

Motion Coordination Method for Numerous Groups of Fast Mobile Robots

Wojciech Turek

Institute of Computer Science
AGH University of Science and Technology, Krakow, Poland

Abstract

The solution presented in this paper is a real-time motion coordination method, that considers robot's dynamics constraints during path planing and path execution. The environment is represented as a dense graph of roads and junctions; path planning algorithm finds a collision-free, time-optimal path, using the graph. Robot's dynamics characteristics is used during planning to create a feasible trajectory and during execution to precisely follow the planned path. Exemplary implementation for FIRA soccer-playing robots has been created and several tests have been performed to verify the solution.

Keywords: mobile robot, motion coordination

1 Introduction

Mobile robots have been found useful in many applications over the last few decades. Systems of automated guided vehicles (AGVs) (6) are widespread used for repeated transportation tasks in factories and warehouses. More advanced solutions, based on semi-autonomous robots, are being introduced into inspection and guarding systems. Every mobile robot system requires a few low-level algorithms, that need to be provided to ensure system stability and robustness. One of the most basic issues, that has to be solved, is a path planning and motion coordination problem.

The path planning problem is relatively simple, if a single robot is considered, the environment is static, and robot's dynamics can be ignored. However, finding optimal (the fastest), collision-free trajectories for multiple robots in a dynamic environment is an NP-hard problem. When limited linear acceleration and maximum acceptable centripetal acceleration is taken under consideration, the problem is becoming extremely complex.

Solutions, that can be found in the literature, usually divide the problem into two phases: planning and execution. Different aspects of the problem can be address in a different phase. In simple environments, with relatively slow robots, satisfactory results can be achieved without planning phase, by using purely reactive algorithms. In (8) a sonar-based method is described, which allows robot to follow a smooth path between simple obstacles.

More complex environments require use of path planning algorithm, to prevent falling into local distance minima. Global path planning is usually based on a graph representation of the environment. In AGV systems the graph is typically determined by predefined roads, used by vehicles. Path planning is a straightforward, shortest path search problem, where each edge is characterized by an average travel time. It is usually assumed, that each edge can be occupied by one robot a time. In (7) author presents a solution to the deadlock detection problem, which can easily arise in this type of system.

If the environment is not limited to a set of predefined paths, the motion coordination problem becomes significantly more complex. There are several solutions, that introduce planning priorities to address the problem in the "open space" (1)(3). At first each robot plans its path, avoiding static obstacles only – a simple, distance-based A* algorithm is applied. Planned paths are then analysed, in order to detect possible collisions in three-dimensional space (x, y, time). Robots with lower priorities must delay their motion to avoid collisions. Many different prioritizing strategies can be used to minimize overall travel time. The greatest drawback of this approach is, that the paths must be planned simultaneously – no real-time paths addition during execution is considered.

Robot's dynamics constraints are rarely taken under consideration in the planning phase of motion coordination. In (5) authors represent a robot's trajectory as a parametric curve consisting in a sum of harmonics. This representation makes it possible to apply a numerical optimization methods. High computational complexity of the approach makes it difficult to apply in motion coordination of numerous groups of robots.

The solution proposed in this paper is a real-time motion coordination method, that considers robot's dynamics constraints in the planning phase. It is focused on maximizing the number of robots, that can simultaneously work in relatively small space.

The key idea is to cover the space with a dense network of virtual roads and junctions, and route robots using these roads. Motion coordination is based on temporary reservation of junctions by robots. Path planning algorithm uses the network to find collision-free, time-optimal trajectories, which meet dynamics constraints.

The solution is intended to be applied in complex agent-based systems designated for solving task management problems. The algorithm will be used for coordinating robots motion in selected regions of the workspace, where large numbers of robots may travel simultaneously.

In the next section the method for building roads network, and finding time-optimal trajectories will be described. In the following sections the motion coordination algorithm will be presented and the motion equations for a robot executing the planned path will be discussed. Finally, the implementation of the test system and the experiments results will be presented.

2 Environment representation and time-optimal path planning

It is assumed, that the environment in which a group of robots is performing tasks, is a known, 2-dimensional space with a number of obstacles. The algorithm covers accessible area with a dense graph, which represents accessible roads(edges) and junctions(nodes). The distance between closest nodes of the graph is constant – it should be equal to robot’s diameter, increased by localization error. Two robots located in two adjacent nodes should never collide. The implementation assumes that all robots are similar in size and shape. The method can be generalized to remove the assumption, making the algorithm far more complex to implement. The assumption does not affect conclusions and results presented in this paper.

To build the graph, the occupancy grid method is used. A node is placed in the middle of each cell, if it does not contain any fragments of obstacles. Edges are then created between pairs of neighbour nodes in horizontal and vertical direction. Additional diagonal edges are created, if a square defined by four closest nodes is empty. Distance between closest nodes (grid resolution) should allow safe placing of two robots in both nodes. An example of a graph fragment is presented in the figure 1.

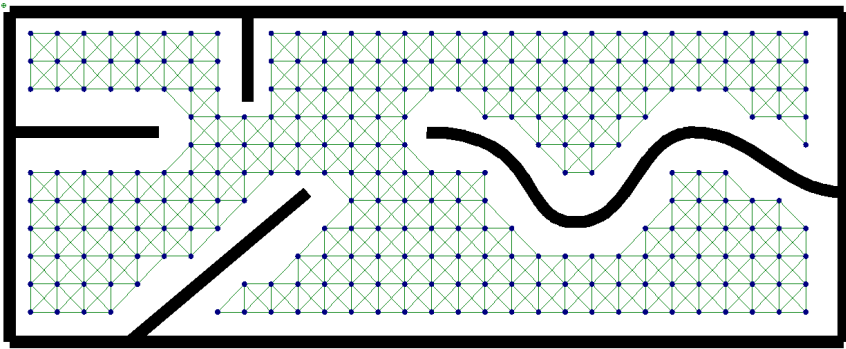


FIGURE 1: Example of an environment and a created graph.

A path from node s to d consists of a list of **steps**. A step is a data structure, which contains information about:

- previous step,
- the node, in which the step ends,
- linear velocity of a robot at the node,
- one of eight possible directions in the node,
- estimated time of arrival (ETA).

The time-optimal path planning algorithm works on a ETA-ordered queue, containing steps, that should be processed. It starts, creating a single step for a node, in which the robot is located (s). Each queued step is processed by several

sub-algorithms, called Step Solvers. For a given step, a Step Solver creates several following steps. Created steps are enqueued for processing, if no faster step to the particular node with the same direction, has been found before.

This procedure continues, until a step to the d node is found, and all enqueued steps have ETA greater than the one. The list of steps leading to the final step defines a time-optimal trajectory between the s and the d node.

Steps returned by different Step Solvers lead to different nodes. The simplest, rotating Solver changes only direction of a robot, without changing the node – it is valid only for differential-drive, holonomic robots. Different Solver moves a robot straight, to the next node ahead. Other Solvers perform turns along different curves. Paths planned by different turning Solvers are shown in figure 2.

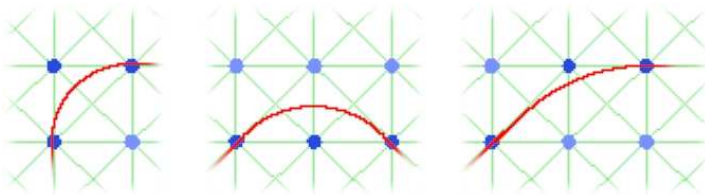


FIGURE 2: Types of turns, that can be planned by different Solvers.

Obviously, many different Solvers could be created, to enable smoother path planning. However, in the following section, some disadvantages of long steps will be described.

Solvers calculate ETA of new steps according to dynamics constraints, that characterize moving robot. The constraints are:

- maximum linear velocity,
- maximum linear acceleration,
- maximum linear deceleration,
- maximum centripetal acceleration,
- maximum angular velocity,
- maximum angular acceleration.

Linear velocity is always increased by Solvers, until one of the limits is reached. However most solvers have initial requirements, concerning the limits (e.g. the turning Solver requires linear velocity equal to 0). Therefore each Solver must be able to convert a planned step into a version with desired final velocity. This operation is always feasible and has an unambiguous solution, provided that previous step's Solver also supports it. It is also used to stop the robot at the destination node.

Implemented version of path planning algorithm is computationally expensive. It seeks for solution in every possible direction, gathering information about fastest way of reaching every node, until the destination node is found. The complexity is directly proportional to the number of nodes in the graph, and to the number

of Solvers used. Overall performance could be significantly improved by applying an A*-based heuristics (4) for searching the graph.

3 Collision-free path planning

Planned trajectory consist of a number of steps. For each of the step, a Solver must create temporary reservations for nodes, that should not be used by other robots. Time of reservation is determined by the ETA of the step and the step before. All nodes used by a step are locked for the same time, therefore Solvers should use as few nodes as possible.

There are two types of node reservations: exclusive and non-exclusive. Non-exclusively reserved nodes can be reserved non-exclusively by a different robot in an overlapping period. Only turning Solvers use non-exclusive nodes reservations; those are marked with a lighter colour in figure 2.

During the path planning process, each Solver must verify planned steps, not to violate the reservations, that are already created. If a reservation period collision is detected, the step beginning time (ETA of a previous step) must be delayed by the previous step's Solver. This operation can be solved in many ways, because redundant time can be spent in different locations. Applied algorithm always postpones the delay as late as possible, to reduce the number of steps to be modified.

Delaying preceding steps can cause other reservations violations, which have to be resolved. This can result in inability of reaching the considered node precisely on the specified time. Therefore, the delaying operation is supposed to return a previous step with the ETA not smaller than desired. The operation is always feasible, although the time returned may be infinite if an infinite reservation is violated.

4 Path execution

Theoretically, there should be no problems with executing a correctly planned sequence of steps. Unfortunately, due to numerous errors in position and direction measurement, velocity estimation, communication delays etc., planned locations and velocities must be treated as a task, not as a control sequence.

Applied algorithm of step execution separates linear and angular velocity calculations. This approach makes it possible to use the same linear velocity equations for all types of steps.

In the first place, length of a curve fragment that a robot should follow to reach step's final location, must be calculated. The curve is approximated by an arc of a circle, that is tangent to robot's current direction and to desired step direction.

Calculated length, desired velocity and time at the end of the step are the constrains, that have to be met by manipulating linear velocity of the robot. Figure 3 shows relation between time and velocity of a robot. The darken region marks the velocities, that can be developed during the step (according to robot's acceleration characteristics), without violating the desired velocity constraint.

It is assumed, that robot's velocity can change according to maximum acceleration or deceleration, or remain constant – which is typical for most mobile robots.

Then the distance, covered by a robot can be expressed as:

$$d = \int_{t_0}^{ETA} v(t)dt = \frac{(v_b + v_c)}{2} * t_b + v_c * t_c + \frac{(v_c + v_d)}{2} * t_d \quad (1)$$

where:

d – remaining distance,

t_0 – current time

v_b – beginning velocity,

v_c – constant velocity, that should be developed by the robot,

v_d – desired velocity at the end of the step

$t_b + t_c + t_d = ETA - t_0 = t$.

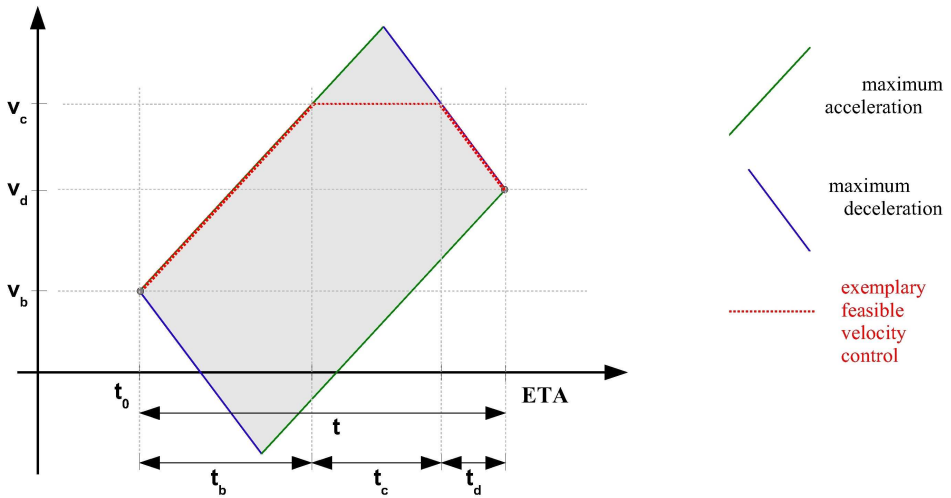


FIGURE 3: Relation between time and velocity during a step.

Values of t_b , t_c and t_d can be expressed as:

$$t_b = \frac{v_c - v_b}{a_1}, \quad t_d = \frac{v_d - v_c}{a_2}, \quad t_c = t - (t_b + t_d), \quad (2)$$

where each of a_1 and a_2 can be either acceleration or deceleration, depending on situation. Equation 1 can be expressed as:

$$v_c^2 \left(\frac{1}{2 * a_2} - \frac{1}{2 * a_1} \right) + v_c \left(t + \frac{v_b}{a_1} - \frac{v_d}{a_2} \right) + \left(\frac{v_d^2}{2 * a_2} - \frac{v_b^2}{2 * a_1} - d \right) = 0. \quad (3)$$

Assuming non-negative values of t_b , t_c and t_d , this quadratic equation has one or no solutions. If a solution cannot be found, at least one of the step constraints must be violated.

Calculated v_c is a linear velocity, which should be combined with second derivative of a curve that robot should follow – exact equations depend on robots shape and dimensions. After that, the value can be passed directly to robot’s engines controllers – provided that acceleration characteristics has been measured accurately, robot’s velocity should follow desired changes.

5 Implementation and experiments

The implementation of the solution was designed for a group of FIRA MiroSot (9) robots, which are small, differential-drive cubes (7.5cm each dimension). Originally the robots were designated for playing robot soccer, but they can be successfully used as a model of a complex multi-robot system. Following values were measured, while testing robot’s performance:

- maximum linear velocity: $4.5m/s$, limited to $1.0m/s$ due to lack of space and problems with localization,
- maximum linear acceleration: $2.1m/s^2$,
- maximum linear deceleration: $2.4m/s^2$,
- maximum centripetal acceleration: $0.6m/s^2$,
- maximum angular velocity: limited to $3\pi/s$ due to problems with localization,
- maximum angular acceleration: $2\pi/s^2$.

The robots are not equipped with any sensors or high-performance computational unit. All algorithms are executed on an external computer, which sends control orders to robots via Bluetooth serial connection. Information about current state of the system are gathered by a camera, which is situated over the environment. Each robot has a unique set of colourful markers, used by image recognition algorithm for localization. A scheme of the robot system is shown in figure 4.

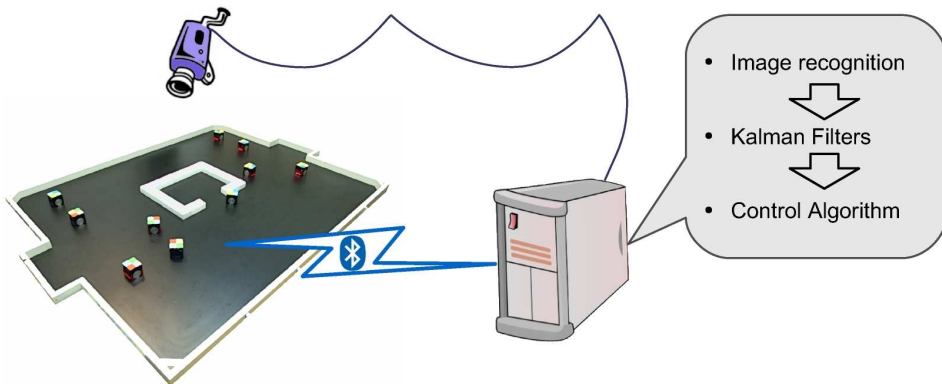


FIGURE 4: Multirobot system used for testing the solution.

The camera provides a new frame every 0.04 seconds, which restricts system's refresh rate to about 25 steps per second. Image recognition algorithm can locate robots moving up to approximately 1 m/s – above this value the image is too blurred. Results of recognition are saddled with random errors caused mostly by camera's noise, which can reach up to $0.003m$ in position, and up to 0.05π in direction. Moreover the information provided to the control algorithm is significantly delayed in respect to the current state – image transmission and recognition can take up to 0.1 second, which can cause localization error of $0.1m$ and orientation error of 0.3π .

This scale of localization and orientation errors would make path execution infeasible. Therefore a set of Kalman filters (2) had to be applied to process states, controls and predict current state of the system. Each robot has a separated extended Kalman filter for direction and each of position dimension. The filters correct the state using 100 ms old information from camera, and predict current state using controls, that have been applied by the algorithm in the meantime. This solution reduced localization errors to approximately $0.01m$ and orientation errors to 0.1π .

Errors values must be taken under consideration while estimating the minimum distance between nodes of the graph covering the environment. In case of FIRA robots, the distance was set to $0.12m$, which is robot's diameter plus double localization error.

Numerous test of the system have been carried out. The results of the most significant are presented in the following sections.

5.1 Narrow passage management.

The environment used in this example is presented in figure 5. Each of the robots taking part in the test was ordered to move to the other side of the obstacles and back.

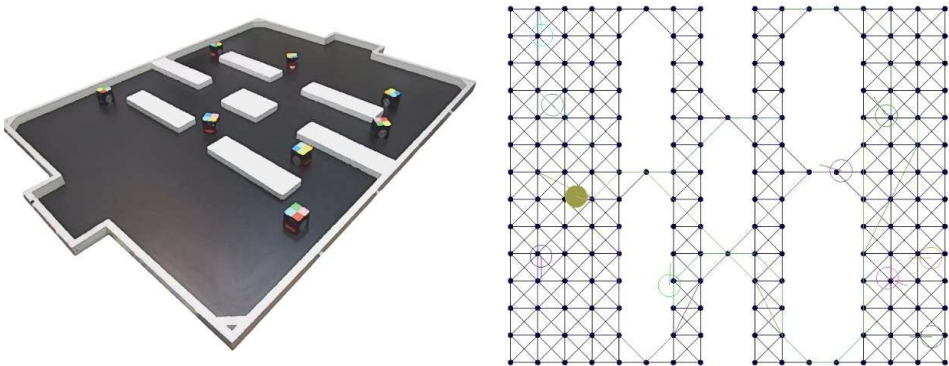


FIGURE 5: The environment and the graph used in the narrow passage management test (different robots layout).

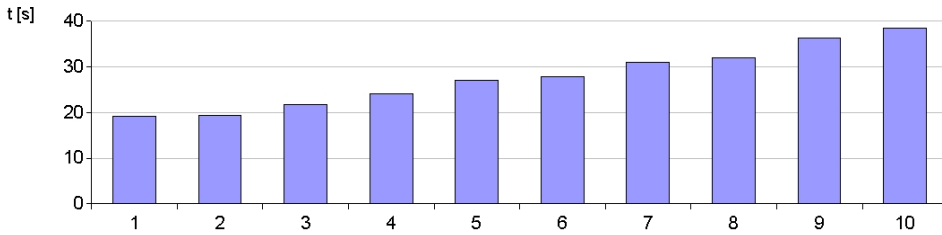


FIGURE 6: Results of the passage management test.

Overall time of the operation was measured, depending on number of robots used. The results are presented in figure 6.

There are two independent paths crossing the obstacles. Therefore the time for two robots is almost identical as for one. Above that value, total time of the movement is raising linearly, which is an expected characteristics. It can be noticed, that addition of an odd robot usually causes bigger raise in time, than an odd robot, which is also caused by an even number of independent passages. No collisions or deadlocks occurred while performing the test.

5.2 Comparison with a robust reactive method

To illustrate features of the method and the advantages over simpler solutions, a comparison with a robust reactive method have been performed. The reactive method used, had been originally created for robot soccer games. The general idea of the method is to detect close obstacles (static or dynamic) on the way to a destination, and select a temporary detour destination, to avoid collisions. An exemplary situation is shown in figure 7. The green robot was ordered to move forward; in second and third frame obstacles are detected, and a temporary destination is selected (marked by a blue '+').

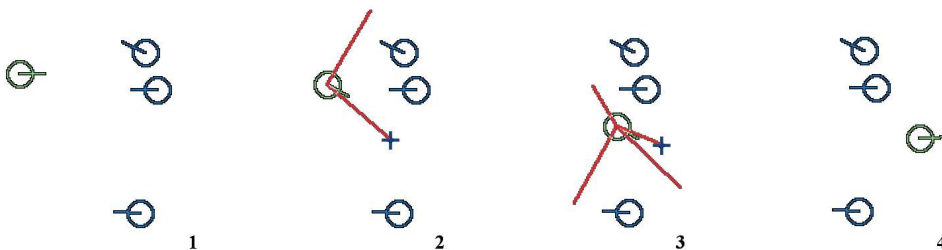


FIGURE 7: Reactive, angle-locking algorithm for collision avoidance.

The algorithm was designated for the same FIRA robots – it optimizes decisions according to robots characteristics and dynamic state. It performed very well in several soccer team control programs.

To reduce the influence of obvious advantages of global path planning approach, no static obstacles were used in the environment for this scenario. Similar tasks were assigned – each robot was ordered to move to the other side of an empty pitch, and back.

Overall time of motion depending on number of robots used, is shown in figure 8.

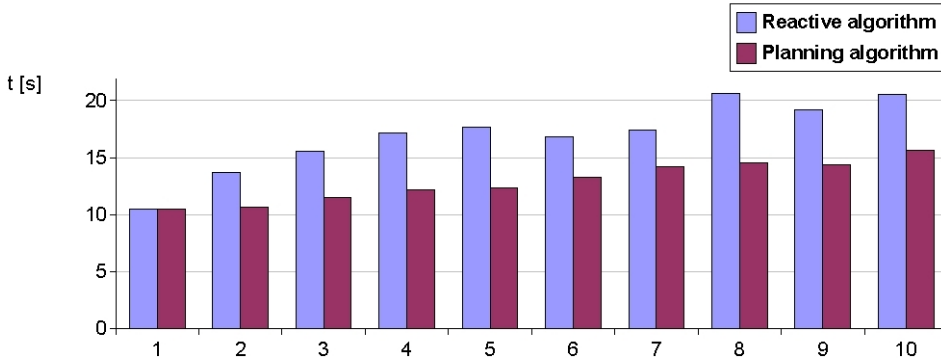


FIGURE 8: Results of the comparison test.

The path planning algorithm performed far better in all cases, except a single robot movement. Moreover, results of the reactive algorithm are very unreliable – large variations occurred among several test runs, due to numerous random collisions and temporary deadlocks.

6 Conclusions and further work

The solution presented in this paper seems to be a promising approach to the problem of real-time motion coordination for numerous group of robots in limited space. Performed tests show, that the algorithm can perform much better than a robust reactive approach, even for very simple tasks. In more complex environments, where global path planning approach is inevitable, the method gave very satisfactory results.

There are several aspects, that need further investigation. The performance of the path planning algorithm can be significantly improved, by adopting an A*-based heuristics. More optimal overall performance could probably be achieved, if an algorithm was developed, which could partially modify paths already being executed.

The solution is going to be applied in complex agent systems, used for robots management. It will be responsible for coordinating motion in most crucial areas of the environment.

References

- [1] K. Azarm, G. Schmidt. Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 3526–3533, Albuquerque, NM, USA, 1997.
- [2] Y. Bar-Shalom, T. E. Fortmann. Tracking and Data Association. *Academic Press*, 1987.
- [3] M. Bennewitz, W. Burgard, and S. Thrun. Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots. *Robotics and Autonomous Systems*, 41(2):89–99, 2002.
- [4] R. Dechter, P. Judea. Generalized best-first search strategies and the optimality of A*. *Journal of the ACM*, 32(3):505–536, 1985.
- [5] P. Gallina, A. Gasparetto. A technique to analytically formulate and to solve the 2-dimensional constrained trajectory planning problem for a mobile robot. *Journal of Intelligent and Robotic Systems*, 41:237–262, 2000.
- [6] I. F. A. Iris. Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research*, 170(3):677–709, 2006.
- [7] E. Roszkowska. Liveness enforcing in closed agv systems with dynamic routing. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 5165–5170, 2004.
- [8] C. Urdiales, E.J. Perez, J. Vazquez-Salceda, M. Sanchez-Marre, and Sandoval F. A purely reactive navigation scheme for dynamic environments using case-based reasoning. *Autonomous Robots*, 21(1):65–78, 2006.
- [9] Specification of the FIRA MiroSot league.
<http://www.fira.net/soccer/mirosot/MiroSot.pdf>, 2007.

