TRANSFORMED NET – COLLISION AVOIDANCE ALGORITHM FOR ROBOTIC SOCCER

Martin Saska[1], Miroslav Kulich[2], Gregor Klančar[3], Jan Faigl[1]
[1] CTU in Prague, The Gerstner Laboratory for Intelligent Decision Making
[2]CTU in Prague, Center of Applied Cybernetics
[3]University of Ljubljana, Faculty of Electrical Engineering

Corresponding Author: Martin Saska
Czech Technical University in Prague
Technická 2, 16627 Prague 6, Czech Republic
Phone: + 420 224 355 765, Fax: + 420 224 357 224
email: saska@labe.felk.cvut.cz

Abstract.
Fast and robust obstacle avoidance plays an important role for design of a successful robot soccer team, although not all teams use it nowadays. The standard algorithms assume that a working environment is static or changing slowly. Moreover, computation time and time needed for realization of the planned path is usually not crucial. Speed of robot soccer players (which act as obstacles) can be several meters per second, what requires low reaction time. One criterion for obstacle avoidance is therefore to plan a path far enough from opponent robots to guarantee that their trajectories will not collide with the planned one. This is in contradiction to a primary goal of robot soccer – to reach the desired position as fast as possible. An obstacle avoidance algorithm suited for robot soccer should find acceptable compromise between these two antagonistic requirements. The novel obstacle avoidance algorithm – Transformed Net, satisfying the above-mentioned criterions, is introduced in the paper. The approach is based on searching a graph that covers only most important part of playfield. The output of proposed obstacle avoidance algorithm is a sequence of strategic points to be passed through in order to avoid obstacles. A trajectory generator produces feasible and smooth reference path from this set of points. The robot is then controlled using a classic state space controller to follow a preplanned collision free path. The above motioned obstacle avoidance algorithm has been compared with standard algorithms.

1. Introduction

Obstacle avoidance is an important part of every robotic system and therefore many different algorithms for obstacle avoidance were mentioned in the robotic literature. Unfortunately, most of the methods require static or slowly moving obstacles, which is in contradiction with needs of robotic soccer where robots move very fast (several meters per second). High dynamics of soccer systems is also a reason why time-complexity of planning and obstacle avoidance algorithms should be low. Generated paths should be moreover collision-free, because collisions can cause damage of the robots. It is impossible to prevent all collisions in robotic soccer due to its dynamics, so algorithms it this domain minimize a risk of collisions or its consequences only.

All obstacle avoidance approaches find a path from an actual position S of the controlled robot to a desired goal position C, with respect to positions and shapes of known obstacles P in the environment and dynamic constraints of the robots. While all these parameters stand as the inputs of the algorithm, the output can be either an optimal trajectory from S to C or direction from the actual position respecting locally optimal trajectory. The penalty function to be minimized by the planning algorithm consists of two parts. While the first one is evaluates a length of the trajectory (or time needed to execute the trajectory), the second part takes safety of the path (i.e. distance to obstacles) into account. To find an acceptable compromise between these requirements is on of the crucial problems of the obstacle avoidance.

Standard algorithms described in the literature are either not fast enough or give bad results in dynamic environments as robotic soccer is. It was the reason, why we looked for a new method satisfying all the requirements mentioned above.

The problem of controlling a robot on a path is solved in the second part of the paper. The output of the obstacle avoidance algorithm is passed to the trajectory generator that produces a feasible and smooth reference path from this set of points. The trajectory generator defines a reference path as a smooth path - spline composed of cubic polynomials. A maximal allowable velocity profiles that the robot could still afford due to acceleration limits is then determined. Control of the robot on the reference path is done using a well-known approach [7] with combination of feed forward and closed-loop actions. The former uses robot inverse kinematics to calculate the feed forward inputs from the reference curve, while the latter cancels the effects of noise, disturbances and initial state errors using state space controller.

The rest of the paper is organized as follows. A short review of most popular obstacle avoidance techniques is presented in the second section, while Transformed Net reflecting special requirements of robotic soccer is described in section 3. A trajectory generator smoothing a planned is presented in section 4 and experimental results in section 5.

2. State of the art

Comprehensive review of path planning and obstacle avoidance methods can be found in [1]. Algorithms usually used in mobile robotics are divided there into two types: local and global. Local approaches find optimal direction from the actual position using information from a local area of the robot. While locality of this approach leads to fast planning, trajectories generated by these methods are not guaranteed to be globally optimal. Moreover, unavailability of a full path can cause problems to low-level regulators and strategy planning that can need a full path for their better performance.

Vector field histograms VFH [3], originally developed for obstacle avoidance of robots equipped with sonars are a typical example of local approaches. Similarly to a rotating sonar exploring a space in 360° range, VFH obtains histogram distances to the closest obstacle in each direction. Directions whose distance is smaller than an adaptive threshold are selected from this histogram. Value of the threshold depends on a distance of the robot to the goal position. The direction that has the smallest angle to vector *SC* is chosen as optimal robot heading. VFH cannot solve situations with high density of obstacles and with U-shape obstacles. Both types of the workspace configurations caused movements oscillations.

Output of global algorithms is a complete trajectory from the actual robot position *S* to the desired position *G*. The main advantage of these approaches is ability to avoid a group of obstacles and to find more optimal trajectories as illustrated in Figure 1. Most of them build a graph representing possible robot paths at the first and then finds an optimal (shortest) trajectory in this graph.

The most widely used collision avoidance method in the robotic soccer is potential field [2]. It minimizes a penalty function *Z* that consists of two parts describing influences of the obstacles in the workspace and intention to go to the desired point. The repulsive part discriminates paths that are close to obstacles – it has a maximum in a center of each obstacle and decreases with a distance. The second part is attractive having minimum in the goal of the robot and uniformly growing with a distance to the goal. The biggest problem of this approach is that optimization can finish in a local minimum and therefore a globally optimal path is not guaranteed to be found. We used the algorithm potential field for comparison results with the function

$$Z(x,y)=\sqrt{(x-G_x)^2+(y-G_y)^2}+c_1\left(\frac{1}{x-A_x}-\frac{1}{x-B_x}+\frac{1}{x-A_y}-\frac{1}{x-B_y}\right)+ \\ +c_2.\sum_{j=1}^{N}(\exp(-c_3((x-P_x)^2+(y-P_y)^2))), \tag{1}$$

where *G* is a desired position of the robot, *P* is a center of the obstacle, *A* is left bottom corner of the playground, *B* is a right top corner of the playground and c_i are constants weighting influences of the attractive and repulsive parts.

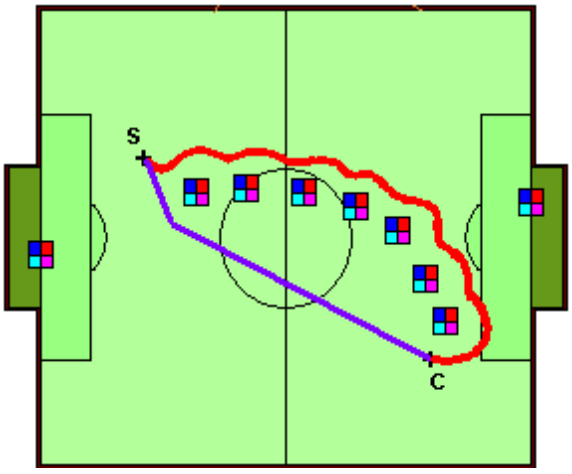

Figure 1. The blue trajectory is obtained by Visibility Graph and the red by Potential field.

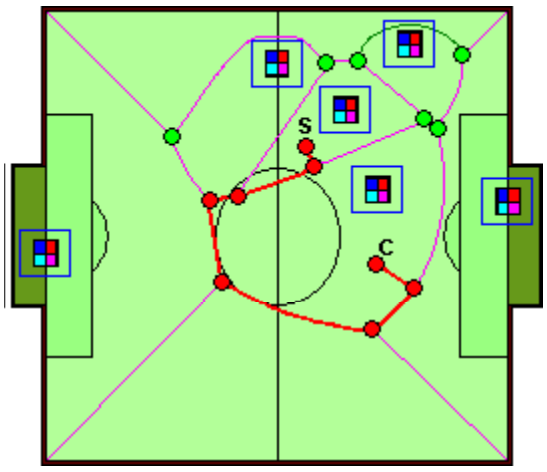

Figure 2. Voronoi diagram used in the robotic soccer.

Visibility graph approach VG [4] constructs a visibility graph of vertices of polygons representing obstacles. It means that two vertices are connected in VG iff there are visible. A shortest path is then determined using standard Dijkstra algorithm [8]. The path found by this algorithm is typically close to the obstacles, which can lead to collision because robots cannot follow planned path precisely. Growing of the obstacles by a certain value can solve this problem, but it is not clear how to optimally determine this value.

Occupancy grid [5] divides a workspace into disjoint cells that cover the whole space. The value of a cell is one if a part of any obstacle is inside the cell or zero otherwise. Centers of the zero-valued cells are nodes of the graph, while edges connect neighboring nodes.

Voronoi diagrams VD [6] divide a workspace into disjoint cells also. Given a set *P* of points in the plane, one cell of VD is a set of points that are closer to the specific point than to any other point in *P*. For robotic purposes, the set *P* contains centers of robots (excluding the one the plan is generated for) plus points representing boards around the playfield. One possibility is to represent each barrier by a set of points placed along each barrier, but it increases computation time. The second way is to compute generalized VD where the set P can contain lines also. As it the previously mentioned algorithms, a shortest path in the graph where neighboring cells of VD are connected can be found by Dijkstra algorithm.

VD finds path that go from obstacles as far as possible. This is useful in dense environments, while paths generated in sparse spaces are needlessly long and cautious (see Figure 2).

3. Transformed Net

In this section we introduce a novel obstacle avoidance algorithm - Transformed Net that is designed to have small computational complexity and to avoid disadvantages of above-described methods. The algorithm should be also able to set weight for safeness (distance to obstacles) and optimality (length) of the path. It is crucial for robotic soccer, because for example intention to avoid obstacles for defender is smaller than for attacker that is not allow to charge the opponent goalkeeper.

The main idea of the algorithm is to cover a whole playfield by a set of nodes and construct a graph, where neighboring nodes are connected by an edge with a weight w.

$$w(e) = length(e)\left(1 + \sum_{j=1}^{n} \frac{c}{d(C(e), C(o_j))}\right), \tag{2}$$

where *length(e)* is a length of edge *e*, d(.) stands for Euclidean distance, *C(e)* is the center of *e*, $C(o_j)$ the center of obstacle o_j, *n* is a number of obstacles and *c* is a constant. This means that the closer is an edge to any obstacle the higher is its weight. The optimal trajectory is then the cheapest path in this graph found by Dijkstra algorithm.

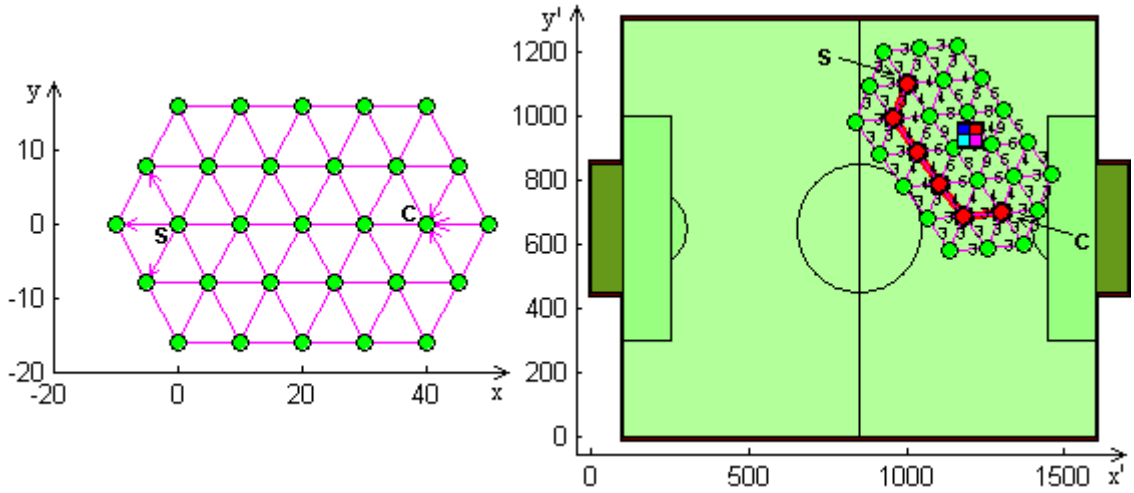

Figure 3. Left: net in the basic form. Right: evaluated Transformed Net. (Edges without arrows direct from the left to the right.)

In order to get optimal results, a distance between neighboring nodes should be comparable to the size of obstacles. This requires generating a net with thousands of edges to cover a robotic soccer playfield. Instead of that only a grid of points covering an area along a straight line connecting robot's actual position with a requested goal position is constructed. A number of points in the net depends on a distance between the start and goal position. In order to increase a speed of the algorithm, the net is not computed on the fly. Instead of this,

several basic nets with a different number of points are pre-computed. An appropriate net is then chosen in a planning phase and transformed (rotated, shifted, and resized) to a correct position (see Figure 3) according to the following equations:

$$x' = \frac{x.\cos(\alpha) - y.\sin(\alpha)}{k} + S_x\,, \tag{3}$$

$$y' = \frac{y.\cos(\alpha) - x.\sin(\alpha)}{k} + S_y\,, \tag{4}$$

where *[x,y]* are coordinates of a node in a predefined net, *[x',y']* coordinates in the resulting net, *[S_x, S_y]* robot start position, and *k* and α scale and rotation parameters, which are calculated as follows:

$$\cos(\alpha) = \frac{C_x - S_x}{d(S,C)}\,, \tag{5}$$

$$\sin(\alpha) = \frac{C_y - S_y}{d(S,C)}\,, \tag{6}$$

$$k = \frac{(w-1).g}{d(S,C)}\,, \tag{7}$$

where *w* is a number of nodes, which lie on the line *SC* and *g* stands for a distance between two neighboring nodes on *SC*.

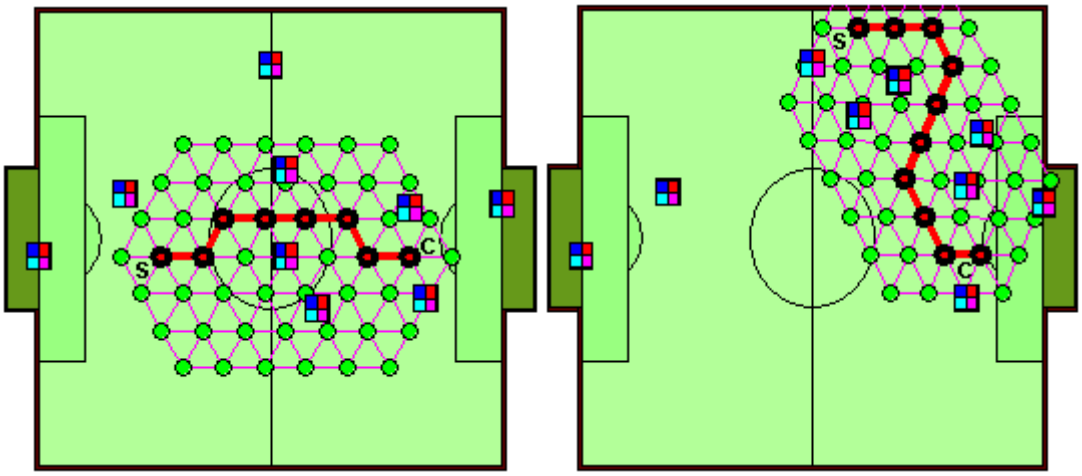

Figure 4. Transformed Net used in situations in the robotic soccer.

4. Trajectory generator

Transformed Net algorithm generates a path as a polyline (i.e. set of points to be visited by a robot supposing that a trajectory between two points is straight line). This kind of path cannot be traversed by a robot precisely due to dynamic and kinematic constraints. The polyline is therefore "smoothed" – approximated by a spline curve that must go thru all points of the planned path and not collide with obstacles.

A two-dimensional curve is obtained by combining two splines, $x(u)$ and $y(u)$, where u is the parameter along the curve. Each spline consists of more segments - cubic polynomials. Knots are points of tangency of two neighbour segments with continuous (the same) derivatives. When the knots are set, the spline parameters can be obtained by solving a linear equation system (8). If the spline consists of m polynomial segments of order p, than the number of parameters to determine is $m \cdot (p+1)$ which requires $n = m \cdot (p+1)$ conditions (linear equations) to completely define spline curve.

The spline curve is therefore fully determined by the start point of the route *S*, an the end point of the route *G*, derivatives in *S* and *G* and knots. The knots are selected to fit in remaining points of the route (all the points except *S* and *G*) and additional points obtained as a geometric average of two neighbouring points of the route. An example of obtained spline paths is given in Figure 6.

A robot will drive on the obtained path if a pure rolling condition of the robot wheels is supposed. This condition is achieved by limitation of the allowed overall acceleration $a = \sqrt{(a_{tang}^2 + a_{rad}^2)}$ which is limited with the friction force and can be decomposed to the tangential acceleration *atang* and to the radial acceleration *arad*.

$$\begin{aligned} a_{tang} &= \tfrac{dv}{dt} \\ a_{rad} &= v \times \omega = v^2 \kappa \end{aligned} \tag{8}$$

where κ is curvature defined as

$$\kappa(u) = \frac{x'(u)y''(u) - x''(u)y'(u)}{\left(x'(u)^2 + y'(u)^2\right)^{3/2}} \tag{9}$$

and derivatives in Eq. (9) are with respect to parameter *u*.

Because the gravity centre of the robot with a differential drive is on a certain height above a ground level, the limit of the tangential acceleration differs from the limit of the radial acceleration. When accelerating in a linear direction, a part of the robot weight is carried by a rear or a front slider, which results in a smaller limit for the tangential acceleration. Measured acceleration limits are shown in Figure 5. (for more details see [10]). The overall acceleration should be somewhere inside or on the ellipse if robots are driven time optimally.

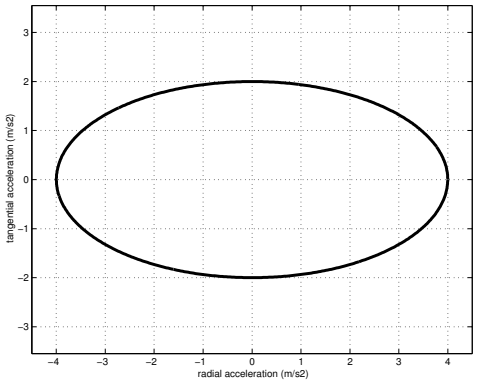

Figure 5. Acceleration limits

For a given path the velocity profile for each robot is calculated as follows. The local extreme (local maximum of absolute value) of the curvature are determined and named turning points (TP). In these points the robot has to move with maximum allowed speed due to radial acceleration limit. Its tangential acceleration *atang* must be 0. Before and after a TP, the robot can move faster, because the curve radius gets bigger than in the TP. Before and after the TP the robot can tangentially decelerate and accelerate respectively as max. allowed by (de)acceleration constraint. In this way the maximum velocity profile is determined for each TP and has the shape of "U" (or "V"). Similarly the maximum velocity profile (due to tangential acceleration/deceleration) is determined for initial point *S* and final point *C* velocity respectively. The highest allowable overall velocity profile is determined as the minimum of all the velocity profiles.

5. Experimental results

The Transformed Net algorithm was tested in two experiments. In the first experiment we studied influence of different algorithm parameters, while in the other one we compared the algorithm with other methods used in robotic soccer domain. For the first experiment we generated an experimental set of 1000 random scenes, for which an optimal trajectory was found by the algorithms. Each scene contains barriers of robotic soccer playfield, 7 randomly generated obstacles (robots), and random start and goal positions of a robot. The set of scenes was a little bit reduced for the second experiment (see section 5.2).

The generated paths were evaluated and compared according to several criteria (see Tables bellow). The first criterion *l* stands for a length of the path; the second one *d* describes ability of the algorithm to find a path in safe distance from all obstacles. Precisely, it is the shortest distance between any obstacle and the trajectory. The values *wc* and *hc* mean a number of collisions. The "weak" collision *wc* occurs if *d* is smaller than the robot radius. In this case the robot lightly touches an obstacle and can push it. In the case *d* is smaller than a half of the robot radius, we speak about a "hard" collision *hc*. The last, but one of the most important criterion is *t* that describes computational complexity. The complexity is described as a ratio of time needed by the algorithm to time needed to solve the same problem by Visibility graph algorithm (it is 7.2ms for 9 obstacles on PC 668MHz, 256 MB RAM).

5.1. Constants setting

In this section we studied influence of different settings of the constant *c* in equation (2) on algorithm behavior and evaluated the results in Table 1. Each row in the table shows values averaged for all 1000 situations from the experimental set.

c[-]	*l*[mm]	*d*[mm]	*hc*[-]	*wc*[-]
500000	801.4	184.4	2	12
50000	791,2	181,9	1	10
25000	784,3	180,3	1	8
5000	717,6	153,6	9	102
2500	707,5	146,2	29	234
250	696,3	139,2	154	309

Table 1: Influence of constant *c*. Size of the net is 11x11 nodes.

Looking at the table, we can see several remarkable facts, which we would like emphasize. First of all, the higher is *c*, the longer is the generated path. It is caused by increasing influence of obstacles, which "pushes" a path to the free space. Thank to this a distance from all obstacles *d* as well as a number of collisions decreases with increasing *c*.

Relation among a size of the net, computational time, and quality of the trajectory is evaluated in Table 2. The rows depict different parameters according to a number of nodes in the net. It can be stated that nets with bigger than 7x7 produce remarkably smaller number of paths with collisions. Interesting behavior is shown in the second and third columns. These two indicators are normally dependent in the sense that a length of the path is increasing when a higher safety is requested, which is not this case: with increasing a net size, paths are shorter and more distant from obstacles. This is caused by finer resolution of larger nets that allow controlling generated paths with higher precision. On the other hand, time complexity is of course higher for bigger nets, so sizes 11x11 and 13x13 looks as a good compromise between quality of generated paths and time complexity.

size[-]	*l*[mm]	*d*[mm]	*hc*[-]	*wc*[-]	*t*[%]
3x3	693,9	138,6	163	310	1,2
5x5	800,0	171,5	26	71	4,0
7x7	791,1	175,6	7	24	7,9
9x9	788,7	179,9	2	10	11,7
11x11	784,3	180,3	1	8	16,9
15x15	782,6	183,2	1	5	40,2
25x25	779,4	184,5	0	2	121,7

Table 2: Influence of different size of the net. Constant *c* is 25000.

5.2. Algorithm comparison with other methods

In this section we describe comparison results of Transformed Net algorithm with two standard methods described above – Potential Field and Visibility Graph. It is not guaranteed that the Potential Field approach finds a trajectory in all situations (even in case a collision-free trajectory exists), because its optimization process can get stuck in a local minimum. If we use the same testing set as in the previous section, the algorithm is successful only in 67.5%. We therefore reduced the set to 675 situations where the Potential Field gets results.

The second column in Table 3 corresponds with expectations. Trajectories found by the Visibility Graph have shortest average lengths. Results obtained by the other approaches are similar – they differ only by 4%. On the other hand, the trajectories found by the Visibility Graph are too close to the obstacles, which causes frequent crashes. This algorithm is therefore not much suitable for a highly dynamic robotic soccer.

All situations used for the statistics in Table 3 were solved without even weak collisions as shown in the third column. That's why we don't use a number of hard collisions as the other indicator, but a number of trajectories *nc* that have a distance to the nearest obstacle smaller than a robot radius multiplied by two. 20% of paths generated by the Visibility Graph algorithm were dangerously near to obstacles, while other two methods give significantly better results.

Algorithm	*l*[mm]	*d*[mm]	*wc*[-]	*nc*[-]	*t*[%]
Potential field	711,1	203,2	0	1	1,8
Transformed Net	741,5	205,2	0	7	28,4
Visibility graph	667,0	175,1	0	134	100

Table 3: Different algorithms used for only 675 situations.

The last indicator in the sixth columns is computational time. Potential Field is fastest according to this criterion, but the value in the table describes time needed to find only an optimal robot heading instead of the whole trajectory. If we use full version of the algorithm generating whole trajectory, computational time decreases dramatically and it is then longer than time needed by the Visibility Graph. The Transformed Net with 13x13 nodes is 3.5 times quicker than the Visibility Graph.

Some graphical results of the second experiment with four typical situations from the robotic soccer solved by compared algorithms are shown in Figure 6. Paths generated by the algorithms were smoothed by the spline generator and final trajectories executed by a robot in the simulator were drown.

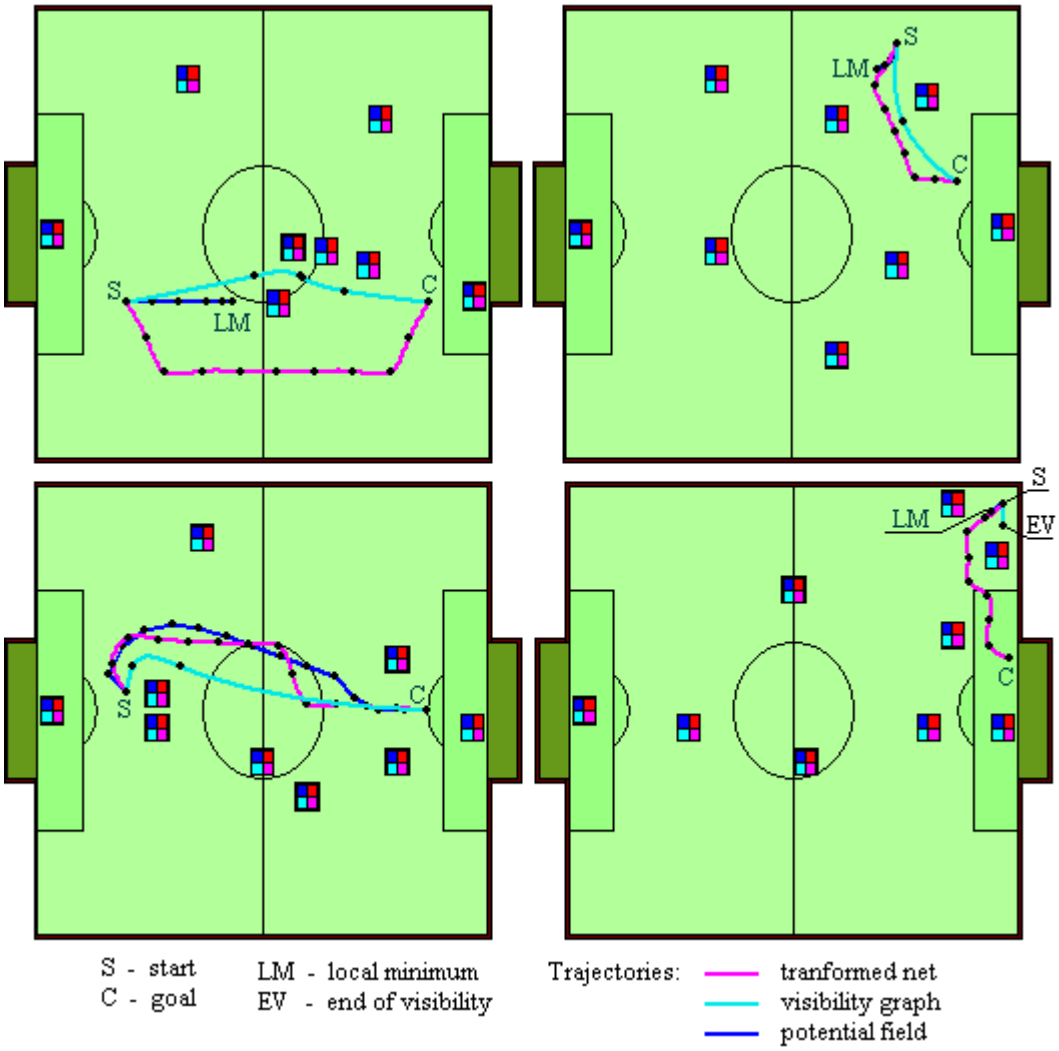

Figure 6. Trajectories used in robotic soccer computed by three methods.

6. Conclusion

In this paper, we propose a novel collision avoidance method based on graph searching. The method was verified in a simulator and compared with standard methods mentioned in the literature – Visibility Graph and potential field.

The Transformed Net algorithm is useful for finding an optimal path in the area of mobile robotic. This approach needs less computational time than the other global algorithms and so it is acceptable in the fast dynamic environments such as robotic soccer. Quickly moving obstacles are the reason, why the algorithm tries

to avoid them with sufficient distance. Other important advantage is the ability to find the trajectory if a free path without collision doesn't exist. Like in a real life, in the robotic soccer holds bad decision is better than no decision, because the robot can shift the obstacles.

The Transformed Net has much lower computational time than other global algorithms. It is of course slower than local algorithms, but these approaches have problems with local extremes and with clusters of obstacles. Potential Field that tries to avoid local minimums is mentioned in [11], but its computational time is similar to time needed by global algorithms.

In the future, we would like to incorporate dynamic obstacles directly into the algorithm. The idea is to predict positions of obstacles (other robots) according to their current states and situation on the field and evaluate edges in the planning graph according to these predictions. Other stream will be focused on integration of dynamic and kinematic constraints into the algorithm.

Acknowledgement

The work has been supported within the Czech-Slovenian intergovernmental S&T Cooperation Programme under project no. 10200506 "Cooperative mobile robots for industry and service". The support of the Ministry of Education of the Czech Republic, under the Project No. 1M0567 to Miroslav Kulich is also gratefully acknowledged.

References

[1] J.C. Latombe, Robot Motion Planning, Norwell, MA: Kluwer,1991.

[2] O.Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. The International Journal of Robotics Research,5(1):90-98. Spring, 1986.

[3] J. Borenstein, Y. Koren. The Vector Field Histogram: Fast Obstacle Avoidance for Mobile Robots. IEEE Journal of Robotics and Automation, 7(3):278–288, June 1991.

[4] R.Kunigahalli, J.S.Russell. Visibility Graph Approach to Detailed Path Planning in CNC Concrete Placement. Automation and Robotics in Construction XI, 141-7. Elsevier, May 1994.

[5] T.Braunl, N.Tay. Combining configuration space and occupancy grid for robot navigation. Industrial Robot, 28(3): 233-41. MCB University Press, 2001.

[6] F.Aurenhammer, R. Klein. Voronoi diagrams. Hand book of Computational Geometry, chapter V, pages 201-290.Elsevier Science Publishers, Amsterodam, 2000.

[7] A. Luca, G. Oriolo. Modelling and control of nonholonomic mechanical systems, *Kinematics and Dynamics of Multi-Body Systems*, J. Angeles, A. Kecskemethy Eds., Springer-Verlag, Wien, 1995.

[8] B.J.Jorgen, G.Gutin. Digraphs: Theory, Algorithms and Applications. Elsevier North Holland, New York, 1979.[1] The MathWorks Inc., Spline Toolbox User's Guide, Version 2, 1999.

[9] The MathWorks Inc., Spline Toolbox User's Guide, Version 2, 1999.

[10] M. Lepetič, G. Klančar, I. Škrjanc, D. Matko, B. Potočnik. Time optimal planning considering acceleration limits, *Robotics and Autonomous Systems*, vol. 45, pp. 199-210, 2003.

[11] I.C.Connolly, R.A.Grupen. On the Applications of Harmonic Functions to Robotics. Journal of Robotics Systems, 10(7):931–946, October 1993.