Dynamic window based approach to mobile robot motion control in the presence of moving obstacles

Conference Paper *in* Proceedings - IEEE International Conference on Robotics and Automation · May 2007

DOI: 10.1109/ROBOT.2007.363613 · Source: IEEE Xplore

2 authors:

Marija Seder
University of Zagreb
33 PUBLICATIONS **420** CITATIONS
SEE PROFILE

Ivan Petrovic
University of Zagreb
233 PUBLICATIONS **2,141** CITATIONS
SEE PROFILE

Dynamic window based approach to mobile robot motion control in the presence of moving obstacles

Marija Seder and Ivan Petrović

***Abstract*— This paper presents a motion control method for mobile robots in partially unknown environments populated with moving obstacles. The proposed method is based on the integration of focused D* search algorithm and dynamic window local obstacle avoidance algorithm with some adaptations that provide efficient avoidance of moving obstacles. The moving obstacles are modelled as moving cells in the occupancy grid map and their motion is predicted by applying a procedure similar to the dynamic window approach. The collision points of the robot predicted trajectory and moving cells predicted trajectories form the new fictive obstacles in the environment, which should be avoided. The algorithms are implemented and verified using a Pioneer 3DX mobile robot equipped with laser range finder.**

I. Introduction

A mobile robot should perform goal-directed tasks in dynamic and unknown environments. Both global path planning and local reactive obstacle avoidance algorithms must be implemented and integrated in a single motion control module in order to provide the robot with this capability. While a global path planning algorithm calculates optimal path to a specified goal, a reactive local obstacle avoidance module ensures tracking of the global path and takes into account the unknown and changing characteristics of the environment based on the local sensory information.

Global path planning is well studied by Latombe in [1] where techniques such as cell-decomposition and road map are examined. Global path planning algorithms produce a graph of possible paths to the goal and then an optimal path is found by a graph search algorithm such as A* [3], D* [4] or focused D* (FD*) [5] algorithm.

Some of the most popular reactive collision avoidance techniques can be divided into directional [6]-[8] and velocity space based approaches [9]-[11]. The directional approaches generate a direction of the robot to head in. Velocity space approaches take robot's kinematic and dynamic constraints directly into account by performing a search in space of translational and rotational velocities. The most popular and here used velocity space approach is Dynamic Window (DW) approach [11].

The major challenge in the motion control of mobile robots is to solve the problems caused by the presence of moving obstacles. Two major categories of the approaches are: the space-time approaches and the potential field extension. While the space-time approaches assume a priori knowledge of the moving obstacles trajectories the potential field extension does not and generates an artificial force that repels the robot from obstacles instead [6]. For example, some of the extensive space-time approaches [12]-[13] assume a moving obstacle as stationary at some location in some future time instant based on the known moving obstacle trajectory. Functionality of these approaches depends on a success of moving obstacle tracking algorithms.

In this paper the proposed approach fits to the space-time approaches. We extend our original motion control method, which is based on the integration of FD* and DW algorithms [15]. While the original integration method provides reliable real-time control of mobile robot in partially unknown environments with static obstacles, it does not ensure collision-free motion in environments populated with moving obstacles. With three modifications introduced in this paper our method ensures safe and smooth robot motion in the presence of moving obstacles.

M. Seder and I. Petrović are with Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia, e-mail: {marija.seder, ivan.petrovic}@fer.hr

II. The Original FD* + DW control method

Here we shortly present control method from our previous paper which integrates FD* and DW algorithms.

A. FD* Path Planning Algorithm

Graph based search algorithms are the most commonly used algorithms for path planning of mobile robots. We use the FD* graph search algorithm [5], which is based on a path cost function g that represents the total cost from the current node of the search to the goal node, and a heuristic function h, which estimates but never overestimates the cheapest solution for achieving the current node from the start node in the (x, y) grid map search space. The total cost function $f = g + h$ determines order of expanding nodes in state space.

FD* search fans out from the goal node, expanding neighbor nodes within the contours of increasing f-value until the start node is reached or all possible obstacle free neighbors are exhausted upon which the algorithm declares no path is found. Initial search by FD* algorithm sets pointer from each state in the searched area to the next state and optimal paths to the goal from every state in the expanded area of the environment are computed simply following the pointers. Any discrepancy that is discovered from the earlier sensory information about the vicinity of the robot environment initiates algorithm on-line execution. New path is then determined redirecting the pointers locally. The number of expanded nodes is minimal and consequently the time of execution.

B. Dynamic Window Obstacle Avoidance

The dynamic window approach is a velocity space based local reactive avoidance technique where search for commands controlling the robot is carried out directly in the space of velocities. Trajectory of a robot can be described by a sequence of circular and straight line arcs as given in the original paper [11].

The search space is reduced by the kinematic and dynamic constraints of the robot to a certain span of velocities around the current velocity vector (v_c, ω_c) that can be reached within the next sampling interval Δt. The dynamic window V_d containing the possible reachable velocities is defined as [11]:

$$V_d = \left\{ (v,\omega) \mid \begin{array}{c} v \in [v_c - \dot{v}_b \Delta t, v_c + \dot{v}_a \Delta t] \wedge \\ \omega \in [\omega_c - \dot{\omega}_b \Delta t, \omega_c + \dot{\omega}_a \Delta t] \end{array} \right\}, \quad (1)$$

where accelerations $\dot{v}_a$ and $\dot{\omega}_a$ are maximal translational and rotational accelerations exertable by the motors and $\dot{v}_b$ and $\dot{\omega}_b$ are maximal translational and rotational breakage decelerations.

A velocity tuple (v, ω) from the V_d is considered safe if the robot is able to stop along the trajectory defined by (v, ω) before hitting any object that may be encountered along that path. The set V_a of admissible velocities can be determined according to:

$$V_a = \left\{ v, \omega \leq \sqrt{2\rho_{min}(v,\omega)\dot{v}_b} \wedge \sqrt{2\rho_{min}(v,\omega)\dot{\omega}_b} \right\}, \quad (2)$$

where the term $\rho_{min}(v, \omega)$ represents the distance to the closest obstacle on the corresponding curvature.

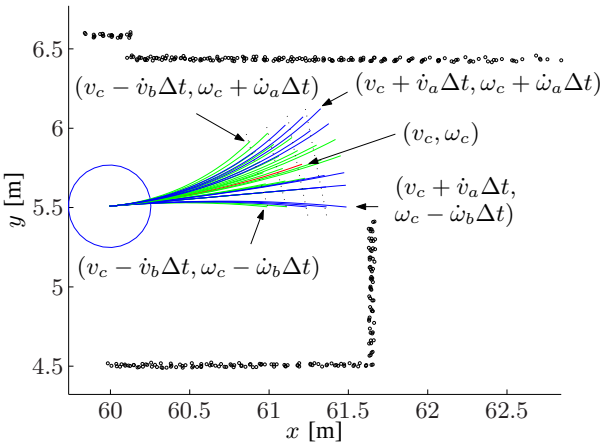

Fig. 1. A snapshot of possible robot trajectories

To make the search for velocities feasible and appropriate for fast reactive response, the dynamic window approach considers exclusively the first sampling interval to choose the optimal velocity vector and assumes that the velocities in the remaining $n-1$ sampling intervals are constant. The reduced search space is two-dimensional over the discrete set of velocity tuple (v, ω) and thus feasible in polynomial time search sense. The search is repeated after each sampling interval and the velocities stay automatically constant if no new commands are given. A snapshot of possible robot trajectories at given time and local obstacle configuration determined by reduced velocity space is depicted in Fig. 1.

The velocity maximizing a certain objective function $\Gamma(v, \omega)$ [11] is chosen from the reduced set of velocities. As reported in [9] and [11], this approach is susceptible to local minima without further global path information. If connectivity of free-space toward goal position is considered, local minima can be avoided. There are a number of more or less successful approaches, such as those described in [5], [2], [16], [17]. However, each of them possess some limitations, which can cause unsatisfactory robot motion. In our previous work [15] we proposed a new approaches based on the integration of DW and FD* algorithms which avoids those limitations. It is shortly described in the next section.

C. Integration of Dynamic Window and FD* Path Planning

A criterion for DW and FD* integration is proposed, which makes comparison of the current possible robot trajectories and the geometric global path by applying the global objective function $\Gamma(v, \omega)$ expressed as weighted sum of objective measures for clearance $\vartheta_{clear}(v, \omega)$ and path alignment $\vartheta_{path}(v, \omega)$ [15]:

$$\Gamma(v,\omega) = \lambda\vartheta_{clear} + (1-\lambda)\vartheta_{path}, \quad (3)$$

where λ is the weighting factor. The clearance objective measure ϑ_{clear} describes how close is a chosen trajectory to potential obstacles:

$$\vartheta_{clear}(v,\omega) = \left\{ \begin{array}{rcl} 0 & : & t_{col} \leq T_b \\ \frac{t_{col}-T_b}{T_{max}-T_b} & : & T_b < t_{col} < T_{max}, \\ 1 & : & t_{col} > T_{max} \end{array} \right. \quad (4)$$

where $T_b(v, \omega) = max(\frac{v}{\dot{v}_b}, \frac{\omega}{\dot{\omega}_b})$ is the breakage time along a certain trajectory determined by (v, ω) and $t_{col}(v, \omega) = \frac{\rho_{min}(v,\omega)}{v}$ is the time until collision with the closest obstacle on the trajectory. It has to be mentioned that the collision calculation is not based on the contact between an object and the robot contour itself but rather between an object and an enlarged safety contour margin around the robot. In [11] it is called speed dependent side clearance SC_v and is used for the translational velocity. This side clearance grows linearly with v and the effect of it is that at higher speeds the free-space areas (i.e. corridors, passages between objects) appear narrower to the robot. $T_{max} = \frac{s_l}{v_{max}}$ is the admissible collision time and it represents the temporal limit above which a trajectory is considered void of obstacles. Its value depends on the maximum translational velocity of the robot v_{max} and the look-ahead distance s_l of used sensors.

The path alignment measure ϑ_{path} for a robot trajectory is defined according to the following expression [15]:

$$\vartheta_{path}(v,\omega) = 1 - \frac{\sum_{i=1}^{N_t}\sum_{j=1}^{N_p} jd_{ij} - D_{min}}{D_{max} - D_{min}}. \quad (5)$$

A trajectory generated by DW, which is a circular arc, is represented by a discrete set of points N_t. Trajectory length L_t is set according to the velocity tuple (v, ω) which characterize current trajectory and the time look-ahead T_{max} beyond which all trajectories are considered clear of

obstacles $L_t(v,\omega) = vT_{max}$. N_p is a set of points on the so called effective path and d_{ij} is the Euclidean distance between the i-th point on the trajectory and j-th point on the effective path. The number of points on both curve is fixed. The limit values $D_{min} = \min_{(v,\omega)} \left\{ \sum_{i=1}^{N_t} \sum_{j=1}^{N_p} j d_{ij} \right\}$ and $D_{max} = \max_{(v,\omega)} \left\{ \sum_{i=1}^{N_t} \sum_{j=1}^{N_p} j d_{ij} \right\}$ are used for normalization along all possible robot trajectories. The index factor j that weights the distance contributions is introduced to penalize more points at the end of the trajectories which define the trajectory deviation from the global path more than the starting points on different trajectories, since they all start from the same robot position.

The so called effective path is the straight line segment connecting the current robot position and a reference point on the path. Its orientation determines the current reference orientation of the robot in relation to the local path configuration affecting the rotational velocity of the robot ω_{ref} and its length determines what the optimal translational velocity v_{ref} should be. Reference point on the path has assigned constant time T_{max} (admissible collision time) as a fixed travelled time from the current robot position to the reference point.

The nominal reference point position is at the point of the second path direction change (path direction changes are multiples of $45°$), as can be seen in Fig. 2. This assumption

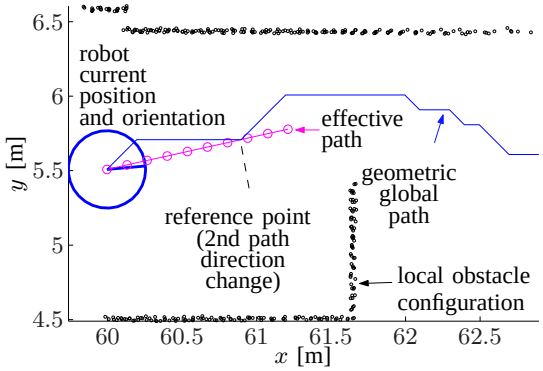

Fig. 2. Determing the nominal reference point position on the global geometric path

is based on the fact that detecting the first path direction change which alters the path direction for $\pm 45°$ starting from the robot position is not enough to determine the tendency of the path change thereafter. The second path direction change then determines whether the path direction changes back to its original direction or continues changing which signifies a stronger curvature change and therefore gives a good local curve tendency information.

III. Adaptations for Moving Obstacles Avoidance

The path planning and control method described above ensures efficient and collision-free motion of a mobile robot in environments with unknown static obstacles. But it can not ensure collision-free motion in environments populated with moving obstacles. Many different approaches to mobile robot motion control in the presence of moving obstacles have been investigated, e.g. see [12]-[14]. While the majority of approaches put the assumption on the shape of moving obstacles, e.g. in [14] all the objects are modelled as ellipses, our approach does not actually consider moving obstacles, but moving cells (MCs) of the grid map. Therefore, the robot is trying to avoid newly occupied cells in its vicinity, which are modelled as empty in the stored occupancy grid map of the environment. The motion of an obstacle can be regarded as the motion of these newly occupied cells. Our algorithm requires velocity vector (v_{mc}, ω_{mc}) and motion heading θ_{mc} for each moving cell. The estimation of these three quantities is not considered in this paper, but it is supposed that they are known.

A. DW adaptation for use with moving obstacles

When a moving obstacle appears in the vicinity of the robot, a set of predicted obstacle trajectories is generated, each trajectory starting from a moving cell. Each MC trajectory is a circular arc defined by velocity vector (v_{mc}, ω_{mc}) which is assumed to be constant for the next n intervals. MC trajectories are recalculated in every sampling instant and are represented by a discrete set of points. The number of points used to represent the MC trajectories is equal to the number of points of DW trajectories calculated for the robot (N_t). The length of a MC trajectory L_{mc} is set equal to $L_{mc} = v_{mc} T_{max}$, where the time constant T_{max} is the same as for DW trajectory. By choosing the same number of points N_t and look ahead time T_{max} for both DW and MC trajectories it is achieved that the i-th point on the MC trajectory has the same assigned time instant t_i as the i-th point on the DW trajectory. This approach greatly simplifies calculation of the time until collision t_{col}, since it is only necessary to check collisions of points on MC and DW trajectories with the same index i, as shown in Fig. 3. The couple of colliding points with the smallest index i determines the time until collision t_{col}, which is then used for calculation of the clearance objective measure $\vartheta_{clear}(v,\omega)$ according to (4). If an obstacle is static then MC trajectories

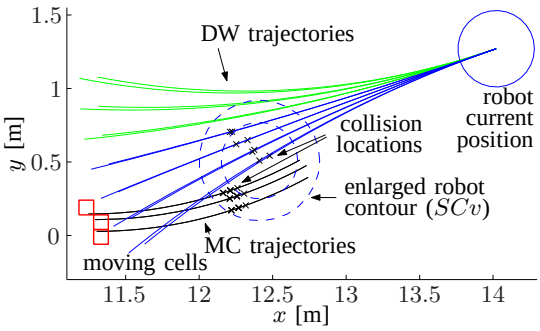

Fig. 3. Collision of MC and DW trajectories

have the zero length and collision calculation turns to be the same as previously defined in section II-B. If the collision is detected between the i-th point on the DW trajectory and

the i-th point on the MC trajectory, then original location of the MC is marked as unoccupied and cell corresponding to the collision point as occupied in the map used by FD* algorithm for the global path recalculation. Therefore, FD* recalculates the path around the expected collision location, and path objective measure $\vartheta_{path}(v, \omega)$ changes the optimal DW trajectory.

B. Path planning algorithm adaptation

The map is continuously changing around moving obstacles, and consequently a number of FD* pointers changes directions in each sampling instant. A situation can occur that FD* generates two mutely distant paths with very similar costs, and that pointer changes cause continuous switching between these two paths. Such a situation is illustrated in Fig. 4, where the controlled robot (right robot in the subfigures) and the other robot (left robot in the subfigures), which presents the moving obstacle, are moving straight ahead towards each other. The FD* algorithm generates pointers on the robot-obstacle connecting line that alternately determine pair of symmetrical paths around the obstacle. The robot would alternately try to follow two different distant paths, which would lead to collision with moving obstacle after a short time.

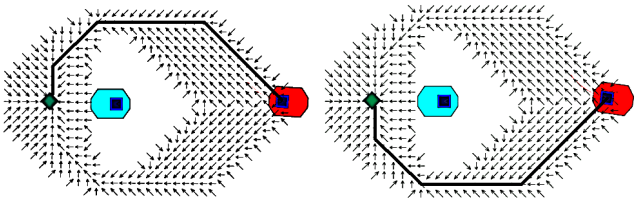

Fig. 4. Path switching

In order to avoid the above described problem we propose the generation of the paths not only from the robot current position cell, but also from the surrounding cells. Total costs of the neighbor paths are simply computed by following the pointers from the neighbor cells to the goal. The paths that are close and geometrically similar to the path from previous sampling instant are preferable, since the robot will not oscillate between distant paths on its traverse. If the path with minimal cost among neighbor paths is distinct from the previous path (e.g. $path1$ with cost c_1 in Fig. 5) it will be chosen only if its cost is lower then the cost of the path with minimal cost among those that are close and geometrically similar to the previous path (e.g. $path2$ with cost c_2 in Fig. 5) decreased by the cost of the transition between starting cells of these two paths (e.g. transition between cells sc_1 and sc_2 in Fig. 5). Otherwise, the robot will choose the path with minimal cost among those that are close and geometrically similar to the previous path ($path2$ in Fig. 5).

The size of the neighborhood around the robot current position that should be considered for choosing a suitable path is determined by the property of DW and FD* integration method explained in the following. Let's assume the worst

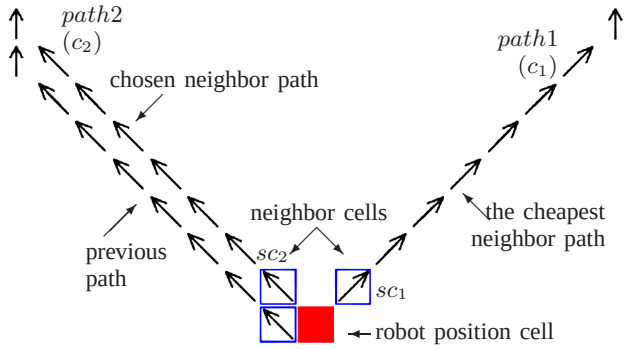

Fig. 5. Path selection

case where the robot is traversing in a free space area. There, the global path is composed of long straight line segments, and there is one path direction change in the robot vicinity and the reference point on the path is at maximal distance $R_{max} = s_l$ from the robot, encouraging maximal translation velocity v_{max}. Robot traverses along sequence of circular and straight line arcs and thus moves away for some distance d from the point of the path direction change. The robot moves away from the global path when the reference point is chosen on the next path segment, i.e. when the robot is within the distance R_{max} to the point of path direction change (see Fig. 6). Maximal departure d is given by:

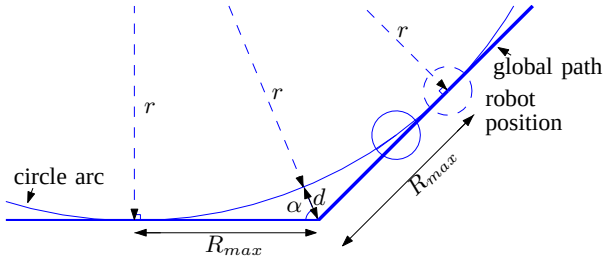

Fig. 6. Maximal robot departure from the path

$$d = R_{max}\frac{1-\sin\alpha}{\cos\alpha} = r\frac{1-\sin\alpha}{\sin\alpha}, \tag{6}$$

where $\alpha = 3{\cdot}45/2$ (since path directions are always multiples of $45°$) and $r = R_{max}\tan\alpha$ is radius of the circular arc. Rotational velocity at which robot traverses along the circular arc is $\omega_r = \frac{v_{max}}{r}$. In reality, robot gradually increases its rotational velocity due to selection of reference point further from the point of the path direction change and thus gradually decreases the radius of the circle arc along which the robot traverses. The smaller is the radius, the smaller is the departure from the global path. Therefore, the distance d given with (6) can be declared as maximal possible robot departure from the global path. Thus, the surrounding cells that are close to the robot within the distance d are used as starting cells for neighbor paths generation.

C. Safety cost mask

Obstacle cells are C-space enlarged to account for robot dimensions (robot mask is two cells in our case). That cells

get prohibitively large safety cost value. But, due to possible robot departure from the global path it can happen that robot comes too close to an obstacle and stops in front of it or even hits it. In order to avoid such situations we propose generation of so called safety cost masks around the C-space enlarged obstacles. Their purpose is to push FD* algorithm to compute global paths as far as possible from the obstacles. The size of the cost mask is also determined by the distance d given with (6) as it is the maximal departure of the robot from the path. Each cell in FD* map within the safety cost mask around the occupied cell gets corresponding safety cost value, which depends on its distance from the occupied cell. The outmost cells get safety cost value for one grater then empty cells out of safety cost mask, and safety cost value of inner cells incrementally increase from outmost cells to the occupied cell. Assigning safety cost values to the cells inside the safety cost mask is illustrated in Fig. 7, where it can be seen that the width of the cost mask is four cells in our case. When the safety cost values of the cells are included in the

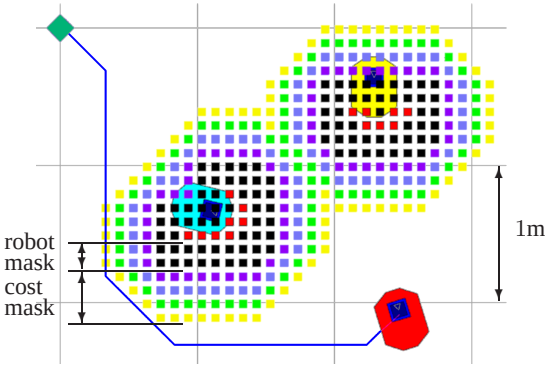

Fig. 7. Safety cost mask

FD* map, the FD* algorithm will compute the global path with the cheapest total cost, which will be out of the safety cost masks around the obstacles if possible, or through the middle of narrower passages if not. The consequence can be that the computed cheapest path is not also the shortest one, as can be seen in Fig. 7. But, this price must be paid in order to achieve collision-free robot motion.

IV. Test Results

The proposed motion control algorithms have been implemented in *Player/Stage* (a free software tool for robot and sensor applications, `www.playerstage.sourceforge.net`). We tested many various situations and our motion control algorithms provided safe and efficient robot motion in all of them. In order to illustrate the functionality of the proposed algorithms, the results of a test are presented in Fig. 8, where two other robots are moving in the vicinity of the controlled robot crossing its path to the goal position. While the left column presents the robot motion obtained with the DW and FD* integration method in the original form [15], the right column presents the robot motion obtained with the same method adapted for robot control in the presence

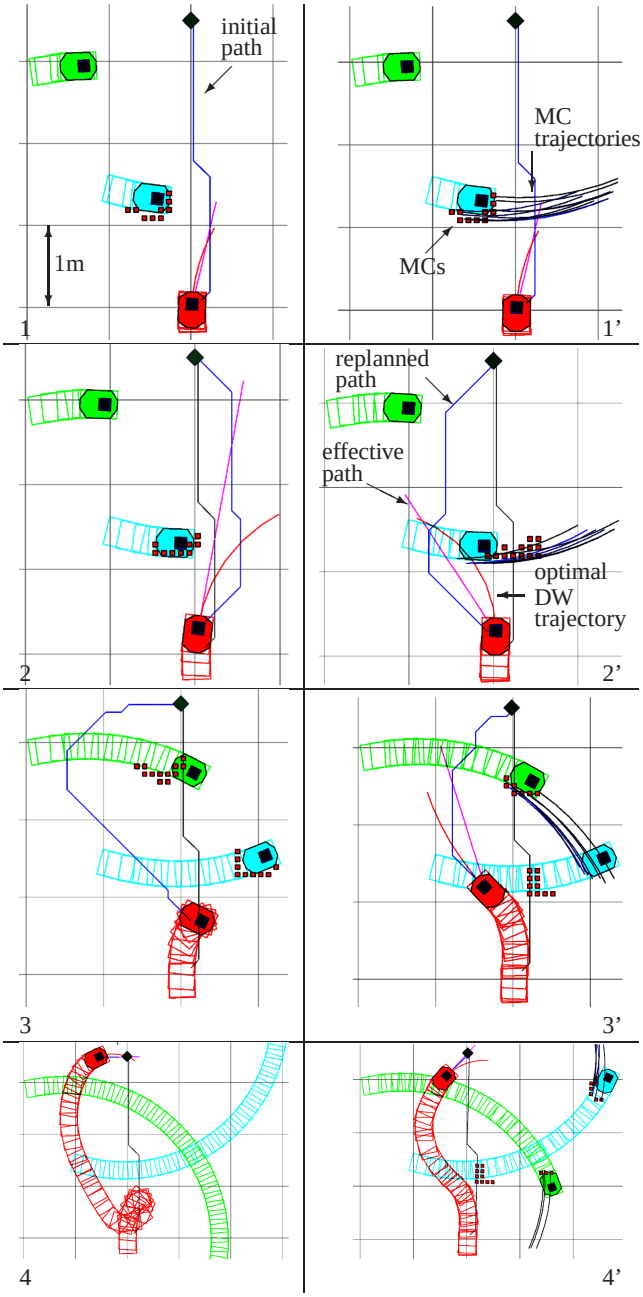

Fig. 8. Motion control comparison

of moving obstacles, as described in section III. Both methods are tested under equal circumstances. A sequence of simulation shots are arranged consecutive one beneath another and indexed. The first snapshot presents the initial configurations, where the global computed path is passing just by the motion heading side of the first moving obstacle, which moves almost perpendicular to it. In the second snapshot, the robot controlled by the original integration method is still trying to avoid the first moving obstacle in the shortest way, i.e. passing by its motion heading side, while the adapted integration method switches the path behind the obstacle. In the third snapshot, the robot controlled by the original method stops and turns in place because the

obstacle closeness forbids robot turning left, while the robot controlled by the adapted integration method continues smooth motion. It is visible that the replanned path to the goal is much shorter for the adapted integration method, what is the result of the second obstacle motion prediction. While the first three snapshots show the configurations with both control method in the same time instants, the last, forth, snapshot shows the configurations when robot is just in front of the goal position. Obviously, the adapted method produces much smoother and shorter path of the robot from the start to the goal position. From the velocity profiles for the original method shown in Fig. 9 and adapted method shown in Fig. 10 it is clearly seen that the adopted method produces much smoother motion and that the robot reaches the goal twice faster (19 seconds vs. 38 seconds). Results of

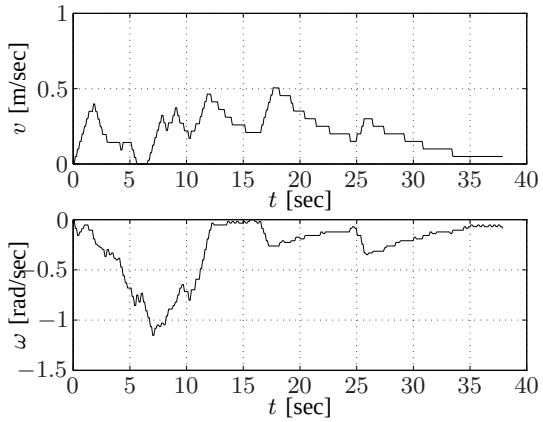

Fig. 9. Velocity profiles of the original integration method

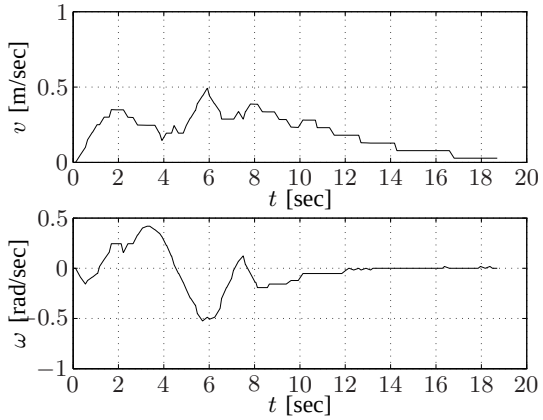

Fig. 10. Velocity profiles of adapted integration method

a number of simulation and real experiments can be seen at `http://act.rasip.fer.hr/groups_amor_path.php`.

V. Conclusion and Future Work

In this paper our original integrated approach to real-time mobile robot control, presented in [15], is extended with capability of collision-free robot motion in the presence of moving obstacles. The approach is based on the integration of the focused D* graph search algorithm for path planning and dynamic window algorithm for generation of admissible robot trajectories around the global path. The motion of an obstacle is regarded as the motion of the occupied moving cells in a grid map. The predicted trajectory of each moving cell is used for the collision calculation with the possible robot trajectories. Our algorithm requires velocity vector and motion heading for each moving cell. The estimation and prediction of these three quantities will be elaborated further.

VI. Acknowledgments

This research has been supported by the Ministry of Science, Education and Sports of the Republic of Croatia under grant No. $036-0363078-3018$.

References

[1] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Dodrecht, Netherlands, 1991.

[2] O. Brock and O. Khatib, "High-speed navigation using the Global Dynamic Window Approach," *ICRA'99, IEEE International Conference on Robotics and Automation*, 1999.

[3] J. Stuart and R. Norvig and P. Norvig, *Artificial Intelligence - A Modern Approach*, Prentice Hall Series, 1995.

[4] A. Stentz, "Optimal and Efficient Path Planning for Partially - Known Environments," *ICRA'94, IEEE International Conference on Robotics and Automation*, 1994.

[5] A. Stentz, "The Focussed D* Algorithm for Real-Time Replanning," *International Joint Conference on Artificial Intelligence*, 1995.

[6] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *The International Journal of Robotics Research*, 5(1), pp. 90–98, 1986.

[7] J. Borenstein and Y. Koren, "The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots," *IEEE Transactions on Robotics and Automation*, 7(3), pp. 278-288, 1991.

[8] J. Minguez and L. Montano, "Nearness Diagram Navigation (ND): A New Real Time Collision Avoidance Approach," *IROS'00, IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000.

[9] R. Simmons, "The Curvature-Velocity Method for Local Obstacle Avoidance," *ICRA'96, IEEE International Conference on Robotics and Automation*, 1996.

[10] R. Simmons, "The Lane-Curvature Method for Local Obstacle Avoidance," *IROS'98, IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.

[11] D. Fox and W. Burgard and S. Thrun, "The Dynamic Window Approach to Collision Avoidance," *IEEE Robotics & Automation Magazine*, 4(1), pp. 23-33, 1997.

[12] R. Kimmel and N. Kiryati and A. M. Bruckstein, "Multivalued Distance Maps for Motion Planning on Surfaces with Moving Obstacles," *IEEE Transactions on Robotics and Automation*, 14(3), pp. 427-436, 1998.

[13] A. Chakravarthy and D. Ghose, "Obstacle Avoidance in a Dynamic Environment: A Collision Cone Approach," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 28(5), pp. 562-574, 1998.

[14] K.-S. Hwang and M.-Y. Ju, "A Propagating Interface Model Strategy for Global Trajectory Planning Among Moving Obstacles," *IEEE Transactions on Industrial Electronics*, 49(6), pp. 1313-1322, 2002.

[15] M. Seder and K. Maček and I. Petrović, "An integrated approach to real-time mobile robot control in partially known indoor environments," *IECON'05, Proceedings of the 31st Annual Conference of the IEEE Industrial Electronics Society*, pp. 1785-1790, 2005.

[16] C. Stachniss and W. Burgard, "An Integrated Approach to Goal-directed Obstacle Avoidance under Dynamic Constraints for Dynamic Environments," *IROS'02, IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[17] K.O. Arras and J. Persson and N. Tomatis and R. Siegwart, "Real-Time Obstacle Avoidance for Polygonal Robots with a Reduced Dynamic Window," *ICRA'02, IEEE International Conference on Robotics and Automation*, 2002.