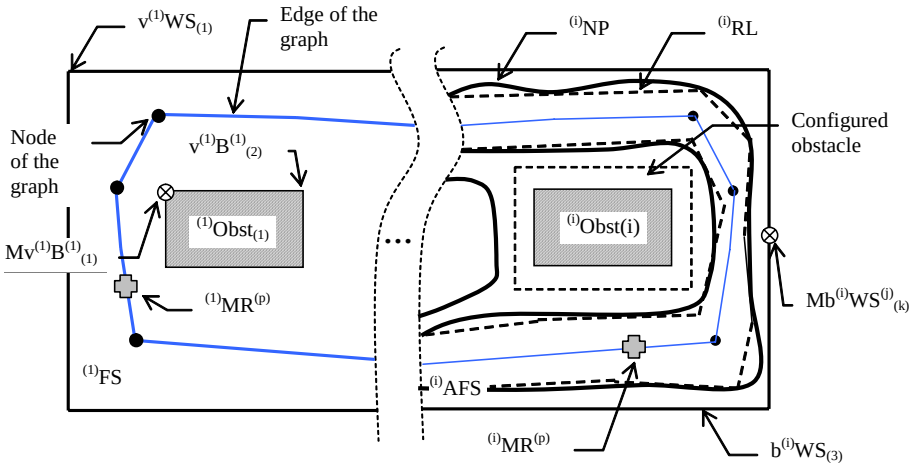


Fig.1.
Basic Definitions on 1st and ith WS

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 are represented on the "*z*" axes.
- *NP* (Navigation's Paths): each *x, y* localization of the WS, where the equation (4) is valid.

4. The algorithm

The algorithm will be shown through a real example, which was realized in *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 developing, but I will show the possibility of utilizing this algorithm in multi agent systems (*MAS*) 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. On my sensory model, 8 ultrasonic sensors are placed around, and one "laser eye" on the top of the robot (See Fig. 2.).

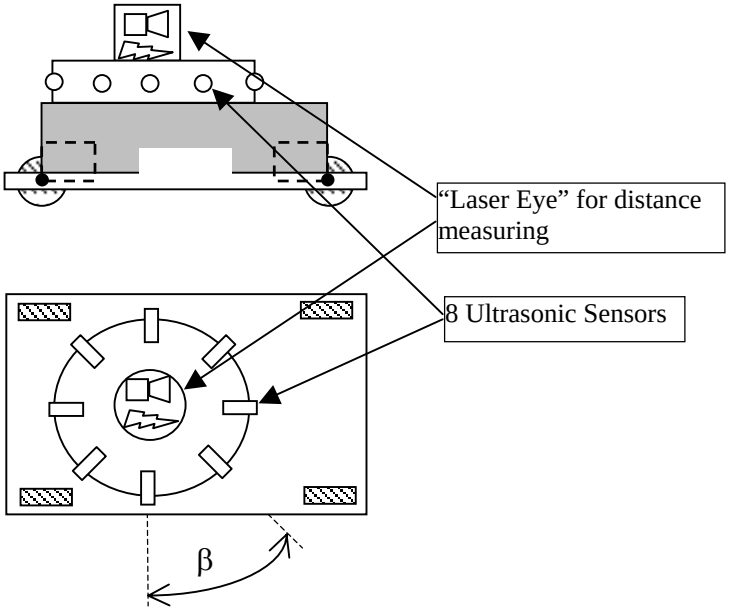


Fig. 2.
Sensory System of the Mobile Robot

Each ultrasonic sensor can detect other agents, (which are distinguished from the obstacles) and the obstacles, measured in the sector enclosed by angleβ.

$$Ns = \frac{2\pi}{\beta} \qquad (10);$$

where, *Ns* –is the number of the equally spaced sensors around the agent. The agents are communicating between each other, and transmitting the dates 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 *.bmp* file. The PF can be expressed as follows [6]:

$$U_{ART}(x) = U_{GOAL}(x) + U_{OBS}(x);$$

$$U_{GOAL}(x) = -\frac{1}{2} kp(x - x_{GOAL})^2;$$

$$U_{OBS}(x) = \begin{cases} \frac{1}{2}\eta\,(\frac{1}{x} - \frac{1}{l_0})^2; \; if \quad x \le l_0; \\ 0 \qquad if \quad x > l_0; \end{cases} \qquad (11);$$

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

where: *kp* – is a positive gain, η is a constant and l_o is a distance threshold, beyond which no repulsive force will be received by 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 be exceeding the dimensions of this paper, for more detailed explanation see [6], [12]. The resulting *.bmp* file is shown in Figure 3.

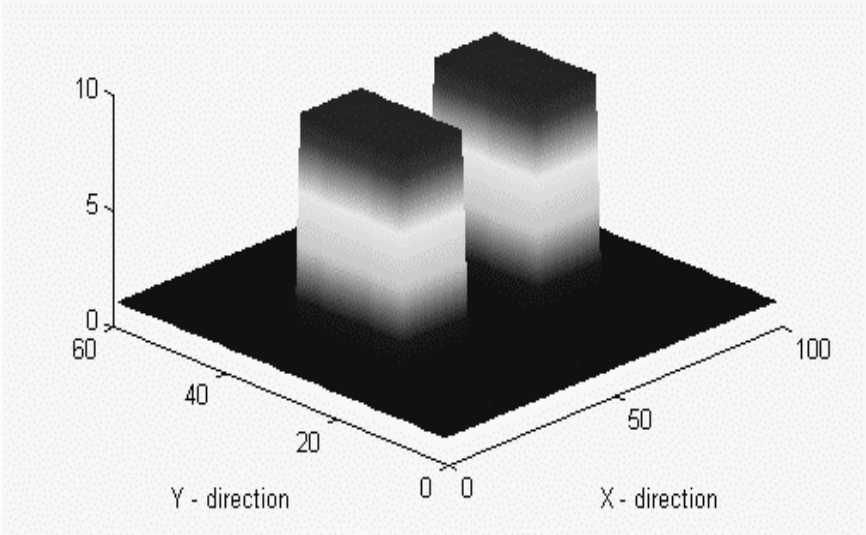


Fig. 3.
Potential Field (PF) map
of the Entire Workspace

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" axes, the 2D mathematical model of the environment is obtained from the 3D *PF*, 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 indexed 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 of 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 at the middle of the boundaries (vertices) of the *WS*, and at the vertexes of the obstacles.

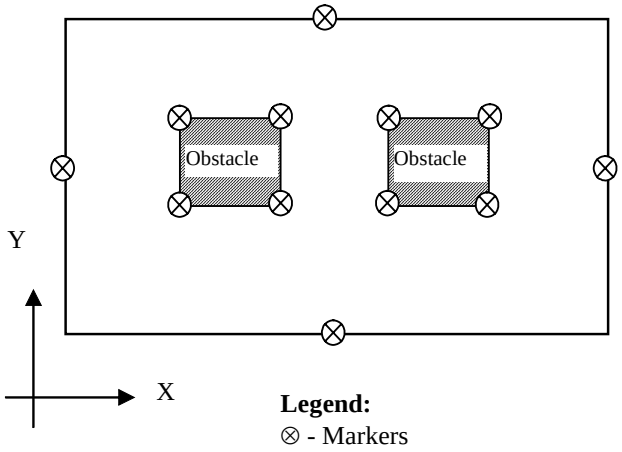


Fig. 4.
The Mathematical Model of the PF map and
the 1st Attempt of the Marker's Displacement

C. *AEF building* – the model of the sensory system of the robot is given with its relative and/or absolute[i] errors. In this present case, the *AEF* is built up through the model of the "laser eye" sensory system [7], when the point represented mobile robot checks its distance from all visible markers at each *x, y* positions (except the obstacles) of the *WS* 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 representing 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:

$$a = \frac{m_1}{\sin \alpha_1};$$

$$b = \frac{m_2}{\sin \alpha_1}; \qquad (12);$$

$$A = b.m_1 = \frac{m_1.m_2}{\sin \alpha_1};$$

where, m_1 and m_2 are the highs belonging to the *a* and *b* sides of the parallelogram.

Further, d_1 and d_2 are the distances measured from M_1 and M_2 markers with given relative/absolute[i] errors. For more exact explanation of this problem, and the conditions of reduction see [8] and Fig. 5. The three dimensional *AEF* of the *WS* (the mathematical model of the WS is given on Fig.4) can be seen on Fig. 6.

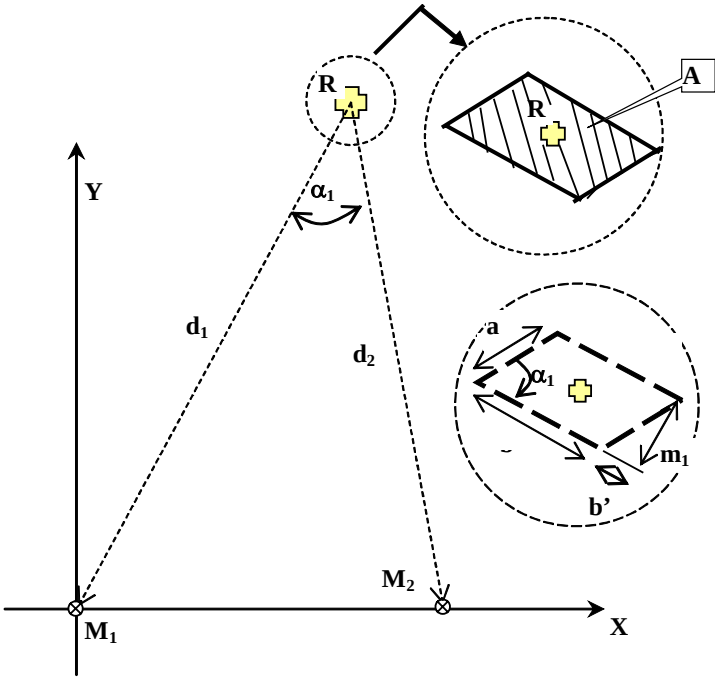
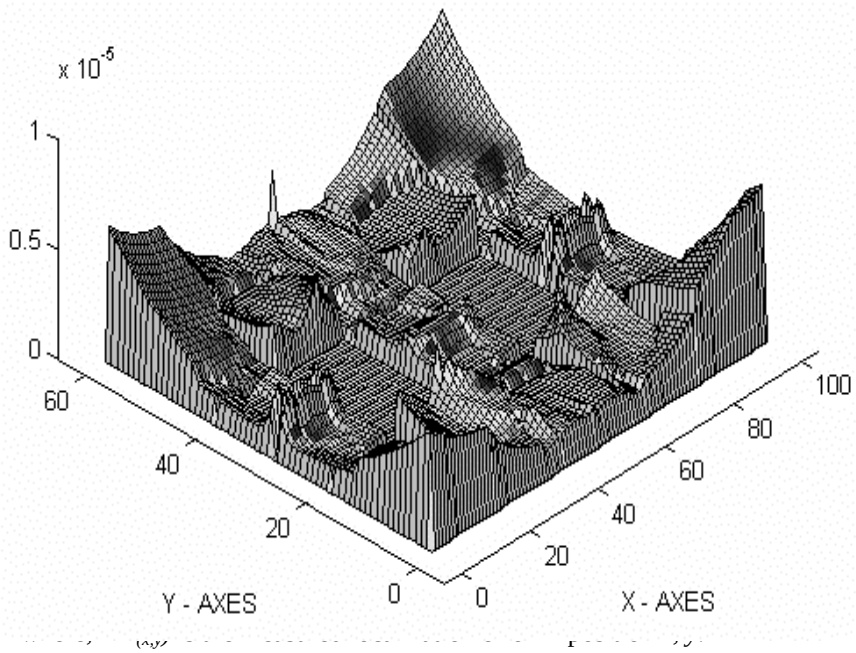


Fig.5
Modeling of the Fault Area

[i] 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 build up on this fact.



D.

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

The reduced navigation path (RL) is given by the piecemeal linearization of the *NP*, see Fig. 7.

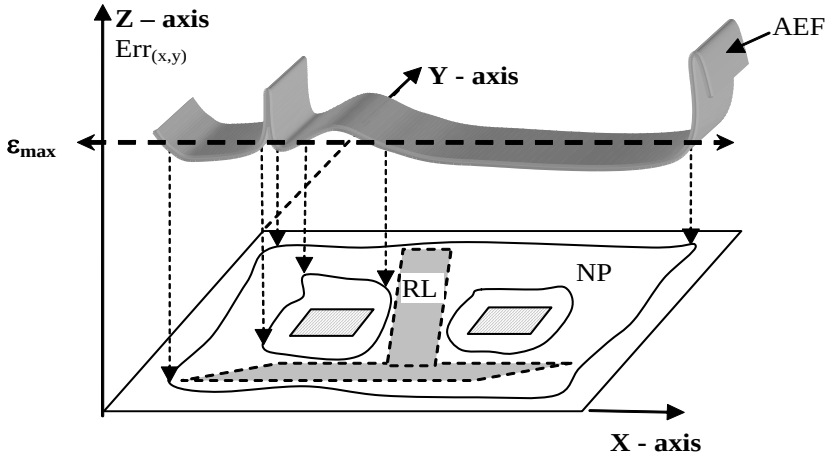


Fig. 7.
Projection onto x, y Plane
and the Rhumb lines (*RL*)

The mathematical explanation of the *NP* and the *RL* see equation (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 consist of edges and nodes. In my case the edges are the centerlines of the *rhumb* lines and the nodes are the cross points of these centerlines. The edges and nodes are weighted differently. A very simplified example is, when the edges are weighted according to its length and the nodes are weighted accordingly to its 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.

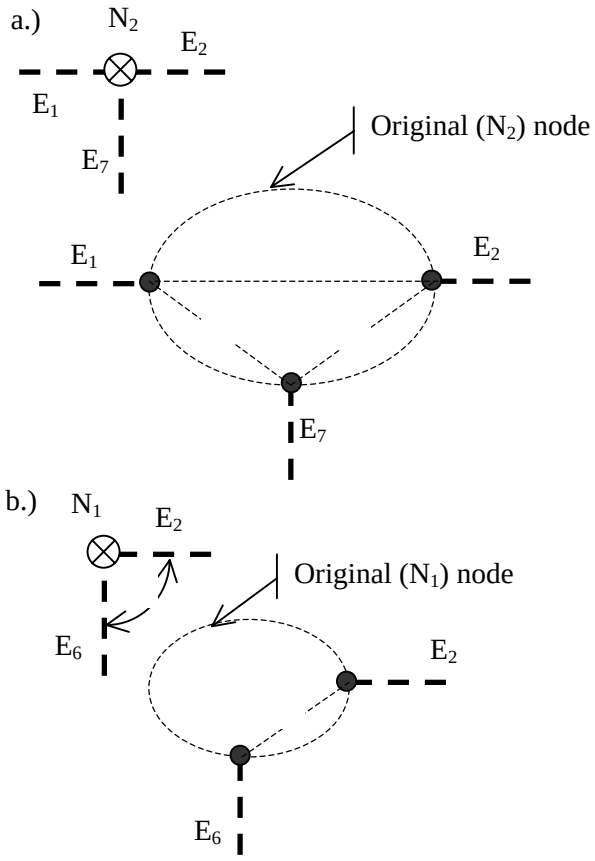


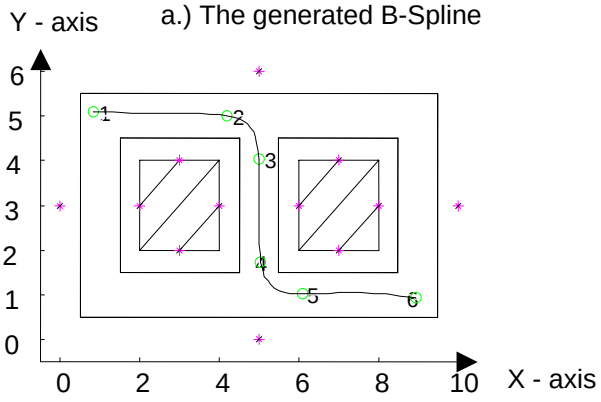
Fig. 8.
Weighting of the Edges and the Nodes

F. *Final route selection* –In the interest of route selection, firstly the goal position has to be designated. After this, the final route is selected accordingly to the weighting of the edges and the nodes. On dependence of 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's 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 case the local minima's are over 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 overflowing the boundary of *NP*, the rules of addition/exclusion of the "passing- through" points can be changed. If this procedure doesn't seem to be efficient, the algorithm has to return to the point *B.1* to generate the 2nd, 3rd, …etc. attempt of the marker's displacement. Finally, we can generate the localization- errors in each point of the final trajectory. See Fig. 9a, 9b,



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

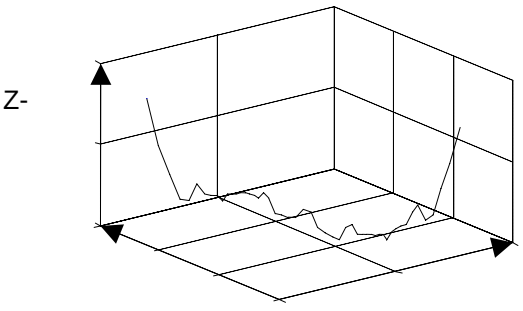


Fig.9.
Spline generation and the Localization Error of the Final Trajectory

where the points in Fig.9a are the "*passing through*" points of the curve. Among these, the point *"1"* is the *START* and the point *"6"* is the *GOAL* position of the mobile robot.

Conclusion

The most of path-planning algorithms in robotics have not considered the dynamical properties of the platform. The use of 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 concrete case the algorithm has been divided for 2 parts. The 1st was a global PF map building process, based on the multi-agent platform, and the 2nd was the AEF calculating and the path planning. The whole second part is pure mathematical, and was built up on the mathematical model of the environment and marker's displacement (Fig. 4.). The possible inaccuracy can be arise at the beginning of 2nd part, with determining the threshold limit. To eliminate this error, we have to establish a coefficient, based on threshold limit, and the mathematical model of the environment has to be calculated through this.

The next important point is the convergence of the algorithm. In case of point represented robot the convergence is insured, because by the re-arrangement (2nd, 3rd, attempt) of the markers, and/or increasing of the number markers we can get always under the given limit of the user defined maximal position error. In case of 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] M. Betke: "Learning and Vision Algorithms for Robot Navigation", PhD-thesis, M.I.T. Dept. of EE&CS., USA, 1992.

[2] R. A. Baeza-Yates, J. C. Culberson, G. J. E. Rawlins: "Searching in the plane", Information and Computation, 106(2):234-252, October 1993.

[3] Ch. H. Papadimitriou, M. Yanakakis,: "Shortest paths without a map", Theoretical Computer Science, 84:127-150, 1991.

[4] I. Nagy, L. Vajta: "Local Trajectory Optimization Based on Dynamical Properties of Mobile Platform", Proc., IEEE Int. Conf. INES'01, Helsinki, Finland, 2001.

[6] J.Liu, J.Wu: „Multi-Agent Robotic Systems", CRC-press, ISBN 084932288, pp186-189, USA, 2001.

[7] I. Nagy: „Marker-Based Mobile Robot Positioning in Respect of Displacement of the Markers", Proc., IEEE Int. Conf., INES'02, Opatija, Croatia, 2002.
[8] I. Nagy, L. Vajta: "Trajectory Planning Based on Position Error Analysis and Fault Area Modeling", Proc. Int. Conf. ICAR'01, Budapest, Hungary, 2001.
[9] V. Lumelsky, A. Stepanov: "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape", Algorithmica, 2(4):403-440, 1987.
[10] J. Somlo, J Podurajev: "Optimal Cruising Trajectory Planning for Robots.", Proc. of the IASTED Int. Conf. Robotics and Manufacturing, Oxford, England, 1993.
[11] Cs. Gürtler, I. Nagy, L. Vajta: "Trajectory Planning for Mobile Robots Based on Dynamical Models", Proc. IEEE. Int. Conf. INES 97, p. 171-174, Budapest, Hungary 1997.
[12] I. Nagy: „Genetic Algorithms Applied for Potential Field Building in Multi-Agent Robotic System", Proc., Int. Conf. on Comp. Cybernetics, ICCC '03, Siófok, Hungary, 2003.
[13] Cs. Gürtler: "Examination of autonomous, sensor-controlled mobile robot's navigational system" BME, Diploma work in Hungarian, Budapest, 1996.
[14] M. Brady, J.M. Hollerbach, T.L. Johnson, T. Lozano-Perez, M.T. Mason: „*Robot Motion: Planning and Control"*. MIT Press, Cambridge, M.A., London, 1982.