Advanced Dynamic Window based Navigation Approach using Model Predictive Control

Domokos Kiss and Gábor Tevesz
Department of Automation and Applied Informatics,
Budapest University of Technology and Economics,
H-1117 Budapest, Magyar Tudósok körútja 2., Hungary
{domokos.kiss, tevesz}@aut.bme.hu

***Abstract*—A well-known reactive motion planning technique, the dynamic window approach (DWA) provides an elegant way to navigate safely in the presence of obstacles, also taking dynamic properties of the robot into account. Most of the DWA-based methods have the same limitation, namely they use an objective function consisting of weighted terms. Different situations require different weights, however, there is no algorithm for choosing them. This paper presents a global dynamic window-based method for motion planning using model predictive control and having no weighted objective function. Former DWA-based methods take dynamic limitations of the robot by acceleration constraints into account. In contrast with that, the proposed approach utilizes a dynamic motion model of the robot.**

I. Introduction

This paper focuses on the motion planning of a mobile robot moving in the plane, in the case of such obstacle distributions where narrow crossings are unavoidable. The majority of planning approaches consider the robot as a single point or assume that it is circular. If there are wide free areas between the robot and its target, the bounding circle is a good approximation but in many cases it is too large to pass through narrowings.

An elegant real-time planning strategy, the *dynamic window approach* (DWA) is proposed in [1]. Some extensions of this approach have also been published in the past years. These methods take the dynamic behavior of the robot into account by acceleration bounds and in most cases the vehicle shape is not taken into account. The contribution of this paper is a global dynamic window-based navigation method, considering the dynamic model of the robot and vehicle shape as well. The proposed method can be considered as a *feedback motion planning* approach [2]. This means that planning and execution are not separate stages. The planning result is a sequence of motion commands, which are generated in such a consecutive way that the motion can be executed on-line during planning. This property is useful because unexpected changes in the robot state or in the environment can be handled. However, environment changes cause other complications, hence we assume a static environment in this paper for simplicity.

The paper is organized as follows. In Section II, we take a short survey of dynamic window-based navigation approaches and summarize the contribution of the presented approach. In Section III the robot models are introduced the method can be applied to. In Section IV, it is shown how the shape of the robot is taken into account. Section V is about the navigation function that plays an important role in the proposed method. In Section VI, we describe the model predictive algorithm and in Section VII we investigate the applicability to different robot kinematics. Section VIII summarizes the paper and gives directions of future work.

II. Dynamic Window Based Navigation Methods

The dynamic window approach of robot navigation [1] differs from former approaches – which use potential fields and virtual forces [3], [4] – in that it assumes a velocity motion model for robots, i.e. velocities are considered as actuating variables. It deals with robots having non-holonolmic kinematic constraints, whose trajectories can be approximated by a sequence of circular arcs. A circular path segment can be characterized by a velocity pair (v, ω) which consists of the translational velocity v and the angular velocity ω of the robot. This approach takes the dynamics of the robot into account by reducing the search space to velocities reachable in a short time interval. This subset of the velocity space is called the *dynamic window*. In addition, only those velocities are considered that are safe with respect to obstacles (admissible velocities). To choose from the set of admissible velocities an objective function is evaluated and maximized. This objective function is a weighted sum of three terms: $\mathrm{heading}(v, \omega)$, which is a measure of going into the direction of the goal, $\mathrm{dist}(v, \omega)$, which is the smallest distance to the next obstacle along the circular path segment belonging to (v, ω) and $\mathrm{velocity}(v, \omega)$, simply the projection of (v, ω) on the translational velocity v (to favor high motion speeds).

Experimental results presented in [1] show that the dynamic window approach to collision avoidance yields a fast and safe robot motion. However, since the DWA and the above mentioned other methods are based on local decisions without taking connectivity information of free space into account, the robot can get trapped in local minima situations (i.e. it can stop far from the goal point) or enter a limit-cycle that prevents reaching the goal. Another problem of the DWA is that different situations require different weighting of the objective function terms to ensure successful motion but there is no algorithm for choosing the weights.

A modified approach, called the Global DWA [5] extends the original method to the case of holonomic robots and addresses the problem of local minima by taking free space connectivity information into account. This is obtained by introducing a navigation function (NF), which is a local minima-free function defined on the discretized configuration space, having a unique minimum at the goal. New terms are added to the objective function to favor NF descent along the robot path.

The problem of local minima is eliminated in many cases through the global distance information represented by the NF terms, but not at all times. It is shown in [6] that limit-cycles can evolve if the velocity term outweighs the NF terms. They reformulate the dynamic window approach as a model predictive control (MPC) problem (also referred to as receding horizon control, RHC). They assume a holonomic robot model and choose acceleration as control signal. It is shown that for a given set of controls the system is stable in the Lyapunov sense (but not asymptotically). However, this property is not enough to ensure convergence, since the case of zero velocity far from the goal cannot be excluded. This is prevented by a timeout condition in their proposed algorithm.

Another improvement of the original DWA is presented in [7], called I-DWA. This variant adds convergence improvements to the original method by pre-calculating an ideal control action that would make the robot converge to the goal if no obstacles were present. The objective function is similar to the one in [1] and favors control actions close to the ideal ones. Because no global information about obstacles is taken into account, the convergence is actually not ensured.

The above mentioned approaches have the same property of assuming a point-like or circle-shaped robot. A circle-shaped robot can also be reduced to a point if the obstacles are dilated by the robot radius. This is on one hand a very effective and simple assumption if the robot actually has a circular shape or the bounding circle can be used for representing the robot in the collision detection phase of the algorithm. On the other hand, if the shape considerably differs from a circle and there are narrow passages between obstacles, this assumption fails and the algorithms report that no collision-free path exists.

To overcome this limitation, some approaches were also proposed that take the vehicle shape into account. In [8] robot shape is handled by using precalculated lookup tables that contain local data about collision risk depending on the current velocity and obstacle configuration. This approach is memory intensive but a real-time local obstacle avoidance can be obtained. The approach in [9] solves the task analytically for polygonal robots without the use of lookup tables. These two approaches assume that obstacles are represented by points (e.g. obtained by laser range sensors). This representation is efficient only in the case of local navigation taking only the neighborhood of the robot into consideration, because in a global case too many obstacle points would have to be stored and handled. The method presented in [10] uses the same obstacle model. Unlike others, it delivers an elegant and analytical solution by "wrapping" former existing obstacle avoidance approaches (e.g. potential field methods) in a framework that allows consideration of robot shape and kinematic and dynamic constraints, while it still has the limitation of locality. Since only local methods can be "wrapped", their inherent convergence problems are not solved.

The authors of the present paper have also proposed a DWA-based approach for mobile robot navigation, the Global Dynamic Window Approach using Receding Horizon Control (GDWA/RHC) [11]. It works in velocity space and assumes a non-holonomic and circular robot model, similar to [1] and [7]. Global information is taken into account by a local-minima-free navigation function which serves as a basis for optimization. The objective function has no weighted terms and the control law looks like as follows:

$$\mathbf{u}(\cdot) = \arg\min_{\mathbf{u}(\cdot)} \mathrm{NF}\left(r_x(t+T), r_y(t+T)\right) \tag{1}$$

where $\mathrm{NF} : \mathbb{R}^2 \to \mathbb{R}$ is the navigation function defined on the configuration space of the robot, $r_x(t+T)$ and $r_y(t+T)$ are the predicted robot position coordinates at the end of a time horizon, which can be derived from the motion equation of the robot.

The contribution of this paper is an improved version of the GDWA/RHC method. The former method, presented in [11], assumed circular shape and considered dynamics as acceleration bounds. The method presented here can be applied to polygonal robot models. In addition to that, the outputs of the planning process (i.e. the actuating variables) are wheel traction forces instead of velocities. As mentioned above, vehicle shape, kinematics and dynamics have already been considered in [10] as well, but the framework presented there can only be applied to local planning techniques having the common disadvantage of being sensitive to cyclical motions and trap situations. These situations are avoided in our method using a global navigation function similar to the one in [11]. The configuration space and the navigation function in this paper are extended to three dimensions in order to incorporate the robot orientation in the optimization process. This advanced method can be applied to robot models with different kinematics. In case of kinematic constraints (e.g. differential drive) a prescribed target position can be reached. This is similar to [10] and [11] but for polygonal robot models and avoiding trap situations at the same time. For robots without kinematic constraints the result is even better: a prescribed position *and* orientation can be reached while still considering the shape and dynamics of the robot.

III. Robot Models

Before going into details of the navigation approach, we present the robot models the method can be applied to. If considering only planar motion, the position and orientation of a robot in the global coordinate system (also referred to as configuration $\mathbf{q}$) can be described by three independent coordinates $\mathbf{q} = [x, y, \varphi]^T$. Two models are considered, which can be derived from a common general model (see Fig. 1). On one hand, we look at a holonomic model having three omnidirectional wheels, on the other hand we investigate differential drive which has a non-holonomic wheel arrangement. Holonomic robots have at least as many degrees of freedom (e.g. number of independently driven wheels) as the dimension of $\mathbf{q}$, in other words they are fully actuated. In contrast with that, non-holonomic robots are under-actuated, since they have less degrees of freedom than it would be necessary to control all position and orientation coordinates independently.

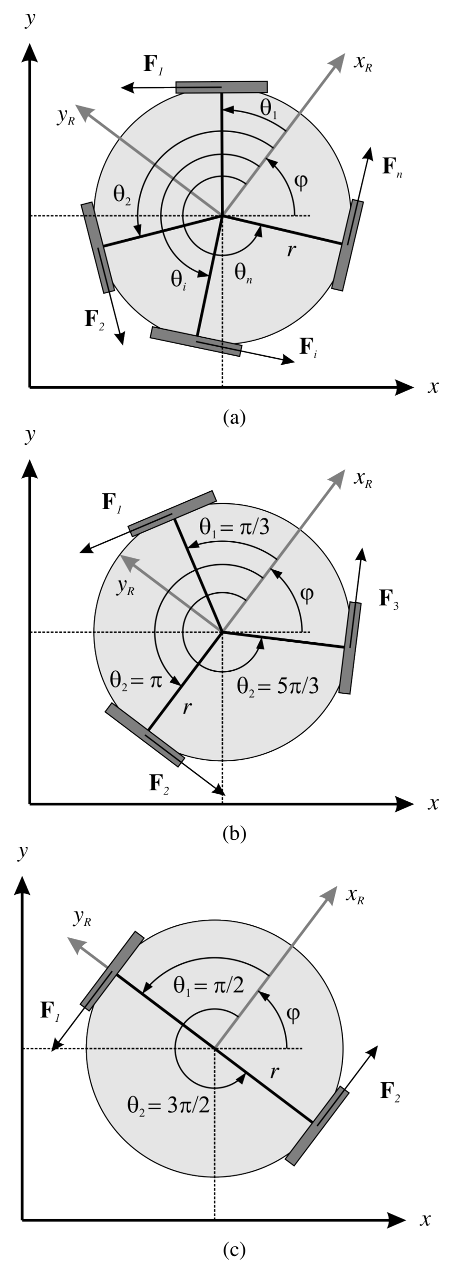

Fig. 1: Robot models. (a) General wheel arrangement, (b) 3-wheeled symmetric omnidirectional drive, (c) Differential drive

A. Omnidirectional drive

Omnidirectional wheels are built in such a way that they provide traction in the direction normal to the motor axis while they can slide frictionless in the motor axis direction, thanks to small rollers along the perimeter of the wheel. A general omnidirectional robot model having n wheels is depicted in Fig. 1a. The angles $\theta_1, \theta_2, \ldots, \theta_n$ of the motor axes are given relative to the x-axis of the robot coordinate frame. Each motor provides a traction force vector $\mathbf{F}_i$ which is the torque of the motor multiplied by the wheel radius. Note that in a real robot also damping and friction forces arise but these are omitted here for simplicity. The wheel traction forces add up to a translational force and a rotational torque that move the robot. The relationship between wheel traction forces and robot acceleration is derived in [12]:

$$\begin{bmatrix} {}^R a_x \\ {}^R a_y \\ \dot{\omega} \end{bmatrix} = \frac{1}{M} \begin{bmatrix} -\sin\theta_1 & -\sin\theta_2 & \cdots & -\sin\theta_n \\ \cos\theta_1 & \cos\theta_2 & \cdots & \cos\theta_n \\ \frac{Mr}{I} & \frac{Mr}{I} & \cdots & \frac{Mr}{I} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}, \quad (2)$$

which can be written as

$${}^R\mathbf{a} = \mathbf{C} \cdot \mathbf{f}, \quad (3)$$

where ${}^R a_x$ and ${}^R a_y$ are the translational acceleration coordinates of the robot reference point expressed in the robot coordinate frame, $\dot{\omega}$ is the angular acceleration of the robot, $\mathbf{f} = [f_1, f_2, \ldots, f_n]^T$ is the vector consisting of wheel traction force magnitudes ($f_i = |\mathbf{F}_i|$). The magnitudes can be positive or negative, depending on the direction of rotation of the motor. The positive directions are shown in Fig. 1a. M denotes the mass, I the moment of inertia of the robot related to the vertical axis through the robot reference point, which is assumed to be the same as the center of mass. The distance of wheels from the robot reference point is denoted by r. $\mathbf{C}$ is called the *force coupling matrix*. Note that it is assumed that no wheel slippage occurs. The acceleration vector can be expressed in world coordinates as well:

$$\mathbf{a} = \mathbf{R}(\varphi) \cdot {}^R\mathbf{a} = \mathbf{R}(\varphi) \cdot \mathbf{C} \cdot \mathbf{f}, \quad (4)$$

where $\mathbf{a} = [a_x, a_y, \dot{\omega}]^T = \ddot{\mathbf{q}}$ and $\mathbf{R}(\varphi)$ is a rotation matrix depending on the actual robot orientation:

$$\mathbf{R}(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5)$$

We will write $\mathbf{R}(\varphi)$ as $\mathbf{R}$ in the sequel, but still keeping in mind that it depends on the actual orientation.

Our goal is to control the robot to a given goal configuration $\mathbf{q}_g$ and to let it stop there, i.e. $\dot{\mathbf{q}} = 0$. For that reason we choose the state vector of the system to $\mathbf{x} = \left[\mathbf{q}^T, \dot{\mathbf{q}}^T\right]^T = [x, y, \varphi, v_x, v_y, \omega]^T$, where $\dot{\mathbf{q}} = [v_x, v_y, \omega]^T$ denotes the velocity vector of the robot. The control vector is $\mathbf{u} = \mathbf{f}$ because our actuating variables are the motor torques. The state equation has the following form:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & \mathbf{I} \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ \mathbf{RC} \end{bmatrix} \mathbf{u}. \quad (6)$$

As it will be shown later, we need to solve this equation in the prediction stage of the navigation algorithm. In order to obtain a computationally tractable iterative solution, we assume discrete time and use the following approximation:

$$\dot{\mathbf{x}} \approx \frac{1}{T_s} \left(\mathbf{x}_{t+T_s} - \mathbf{x}_t\right), \quad (7)$$

where T_s denotes the sampling period, $\mathbf{x}_t$ the current and $\mathbf{x}_{t+T_s}$ the next state. After realigning (6) using (7) we get the state transition equation

$$\mathbf{x}_{t+T_s} = \begin{bmatrix} \mathbf{I} & T_s\mathbf{I} \\ 0 & \mathbf{I} \end{bmatrix} \mathbf{x}_t + T_s \begin{bmatrix} 0 \\ \mathbf{R}_t\mathbf{C} \end{bmatrix} \mathbf{u}_t, \quad (8)$$

where $\mathbf{R}_t$ is a short form of $\mathbf{R}(\varphi_t)$.

To obtain a fully actuated robot at least three omnidirectional wheels are needed. A reasonable wheel distribution is shown in Fig. 1b, where the angles of motor axes are chosen to $\theta_1 = \pi/3$, $\theta_2 = \pi$ and $\theta_3 = 5\pi/3$. In this case the force coupling matrix is

$$\mathbf{C} = \frac{1}{M}\begin{bmatrix} -\frac{\sqrt{3}}{2} & 0 & \frac{\sqrt{3}}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \\ \frac{Mr}{I} & \frac{Mr}{I} & \frac{Mr}{I} \end{bmatrix}. \tag{9}$$

B. Differential drive

Differential drive can be considered as a special case of the general wheel arrangement with $n = 2$ and motor axis angles $\theta_1 = \pi/2$ and $\theta_2 = 3\pi/2$ (see Fig. 1c). In this arrangement the wheel traction forces are parallel with the x-axis of the robot coordinate frame, hence it is impossible to obtain a y-directed acceleration using the actuating forces. This can be seen after substituting θ_1 and θ_2 into (2):

$$\begin{bmatrix} {}^R a_x \\ {}^R a_y \\ \dot{\omega} \end{bmatrix} = \frac{1}{M}\begin{bmatrix} -1 & 1 \\ 0 & 0 \\ \frac{Mr}{I} & \frac{Mr}{I} \end{bmatrix}\begin{bmatrix} f_1 \\ f_2 \end{bmatrix}. \tag{10}$$

Note that on differentially driven robots regular wheels are used without rollers on their perimeter. This means on the one hand that the robot cannot slide sideways frictionless (if we assume no wheel slippage). On the other hand, a y-directed acceleration can exist thanks to static friction. In case of curved motion this friction force is responsible for the centripetal acceleration, which is always parallel with the y-axis of the robot reference frame. It can be seen that this robot model is under-actuated because it has only two degrees of freedom (dimension of $\mathbf{f}$) thus the three variables of position and orientation cannot be independent. The dependency can be characterized by the above mentioned kinematic constraint regarding the y-directed acceleration:

$${}^R a_y = a_c = {}^R v_x \omega. \tag{11}$$

Note that this constraint is equivalent with the fact that the instantaneous velocity of the robot is always parallel with the x-axis of the robot coordinate frame. In order to obtain the motion equation of differential drive, this constraint has to be added to (10):

$$\begin{bmatrix} {}^R a_x \\ {}^R a_y \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 \\ {}^R v_x \omega \\ 0 \end{bmatrix} + \frac{1}{M}\begin{bmatrix} -1 & 1 \\ 0 & 0 \\ \frac{Mr}{I} & \frac{Mr}{I} \end{bmatrix}\begin{bmatrix} f_1 \\ f_2 \end{bmatrix}. \tag{12}$$

Let us write this as

$${}^R\mathbf{a} = \mathbf{\Omega} \cdot {}^R\mathbf{v} + \mathbf{C} \cdot \mathbf{f}, \tag{13}$$

where ${}^R\mathbf{v} = \left[{}^R v_x, {}^R v_y, \omega\right]^T$ is the vector of instantaneous robot velocities and

$$\mathbf{\Omega} = \begin{bmatrix} 0 & 0 & 0 \\ \omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{14}$$

Let (13) be expressed in world coordinates:

$$\begin{aligned} \mathbf{a} &= \mathbf{R} \cdot {}^R\mathbf{a} = \\ &= \mathbf{R\Omega} \cdot {}^R\mathbf{v} + \mathbf{RC} \cdot \mathbf{f} = \\ &= \mathbf{R\Omega R}^T \cdot \mathbf{v} + \mathbf{RC} \cdot \mathbf{f} \end{aligned} \tag{15}$$

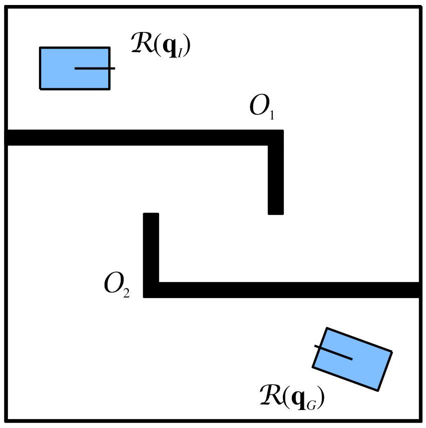

Fig. 2: Polygonal robot and obstacles. $\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2$ is the obstacle region, $\mathcal{R}(\mathbf{q}_I)$ is the robot in the initial configuration, $\mathcal{R}(\mathbf{q}_G)$ is the robot in the goal configuration.

Using this, the state equation of the system becomes

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & \mathbf{I} \\ 0 & \mathbf{R\Omega R}^T \end{bmatrix}\mathbf{x} + \begin{bmatrix} 0 \\ \mathbf{RC} \end{bmatrix}\mathbf{u} \tag{16}$$

and after applying the discrete-time approximation the following state transition equation is obtained:

$$\begin{aligned} \mathbf{x}_{t+T_s} &= \begin{bmatrix} \mathbf{I} & T_s\mathbf{I} \\ 0 & \mathbf{I} + T_s\mathbf{R}_t\mathbf{\Omega}_t\mathbf{R}_t^T \end{bmatrix}\mathbf{x}_t + \\ &+ T_s\begin{bmatrix} 0 \\ \mathbf{R}_t\mathbf{C} \end{bmatrix}\mathbf{u}_t, \end{aligned} \tag{17}$$

where $\mathbf{R}_t$ and $\mathbf{\Omega}_t$ stand for $\mathbf{R}(\varphi_t)$ and $\mathbf{\Omega}(\omega_t)$, respectively.

IV. Configuration Space

In order to be able to take the shape of the robot into account, the navigation problem is transformed into the configuration space (also referred to as *C-space*). We recall some considerations regarding the C-space of a planar robot with polygonal shape from [2]. The configuration space $\mathcal{C}$ of a 2-dimensional robot that can translate and rotate in the workspace $\mathcal{W} = \mathbb{R}^2$ is the manifold $\mathbb{R}^2 \times \mathbb{S}^1$. The obstacle region $\mathcal{O} \subset \mathcal{W}$ and the robot $\mathcal{R} \subset \mathcal{W}$ are given by a polygonal model (an example is depicted in Fig. 2). The configuration space obstacle region $\mathcal{C}_{obs} \subset \mathcal{C}$ is defined as

$$\mathcal{C}_{obs} = \{\mathbf{q} \in \mathcal{C} \mid \mathcal{R}(\mathbf{q}) \cap \mathcal{O} \neq \emptyset\} \tag{18}$$

which consists of all configurations $\mathbf{q}$ for which the transformed robot $\mathcal{R}(\mathbf{q})$ is in collision with the obstacle region $\mathcal{O}$.

An algorithm for collision detection of convex polygonal shapes is described in [2, pp. 164-166]. In the case of nonconvex obstacle and robot shapes these can be considered as the union of their convex parts. A useful method for minimal convex decomposition of simple polygons is proposed in [13].

To obtain an explicit model of $\mathcal{C}_{obs}$, we discretize the configuration space with resolutions k_x, k_y and k_φ, which are positive integers. Let

$$\begin{aligned} \Delta\mathbf{q}_1 &= [x_{max}/k_x, \quad 0, \quad 0]^T \\ \Delta\mathbf{q}_2 &= [0, \quad y_{max}/k_y, \quad 0]^T \\ \Delta\mathbf{q}_3 &= [0, \quad 0, \quad \pi/k_\varphi]^T \end{aligned} \tag{19}$$

and let a grid point $\mathbf{q}'$ be expressed as

$$\mathbf{q}'(i,j,k) = i\Delta\mathbf{q}_1 + j\Delta\mathbf{q}_2 + k\Delta\mathbf{q}_3, \tag{20}$$

where $i \in \{0,\ldots,k_x\}$, $j \in \{0,\ldots,k_y\}$ and $k \in \{-k_\varphi,\ldots,k_\varphi\}$. Every grid point $\mathbf{q}'(i,j,k)$ is tested for collision and the C-space obstacle region is redefined as

$$\mathcal{C}_{obs} = \{\mathbf{q} \in \mathcal{C} \mid \mathcal{R}(f(\mathbf{q})) \cap \mathcal{O} \neq \emptyset\}, \tag{21}$$

where $f(\mathbf{q})$ is a function that returns the grid point $\mathbf{q}'(i,j,k)$ that lies closest to $\mathbf{q}$. The indices i, j and k of the closest grid point are determined as follows:

$$\begin{aligned} i &= \lfloor x \cdot k_x / x_{max} + 0.5 \rfloor, \\ j &= \lfloor y \cdot k_y / y_{max} + 0.5 \rfloor, \\ k &= \lfloor \varphi \cdot k_\varphi / \pi + 0.5 \rfloor. \end{aligned} \tag{22}$$

where $\lfloor \ldots \rfloor$ denotes the integer part of a real number. Using this definition, the C-space obstacle region is built up of small “bricks” located at the occupied grid points.

This representation allows the motion planning problem of a planar polygonal robot to be expressed as a planning problem of a single translating point in the (3-dimensional) configuration space. Note that at higher resolutions and in the case of complicated environment or robot shape the process of obtaining an explicit model for the configuration space is quite time consuming.

V. Navigation Function

As mentioned above, the task is to find a collision-free path in the C-space between initial and target configurations. The convergence can only be ensured if global information about free and occupied areas are taken into account. Similar to [1] and [5], we utilize a navigation function to achieve this. A navigation function (NF) is a real-valued function defined on the unoccupied part of the configuration space $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$, which has exactly one minimum, namely at the goal configuration $\mathbf{q}_G$.

To define such a function, it is convenient to use the discretized configuration space model (20). Starting from $\mathbf{q}'_G = f(\mathbf{q}_G)$, the NF values at every reachable $\mathbf{q}' \in \mathcal{C}_{free}$ are obtained by a wavefront propagation algorithm [2]. This algorithm performs a breadth-first-like numerical search and labels the grid points with ascending function values starting from $\mathrm{NF}(\mathbf{q}'_G) = 0$.

For the situation depicted in Fig. 2, a cross section of the discrete navigation function at $\varphi = \varphi_G$ is shown in Fig. 3. The black area represents occupied configurations and the “$\times$” mark shows the goal configuration, where the wavefront propagation was started. Greater function values are represented by lighter colors.

At this point, NF values have been assigned to grid points. As any configuration $\mathbf{q} \in \mathcal{C}_{free}$ is allowed for the robot, NF values in all these points should be determined. We use trilinear interpolation between grid points. It has the convenient property of having no local minima if neighboring points do not have equal values, which is ensured by the wavefront propagation algorithm.

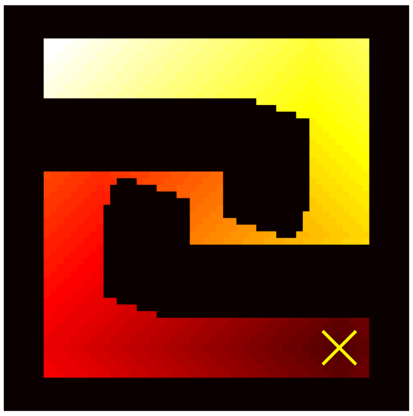
Fig. 3: Cross-section of $\mathrm{NF}(\mathbf{q}')$ at $\varphi = \varphi_G$

VI. Model Predictive Navigation Algorithm

The obstacle avoidance problem is considered as a constrained optimization problem over the control space of the robot, similar to (1). The objective function which has to be minimized is the navigation function itself, constructed as described in Section V.

A control $\mathbf{u}(\cdot)$ has to be determined at every time instant t which minimizes the NF value at the end of a time horizon $[t, t+T]$:

$$\mathbf{u}(\cdot) = \arg\min_{\mathbf{u}(\cdot)} \mathrm{NF}\left(\mathbf{q}(t+T)\right) \tag{23}$$

where $\mathbf{q}(t+T)$ can be derived from the motion equation. At t the control $\mathbf{u}(t)$ has to be applied to the robot model. Safety is ensured by a constraint of admissible controls. A control $\mathbf{u}(\cdot)$ is admissible, if

$$\begin{gathered} \mathbf{q}(\tau) \notin \mathcal{C}_{obs}, \\ \forall \tau \in [t, t+T] \cup [t+T, t+T+T_{brake}] \end{gathered} \tag{24}$$

We assume that a maximal braking control $\mathbf{u}^*(\cdot)$ is applied in the time interval $[t+T, t+T+T_{brake}]$ where T_{brake} is the time needed to halt the robot from the velocity at $t+T$. This means that only those controls are admissible that do not cause a collision inside the horizon and allow the robot to stop safely beyond the horizon.

We use variable horizon length, where T depends on the current velocity. It is chosen to $T \geq T_{brake}$ in order to ensure the possibility of stopping inside the horizon. This allows a smooth halt at the target. In contrast with that, a too short horizon would cause an “overshoot” or oscillation while approaching the target.

During the optimization process we have to choose a control value at every discrete time instant that satisfies (24) and velocity and force constraints, while minimizing (23). To do this, the control space is also quantized, which results in a countable set of wheel forces. The dynamic window is the set of all possible discrete wheel force configurations. We take each admissible control value from the dynamic window and calculate its effect to the robot model for the duration of $T = hT_s$, $h \in \mathbb{N}$ using (8) or (17). The control resulting in the smallest NF value at $t + hT_s$ will be chosen and applied to the robot model in the time interval $[t, t+T_s]$.

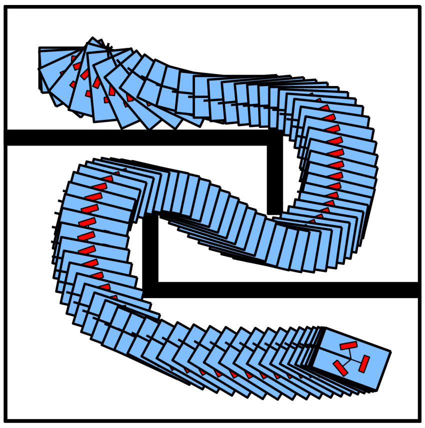

Fig. 4: Three-wheeled omnidirectional robot

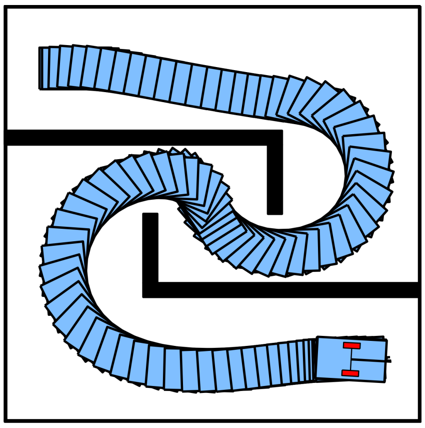

Fig. 5: Differential drive reaching the target position while omitting the desired orientation

VII. Application to Different Robot Models

The method has been tested in simulations for both presented robot models. The trajectory of a rectangle-shaped robot having three omnidirectional wheels is depicted in Fig. 4. It can be seen that the narrow crossing caused no problem and the target position and orientation (see Fig. 2) have been reached successfully.

In Fig. 5 a robot model with differential drive is depicted. In this case the navigation function is modified in order to let it be independent of orientation. This means that $\mathrm{NF}(x, y, \varphi) = \mathrm{NF}(x, y)$, for every φ. In other words, only the target position is prescribed without orientation. As it can be seen, the goal position is reached, while omitting the target orientation. However, since a robot with differential drive is able to turn in one place, the process can be divided into two stages. At first the robot travels to the target position and after that it is commanded to turn to the desired orientation. This can be done under the assumption that the target position is given such that obstacles are not closer than the robot collision radius.

Note that this method causes the robot to graze the obstacles (similar to other planning methods that look for shortest paths). This can be seen e.g. in Fig. 4. However, this inconvenience can be avoided by virtually dilating the obstacles by a "safe zone".

VIII. Conclusions and Future Work

A dynamic window-based navigation approach was presented for polygonal robots having omnidirectional or differential drive. The method utilizes the idea of model predictive control and a navigation function serves as optimization objective, defined on the configuration space. Dynamics of the robot is taken into account by a dynamic model, using wheel traction forces as actuating variables. The method works fine for holonomic robots, and also can be applied to robots with constrained kinematics under the assumption that turning is possible at the target position. Further effort has to be taken in order to eliminate this limitation regarding robot models having differential constraints. Future work includes also improvements on the calculation of the navigation function. The wave propagation algorithm in the current form is quite time-consuming for wider working areas and higher grid resolutions. If the NF computation could be accelerated, the presented planning algorithm would also be applicable in changing environments.

Acknowledgments

The work in the paper has been developed in the framework of the project "Talent care and cultivation in the scientific workshops of BME". This project is supported by the grant TÁMOP - 4.2.2.B-10/1–2010-0009

References

[1] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, pp. 23–33, 1997.

[2] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[3] O. Khatib, "Real-time obstacle avoidance for robot manipulator and mobile robots," *The International Journal of Robotics Research*, vol. 5, pp. 90–98, 1986.

[4] J. Borenstein and Y. Koren, "The vector field histogram - fast obstacle avoidance for mobile robots," *IEEE Trans. Robot. Autom.*, vol. 7, pp. 278–288, 1991.

[5] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Detroit, MI, 1999, pp. 341–346.

[6] P. Ogren and N. Leonard, "A convergent dynamic window approach to obstacle avoidance," *IEEE Trans. Robot.*, vol. 21, pp. 188–195, 2005.

[7] H. Berti, A. Sappa, and O. Agamennoni, "Improved dynamic window approach by using Lyapunov stability criteria," *Latin American Applied Research*, vol. 38, pp. 289–298, 2008.

[8] C. Schegel, "Fast local obstacle avoidance under kinematic and dynamic constraints for a mobile robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Victoria, B.C., Canada, 1998, pp. 594–599.

[9] K. Arras, J. Persson, N. Tomatis, and R. Siegwart, "Real-time obstacle avoidance for polygonal robots with a reduced dynamic window," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, DC, 2002, pp. 3050–3055.

[10] J. Minguez and L. Montano, "Extending collision avoidance methods to consider the vehicle shape, kinematics, and dynamics of a mobile robot," *IEEE Trans. Robot.*, vol. 25, pp. 367–381, 2009.

[11] D. Kiss and G. Tevesz, "A receding horizon control approach to navigation in virtual corridors," in *Applied Computational Intelligence in Engineering and Information Technology*, ser. Topics in Intelligent Engineering and Informatics. Springer-Verlag, 2012, vol. 1, pp. 175–186.

[12] R. Rojas and A. Förster, "Holonomic control of a robot with an omnidirectional drive," *Künstliche Intelligenz*, vol. 11, pp. 12–17, 2006.

[13] M. Keil and J. Snoeyink, "On the time bound for convex decomposition of simple polygons," *International Journal of Computational Geometry and Applications*, vol. 12, pp. 181–192, 2002.