A Convergent Dynamic Window Approach to Obstacle Avoidance

Petter Ögren
petter@math.kth.se
Optimization and Systems Theory
Royal Institute of Technology
SE-100 44 Stockholm, Sweden

Naomi Ehrich Leonard
naomi@princeton.edu
Mechanical and Aerospace Engineering
Princeton University
Princeton, NJ 08544, USA

***Abstract*— The dynamic window approach (DWA) is a well known navigation scheme developed by Fox et. al. [1] and extended by Brock and Khatib [2]. It is safe by construction and has been shown to perform very efficiently in experimental setups. However, one can construct examples where the proposed scheme fails to attain the goal configuration. What has been lacking is a theoretical treatment of the algorithm's convergence properties. Here we present such a treatment by merging the ideas of the DWA with the convergent but less performance-oriented scheme suggested by Rimon and Koditschek [4]. Viewing the DWA as a Model Predictive Control (MPC) method and using the Control Lyapunov Function (CLF) framework of [4] we draw inspiration from a MPC/CLF framework put forth by Primbs [3] to propose a version of the DWA that is tractable and convergent.**

***Index Terms*— Mobile Robots, Robot Control, Obstacle Avoidance, Navigation Function, Model Predictive Control, Receding Horizon Control, Lyapunov Function.**

I. INTRODUCTION

The problem of robotic motion planning is a well-studied one, see for instance [10]. One of the early approaches is artificial potential fields, where the robot is driven by the negative gradient of a sum of potentials from different obstacles and the goal. Many of these methods suffer from the problem of local minima, i.e. positions different from the goal, where the robot could get stuck. The most refined method along these lines is perhaps [4], where advanced mathematics is applied to construct an artificial potential function without such local minima. Other approaches have used ideas from fluid mechanics or electro magnetics [18],[19] to construct functions free of local minima, but they are in general computationally intensive and therefore ill suited for dynamic environments.

There is also a direction of research towards biologically motivated, non-model-based methods. These include fuzzy or neural approaches and the behavior-based paradigm described in e.g. [12]. A recent attempt to incorporate mathematical formalism into such frameworks can be seen in [8], but they are still in general hard to analyze from a convergence perspective. In cluttered environments the exact shape of the robot must be taken into account; these aspects have been investigated in [16], [13]. Most work however, as well as our study, assume a circular or point shaped robot.

We build our proposed scheme on the combination of a model-based optimization scheme and a convergence-oriented potential field method. A large class of model-based techniques use optimization to choose from a set of possible trajectories, [1], [2], [14], [15]. We argue that these optimization based techniques can be seen as applications of a Model Predictive Control approach (or, equivalently, a Receding Horizon Control approach). Having made this observation, we look at the method of Exact Robot Navigation using Artificial Potential Functions put forth by Rimon and Koditschek [4]. After constructing a continuously differentiable navigation function (artificial potential) Rimon and Koditschek use Lyapunov theory to prove convergence. Bounded control and safety is shown, but the method has the drawback of almost never using the full control authority and is furthermore not suited for dynamic environments where fast response to changes is essential. Inspired by Primbs [3] we present a way to merge the convergent Koditschek scheme with the fast reactive DWA. This is done by casting the two approaches in a MPC and CLF framework respectively and combining the two as suggested by Primbs. The conceptual flowchart of this combination is depicted in Figure 1.

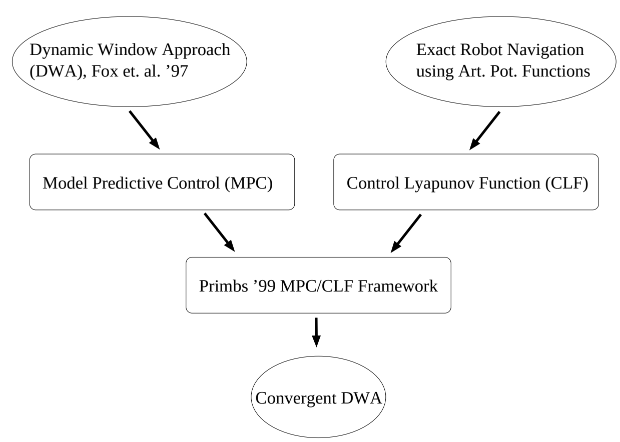

Fig. 1. The idea of the proposed approach can be seen as combining elements from the original DWA with a construction guaranteeing convergence proposed by Rimon and Koditschek. This is done with inspiration from a MPC/CLF framework suggested by Primbs.

Previous short versions of this paper appeared in IFAC 2002 and IEEE IROS 2002. The work by the first author was sponsored by the Swedish Foundation for Strategic Research through its Center for Autonomous Systems at KTH. The work by the second author was partially supported by the Office of Naval Research under grants N00014–98–1–0649 and N00014-01-1-0526, by the National Science Foundation under grant CCR–9980058 and by the Air Force Office of Scientific Research under grant F49620-01-1-0382.

Corresponding Author: Petter Ögren, Fax: +46 8 22 53 20

The organization of this paper is as follows. In Section II we review the work of [1] and [2] as well as [3]. Then, we explain our proposed scheme in detail in Section III. In Section IV we discuss the theoretical properties of our approach and in Section V we give a simulation example. The conclusions can be found in Section VI. This paper builds on our earlier work [5], [6].

II. Previous Work used in this Paper

In this section we discuss the ideas of Figure 1 in some detail.

A. *The Dynamic Window approach and its extension*

The Dynamic Window approach [1] is an obstacle avoidance method that takes into account the dynamic and kinematic constraints of a mobile robot (many of the vector field and vector field histogram approaches do not). The basic scheme involves finding the *admissible controls*, those that allow the robot to stop before hitting an obstacle while respecting the above constraints. Then an optimization is performed over those admissible controls to find the one that gives the highest utility in some prescribed sense. There are different suggestions for the utility function in [1] and [2], including components of velocity alignment with preferred direction, large minimum clearances to obstacles, possibility to stop at the goal point and the norm of the resulting velocity vector (large being good).

Brock et. al. [2] extended the work in [1] by looking at holonomic robots (Fox et. al. considered synchro drive ones) and more importantly by adding information about connectivity to the goal. The latter was done by replacing the goal direction term with the gradient of a *navigation function* defined as the length of the shortest (unobstructed) path to the goal [10]. Thus, they were able to eliminate the local minima problems present in so many obstacle avoidance schemes (hence the term "Global" in the title of [2]).

The experimental results reported in [1] and [2] are excellent, showing consistent safe performance at speeds up to 1.0 m/s with a Nomadic Technologies XR4000 robot, [2]. The results demonstrate an algorithm that is safe by construction (in the sense that the robot never hits obstacles) and displays high efficiency in extensive experimental tests. But although Brock and Khatib argue that the use of a navigation function makes the approach "Global", it is never formally shown. In fact, examples can be constructed where the robot enters a limit cycle, never reaching the goal or actually consistently moves away from the goal (see Section III-D).

B. *Exact Robot Navigation using Artificial Potential Fields*

One of the main contributions of [4] is the clever construction of a special artificial potential. This potential has no local minima except the global minimum at the goal. It is furthermore continuously differential and attains its maximum value at all the obstacle boundaries.

Combining such a potential $NF(r)$, where $r \in \mathbb{R}^2$ is position, with a kinetic energy term one can construct a control Lyapunov function as

$$V(r,\dot{r}) = \frac{1}{2}\dot{r}^T\dot{r} + NF(r).$$

The dynamics $\ddot{r} = u$ and the control $u = -\nabla NF + d(r,\dot{r})$, where $d(r,\dot{r})$ is a dissipative force, yields $\dot{V} \leq 0$ and stability in the Lyapunov sense.

The controls are bounded since ∇NF is continuous on a compact set. Further, the fact that $\dot{V} \leq 0$ and $NF(r) = NF_{max}$ at the obstacle boundaries guarantees against collisions if the initial velocity is small enough. The construction is however only valid in *a priori* known 'generalized sphere worlds' containing obstacles of specific categories. To adjust the scheme to a specific robot requires a scaling of NF to make the maximal $\nabla NF(r)$ smaller than the robot control bound. This in turn will make the vast majority of prescribed control signals far below the bound resulting in very slow progress towards the goal.

We draw inspiration from the work in [4]; however, we relax the constraints on ∇NF from continuous to piecewise continuous and remove the requirement that $NF(r) = NF_{max}$ at the boundaries. By doing this we hope to gain computational efficiency in calculating (and recalculating in case of new information) the NF and also to allow quite general obstacle shapes. Removing these constraints furthermore gives the possibility to add the constraint $||\nabla NF(r)|| = k$, used to enhance performance.

C. *Control Lyapunov Functions and Model Predictive Control*

In an interesting paper by Primbs, Nevistic and Doyle, [3], the connection between Control Lyapunov Functions (CLF) and Model Predictive Control (MPC) is investigated (MPC is also known as Receding Horizon Control, RHC). They note the complementary properties shown in Table 1. In view of these properties they suggest the following framework to combine the complementary advantages of each approach. The control law is chosen to satisfy a short-horizon optimal control problem under constraints that ensure the existence of a CLF.

CLF	MPC
Global information	Local information
Stability oriented	Performance oriented
Off-line analysis	On-line computation
Hamilton Jacobi Bellman type	Euler-Lagrange type

TABLE I

COMPLEMENTARY PROPERTIES OF THE TWO APPROACHES.

The problem becomes one of finding a control u and a CLF $V(x)$ that satisfy (1) through (4) as follows:

$$\inf_{u(\cdot)} \quad \int_t^{t+T} (q(x) + u^T u) d\tau \tag{1}$$

$$\text{s.t.} \quad \dot{x} = f(x) + g(x)u \tag{2}$$

$$\frac{\partial V}{\partial x}(f + gu) \leq -\epsilon\sigma(x(t)) \tag{3}$$

$$V(x(t+T)) \leq V(x_\sigma(t+T)), \tag{4}$$

where $q(x)$ is a cost on states, $\epsilon > 0$ is a scalar, $T > 0$ is the horizon length, $\sigma(x)$ is a positive definite function and x_σ

is the trajectory when applying a pointwise minimum norm control scheme (for details see [3]). This formulation inspires our choice of a more formal, continuous time formulation of the Dynamic Window Approach, allowing us to prove convergence.

III. A Proven Convergent Dynamic Window Approach

In the main parts of this paper we will use the notation $x = (r, \dot{r}) = (r_x, r_y, \dot{r}_x, \dot{r}_y)$ for the state of the system. We adopt the robot model from [2], which is basically a double integrator in the plane $\ddot{r} = u, \; r \in \mathbb{R}^2$ with bounds on the control $||u|| \leq u_{max}$ and on the velocity $||\dot{r}|| \leq v_{max}$. Note that it was shown in [9] that an off-axis point on the unicycle robot model described by

$$\begin{aligned} \dot{r}_x &= v\cos\theta, \\ \dot{r}_y &= v\sin\theta, \\ \dot{\theta} &= \omega, \\ \dot{v} &= F/m, \\ \dot{\omega} &= \tau/J, \end{aligned}$$

can be feedback linearized to $\ddot{r} = u$. For the environment we assume that the robot's sensors can supply an occupancy grid map, i.e. a rectangular mesh with each block being marked as either free or occupied, over the immediate surroundings. Here the size of the robot must be taken into account and additional safety margins can be added. A position marked as free means that the robot doesn't intersect any obstacles when occupying that position. Thus a map can be incrementally built as the robot moves around. We assume, as did Brock and Khatib, that the simultaneous localization and mapping (SLAM, see e.g. [20]) problem is solved for us. In cases where we are given a probabilistic occupancy grid a heuristic weighting in the utility function (10) is conceivable.

A. Navigation Function

In our setting, the navigation function $NF(r)$, [10], [2], [4] approximately maps every free space position to the length of the shortest, collision-free path going from that position to the goal point. Note that this version, taken from [2] is slightly different (e.g. no C^2 property) from the definition in [4]. To be more precise we make the following definition.

Definition 3.1 (Navigation Function): By a Navigation Function (NF) we mean a continuous function defined on the simply connected part of the obstacle-free space $\subset \mathbb{R}^2$, containing the goal point and mapping to the real numbers. A NF has only one local minimum, which is also the global minimum. The set of local maxima is of measure zero. ∇NF is piecewise continuous and the projection of the left and right limits along the discontinuity edges satisfy $e^T \nabla_{left} NF(r) = e^T \nabla_{right} NF(r)$, where $e \in \mathbb{R}^2$ is the direction of the edge and $\nabla_{left(right)} NF(r)$ is the gradient on each side of it.

Before investigating how to construct such a function in detail we note that it is shown in [2] how to deal with the case when the robot at first only knows its immediate surroundings by use of its sensors. The idea is to assume free space at the unknown positions and then recalculate the navigation function when sensor data showing the opposite arrives. In this way the robot guesses good paths and updates them when new information arrives. The information is immediately taken into account in the optimization (10) thus avoiding collisions. The less urgent recalculation of the Navigation Function is then done. These updates are made at a time scale much slower than the actual motion control so in our considerations below we assume the map to be static. Brock and Khatib, [2], used the gradient of the navigation function as the desired heading instead of using just the goal direction as Fox did [1].

To compute the navigation function we will use a technique similar to the one suggested in [2]. There, however, the NF was piecewise constant in the grids; here, we need a local-minima-free continuous function defined on all free space making things somewhat more complicated. The basic idea is to solve the shortest path problem in a graph discretization and then make a careful continuous interpolation for the positions in between the discretization points. An example of the discretization can be seen in Figure 2.

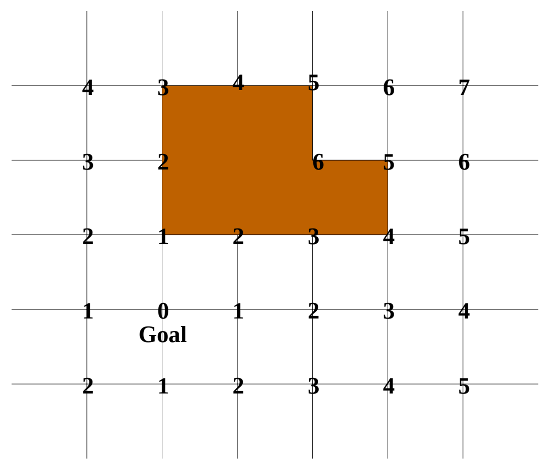

Fig. 2. The graph discretization and computed shortest path distances. The shaded squares correspond to an obstacle.

Lemma 3.1 (Construction of NF): A Navigation Function can be created by the following procedure:

1) Make a graph out of the rectangular mesh of the obstacle grid map, with vertices at the corners of each square and edges along the square edges. Remove vertices and edges that are in the interior of obstacles.
2) One of the vertices is chosen as goal point.
3) Solve the shortest path problem in the graph (can be done with polynomial time algorithms, [11]). Mark each vertex with the corresponding path length and let this length be the value of NF at the vertex.
4) Divide the squares into triangles by drawing a diagonal through the corner with highest NF value (this is shown to be unique below).
5) In the interior of each resulting triangle, let $NF(r)$ be a linear interpolation between NF at the three vertices, i.e. let the value of $NF(r)$ be a plane intersecting the three vertices.

Proof: We begin by showing that there are no local minima on the graph vertices and edges. Note that for a given pair of vertices (A, B), if one path between A and B has even (odd) length (in multiples of the edge length l), so has any other path. Therefore, two neighbors cannot have the same NF value. Since $NF(r)$ on the edges is a linear interpolation between the value on the vertices, the edges have no local minima. Furthermore, from every vertex, there is a shortest path to goal and along this path NF decreases monotonically. Thus, there can be no local minima vertex, except the goal point.

Now we look at the values in the interior of the squares. Given a square, look at the corner of lowest value, which may or may not be unique. If it is unique and of value k, then the two adjacent corners must have value $k + l$ where l is the side length and the opposite corner must have $k + 2l$ since no adjacent corners have the same value (see Figure 3, left). The diagonal is from $k + 2l$ to k and in this case the two triangles will actually form an inclined plane which obviously has no interior local minima.

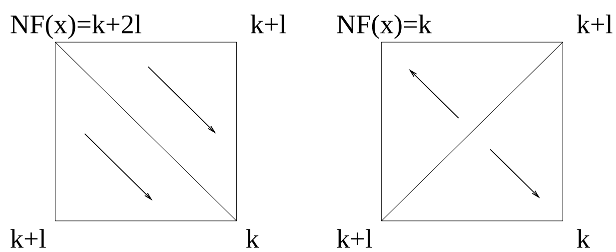

Fig. 3. The two possible cases, unique lowest $NF(x)$ (left, as in the square northwest of the goal point in Figure 2) and two equal (right, as in the square northeast of the obstacle in Figure 2). The arrows indicate $-\nabla NF$.

If the lowest value is not unique, the opposite corner must have the same value k and the two adjacent ones $k + l$ (see Figure 3, right). The diagonal is between the two $k + l$ corners (unique as stated in the construction) and this diagonal composes a ridge of local maxima. There are however no local minima.

Thus we have seen that there is only one local minima, the goal point. There might be local maxima on some diagonals, as in Figure 3 (right), but they are isolated lines and thus of measure zero. Finally, since NF is composed of triangles glued together the projection along the edges fulfills $e^T \nabla_{left} NF(r) = e^T \nabla_{right} NF(r)$ as above. ■

With a Navigation Function at our disposal, we are ready to look at the actual choice of control.

B. General Control Scheme

The basic idea for the convergence proof is the same as in [4]. First we write the problem as a conservative system with an artificial potential and then we introduce a dissipative control term. In the conservative system we choose the artificial potential to be $(k/\sqrt{2})NF(r)$, where k is a positive constant that must be chosen smaller than the control bound u_{max}. The Control Lyapunov Function is

$$V(x) = \frac{1}{2}\dot{r}^T\dot{r} + \frac{k}{\sqrt{2}}NF(r), \tag{5}$$

where $NF(r)$ is the navigation function as explained above. Incorporating the upper bounds on the control magnitude, we define the dissipative control set as follows.

Definition 3.2 (Dissipative Controls, $C_d(r, \dot{r})$):

$$\begin{aligned} C_d(r, \dot{r}) = \{u : \quad & u = u_c + u_d, \\ & u_c = -\frac{k}{\sqrt{2}}\nabla NF \\ & u_d :\ u_d^T\dot{r} < -\epsilon||\dot{r}|| < 0,\ \text{if } \dot{r} \neq 0 \\ & ||u|| \leq u_{max}, \} \end{aligned}$$

for some given $\epsilon > 0$. We write $u \in C_d(r, \dot{r})$.

A typical shape of the $C_d(r, \dot{r})$ set is the shaded regions in Figure 4 where C_1 (the nine dots) is a discretized (finite) subset of C_d. Note that $||u_c|| = ||(k/\sqrt{2})\nabla NF(r)|| = k$, since $||\nabla NF(r)|| = \sqrt{2}$, (the directional derivative along each axis of the grid is equal to 1, the gradient direction is diagonal to the grid and the magnitude is $\sqrt{1^2 + 1^2}$, see Fig. 3). Thus u_c lies on a circle of radius k. The outer circle of radius u_{max} bounds the control set.

Now the problem is to make sure the robot does not run into obstacles. In the standard Dynamic Window Approach this is taken care of by choosing among *admissible*, i.e. not colliding, controls in an optimization. Here we shall do the same.

A general formulation of the combined CLF/MPC scheme now looks like this

$$\min_{u(\cdot)} \quad V(x(t + T)) \tag{6}$$

$$\text{s.t.} \quad u(s) \in C_d(r, \dot{r}),\ \forall s \in [t, t + T] \tag{7}$$

$$r(s) \text{ collision free},\ \forall s \in [t, t + T] \tag{8}$$

$$\dot{r}(t + T) = 0, \tag{9}$$

where T is the horizon length of the MPC. Here (7) gives stability in the Lyapunov sense. Safety is guaranteed by (8,9), i.e. a planned collision free trajectory ending with the robot standing still. This corresponds to the policy of driving a car slow enough so that you can always stop in the visible part of the road. Perhaps somewhat conservative on a highway, but sensible on a small forest road where fallen trees might block your way.

Note that the above formulation can in principle be applied to enhance the performance of any approach with an artificial potential having a well-behaved gradient, e.g. the original NF suggested in [4] or a version of [18], [19]. We still believe however that the NF suggested above is a choice yielding high robot velocities and good computational efficiency.

The optimal control problem of the MPC above can be computationally intensive, as seen in related approaches such as [7]. Therefore, we devote the next section to showing how a very coarse discretization can still yield quite good performance. In Section IV we give detailed proofs of the theoretical properties of the proposed method.

C. Discretized Control Scheme

To end up with a computationally tractable version of the MPC above we discretize the set of dissipative controls, C_d, into two sets with piecewise constant, relative to the velocity direction, controls. The horizon length is $T = T_1 + T_2$ where T_1 is the time over which the resulting control will be applied. After time T_1 a new optimization is performed. The control set C_d is discretized into two sets, C_1 and C_2, corresponding to the two time intervals of length T_1 and T_2. A third set C_s is used when starting from a standstill, see Definition 3.3. To make the scheme precise we formulate the following algorithm.

Algorithm 3.1 (Control Scheme): The control algorithm is composed of the following steps, where number 3 is the main one.

1) If $\dot{r} = 0$, choose the control pair $(C_s, C_{2middle})$, as given in Definition 3.3, i.e. start out in a good direction.
2) Else, if $t \geq t_0 + T_{timeout}$, then set the new $t_0 = t$. If furthermore, $V(t_0) - V(t) \leq \Delta V_{timeout}$, then take the C_2 part C_{2prev} of the previous control pair and choose the control pair (C_{2prev}, C_{2prev}), i.e. reset and stop safely.
3) Else, choose the optimal solution to the MPC control problem

$$\min_{C_1 \times C_2} \quad V(x(t+T)) \qquad (10)$$
$$\text{s.t.} \quad x(\cdot) \text{ is collision free.}$$

4) Apply the first part of the chosen control pair for T_1 time units, then repeat from 1.

Here $\Delta V_{timeout}$ is a user defined decrease in V over the time $T_{timeout}$. This timeout construction is needed to guarantee against the hypothetical case of the robot velocity slowly approaching zero. Then the decrease bound $\dot{V} \leq \epsilon ||\dot{r}||$ is not enough to yield convergence. The control sets are defined as follows.

Definition 3.3 (Control Sets, C_1, C_2 and C_s): Let C_1 be the set of 9 controls depicted in Figure 4, left and C_2 be the set of 4 controls (a subset of C_1) in Figure 4, right. Note that C_1 and C_2 are defined relative to the velocity direction $\dot{r}$. Furthermore, let C_s be the control directed towards the corner of the current grid with the lowest $NF(r)$ value (if there are more than one such corner the one closest to the robot is chosen) and with control magnitude such that after applying C_s for time T_1 and the middle, hardest braking, control of C_2 for time T_2 the robot stops at the corner.

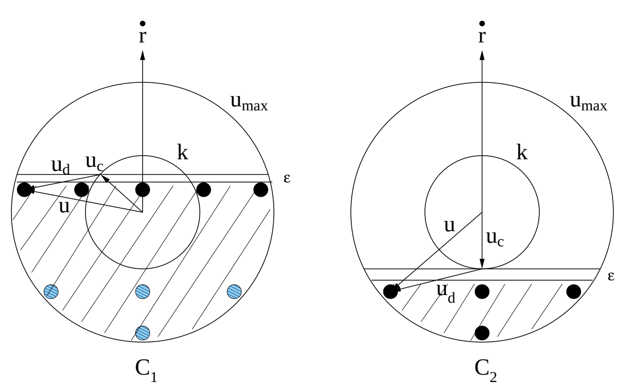

Fig. 4. The Control sets C_1 and C_2.

Note that $C_s \subset C_d$ since the acceleration is towards the corner closest to goal. We further assume the grids to be small enough to be traversable by the $C_s \times C_2$ control in $T_1 + T_2$ time units. To make the 'current grid' unique, positions on the boundary of grids are assigned to belong to one of the adjacent grids.

The interval $[t, t+T]$ is divided into two parts, $[t, t+T_1]$ and $[t+T_1, t+T]$, where T_1 is the time step of the MPC control loop. In the first part $[t, t+T_1]$, a control from C_1, the set of controls in Figure 4, left, is chosen. In the second part $[t+T_1, t+T]$, a control from C_2, the set of controls in Figure 4, right, is chosen. Note that the controls in the sets above are constant with respect to the direction of the robot velocity, $\dot{r}$. C_2 consists of 4 controls (the dots in Figure 4, right) all reducing the speed of the robot. C_1 consists of 5 controls on the border of the dissipative region (shaded area) and the 4 C_2 controls (the shaded dots). Thus, the whole set $C_1 \times C_2$ consists of $(5+4) \cdot 4 = 36$ control sequences.

In Figure 5 we see parts of an executed trajectory together with all the options evaluated in the optimization. An obstacle as well as the level curves of the navigation function are also depicted.

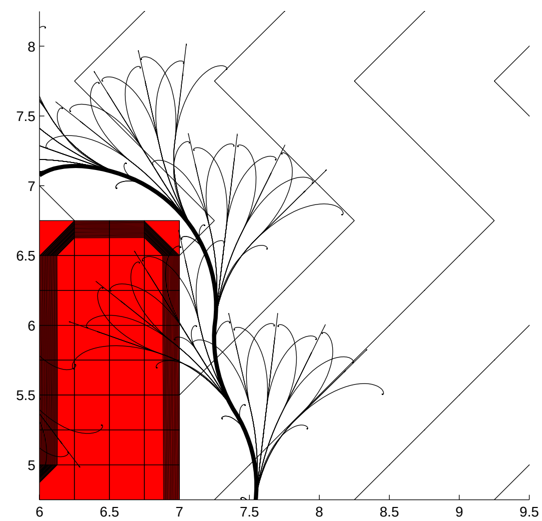

Fig. 5. An obstacle, level curves of the NF and parts of a trajectory as well as all the considered options.

It can be seen that in the first time step the leftmost control of C_1 is chosen and in the second time step the rightmost is. Since T_1 is the length of the time step, it is only the C_1 part of $C_1 \times C_2$ that is actually executed.

The purpose of the C_2 part is to guarantee safety. The time $T = T_1 + T_2$ should be chosen long enough for the robot to stop at (or before) time $t + T$ (since all the C_2 controls are braking). The optimization is done with the constraint that the resulting trajectory doesn't hit any obstacles, hence the choice of the rightmost control the second time. The fact that the previously chosen C_2 control is an option in C_1 makes the last part of the (safe) previously chosen control sequence an option in the next optimization. As a result there is always at least one admissible (and therefore safe) control sequence available. As stated above, this is similar to always making sure you can stop in the visible part of the road when driving a car.

One might argue that discretization and exhaustive search is an inelegant solution. But we chose it for two reasons. The utility function $V(x(t+T))$ varies rather slowly over the admissible set of controls and this set in turn can be very complex, e.g. unconnected if there are traversable paths to the right and to the left but the road straight ahead is blocked by an obstacle. The constraints are also far from being a differentiable function inequality. Due to these facts a steepest decent approach will not do well.

D. Example of Convergence Failure of Previous Approach

The Utility function of [2] that is to be maximized is

$$\begin{aligned}\Omega_g(p,v,a) &= \alpha \cdot nf1(p,v) + \beta \cdot \text{vel}(v)\\ &\quad +\gamma \cdot \text{goal}(p,v) + \delta \cdot \Delta nf1(p,v,a)\end{aligned}$$

where p, v, a is the current position and desired velocity and acceleration. $\alpha,\ \beta,\ \gamma,\ \delta$ are scalar weights, $nf1(p,v)$ increases if the velocity is aligned with the navigation function gradient, vel(v) increases with velocity (if far from goal), goal(p,v) is binary, 1 if the trajectory will pass through the goal point and $\Delta nf1$ is the decrease in navigation function value.

Consider a 'T' shaped, very narrow corridor, with the robot initially in the top left end and the goal defined in the bottom end. This will leave the robot accelerating maximally towards the right. If the corridor is long and narrow enough, the speed is going to be too great to allow a right turn at the intersection. Thus, the robot will continue away from the goal. In particular, when the corridor is very narrow and turning is not an option, the robot must either brake or not. If the weights are such that the velocity term, vel(v), outweighs the $\Delta nf1$ term, the robot will just keep on going. Otherwise, it will brake maximally. If, however, the acceleration is as powerful as the retardation, the robot will oscillate back and forth in the upper part of the 'T' and never be able to make the sharp turn into the goal part of it. A similar counter example was independently presented in [7]. Note that the Lyapunov property of equation (11) below removes these kinds of problems in the proposed approach.

IV. Proof of Convergence and Safety

Before we formulate the main theorem of this paper we need a lemma.

Lemma 4.1 (Control Lyapunov Function): The function

$$V(x) = \frac{1}{2}\dot{r}^T\dot{r} + \frac{k}{\sqrt{2}}NF(r),$$

is a Control Lyapunov Function and any C_d control satisfies the following inequality

$$\dot{V}(x) \leq -\epsilon||\dot{r}||. \tag{11}$$

Proof: The candidate Lyapunov Control Function is $V(x) = \frac{1}{2}\dot{r}^T\dot{r} + \frac{k}{\sqrt{2}}NF(r)$, which is clearly positive definite with a global minimum at $x_{goal} = (r_{goal}, 0)$. Differentiating with respect to time gives $\dot{V}(x) = \dot{r}^T u + \frac{k}{\sqrt{2}}\dot{r}^T\nabla NF(r) = \dot{r}^T(u_c + u_d + \frac{k}{\sqrt{2}}\nabla NF(r)) \leq -\epsilon||\dot{r}||$, by the constraints on u. The Navigation Function is however not differentiable everywhere. Along the triangle edges of NF there are in general two different (left and right) limits of the gradient. The projections along the edge is however the same, $\dot{r}^T\nabla_{left}NF(r) = \dot{r}^T\nabla_{right}NF(r)$ making the inequality true. Furthermore, it is this projection that is needed when determining u according to the definition of u_d, see Definition 3.2 and Figure 4.

If $\dot{r}$ is not parallel to an edge, the problem with undefined $\nabla NF(r)$ in the control will only occur in one isolated time instant and thus $\int \dot{V}\ dt$ is not changed by whatever value we use. ■

Theorem 4.1 (Finite Completion Time): Suppose the control scheme in Algorithm 3.1 is used and there is a traversable path from start to goal in the occupancy grid. Then, the robot will reach the goal position in a time bounded above by

$$\begin{aligned}T_{goal} &\leq (V_0\frac{\sqrt{2}}{k}\frac{1}{l}2)^2(T_{timeout} + T_1 + T_2)\\ &\quad + \frac{V_0}{\Delta V_{timeout}}T_{timeout},\end{aligned}$$

where $V_0 = V(x_{start})$ is the value of the CLF at the starting position.

Proof: By Lemma 4.1 we have that $\dot{V}(x) \leq -\epsilon||\dot{r}||$. Thus the system is stable in the sense of Lyapunov.

After a stop the robot starts moving towards the corner of the current grid closest to goal, i.e. with lowest $NF(r)$. Then, the optimization improves on this, making the outcome at least as good as stopping at that corner (at a stop, $V(r, \dot{r} = 0) = \frac{k}{\sqrt{2}}NF(r)$). Together with the fact that $\dot{V}(x) \leq 0$, this means that the robot will never stop in that grid again. Thus the number of possible grids to occupy (which is finite) is reduced by at least one between each pair of stops. Since NF is the path length, the number of possible grids is bounded by $(V_0\frac{\sqrt{2}}{k}\frac{1}{l}2)^2$. Therefore, this is also a bound on the number of stops. The timeout induces a stop if $V(t_0) - V(t_0 + T_{timeout}) \leq \Delta V_{timeout}$. This makes the number of $T_{timeout}$-sized intervals without a stop bounded by $\frac{V_0}{\Delta V_{timeout}}$. Combining the two we get $T_{goal} \leq (V_0\frac{\sqrt{2}}{k}\frac{1}{l}2)^2(T_{timeout} + T_1 + T_2) + \frac{V_0}{\Delta V_{timeout}}T_{timeout}$. ■

Remark 4.1: Note that this is an extreme worst case analysis. In the simulations the robot did not stop at all before reaching the goal position.

Theorem 4.2 (Safety): Suppose the control scheme in Algorithm 3.1 is used and the robot starts at rest in an unoccupied position. Then, the robot will not run into an obstacle.

Proof: The proof relies on the recursive structure of $C_1 \times C_2$. The subset of non colliding controls in $C_1 \times C_2$ (that we are optimizing over) is never empty since we can always choose the C_2 (not yet applied) part of the previous $C_1 \times C_2$ control sequence as our new C_1 control. ■

V. Simulation Example

To illustrate the approach we chose a setting with three large obstacles in a 9 by 9 meter area, as seen in Figure 6. We used parameters for the Nomadic Technologies XR4000 robot obtained from [2], $||u|| \leq u_{max} = 1.5m/s^2,\ ||\dot{r}|| \leq v_{max} = 1.2m/s$. The resulting robot trajectory and the level curves of the navigation function are also depicted in the figure. Note the absence of local minima as guaranteed by the construction in Lemma 3.1. In the beginning and the end of the trajectory it can be seen how the MPC minimization in equation (10) favors going perpendicular to the level curves.

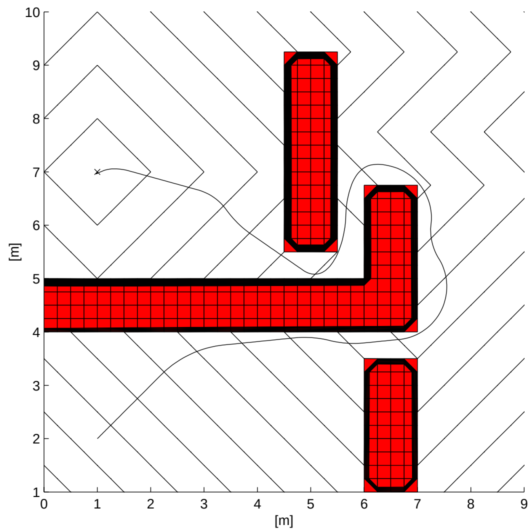

Fig. 6. The Obstacles and robot trajectory.

In Figure 7 parts of the trajectory as well as the Model Predictive Control options are shown. The robot slows down in two places, as can be seen in Figure 8. These are located at $(6.5, 7.2)$ and $(5.5, 5.5)$ respectively. In the second instance all the non braking C_1 controls makes the robot collide. The sharp braking right turn, the C_2 part of the previous choice, is however safe and therefore applied. At the first instance, $(6.5, 7.2)$, the non braking right turn is safe, but the braking left turn yields a lower $V(x(t+T))$ value (it is closer to the goal) and is thus chosen.

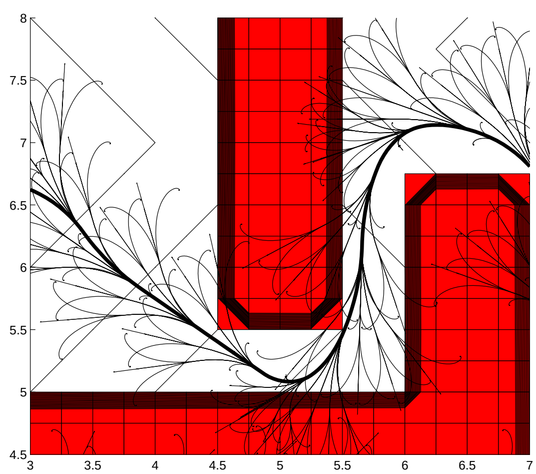

Fig. 7. A closeup of parts of the trajectory including all the evaluated options of the Model Predictive Control. The robot slows down in two places, $(6.5, 7.2)$ and $(5.5, 5.5)$. in order not to collide.

To enforce the $||\dot{r}|| \leq v_{max} = 1.2m/s$ bound, we impose the additional acceleration constraint

$$u^T \dot{r} \frac{1}{||\dot{r}||} \leq \frac{v_{max} - ||\dot{r}||}{T_1}.$$

This constraint is of the same sort as the dissipative constraint, i.e. it can be depicted as another horizontal line above or below the ones in Figure 4. If the new constraint is more restrictive (lies below the first two) the C_1 options will be placed just below it instead of just below the others. Thus, in Figure 8 the robot reaches a "steady state" velocity close to $v_{max} = 1.2m/s$ in the open areas.

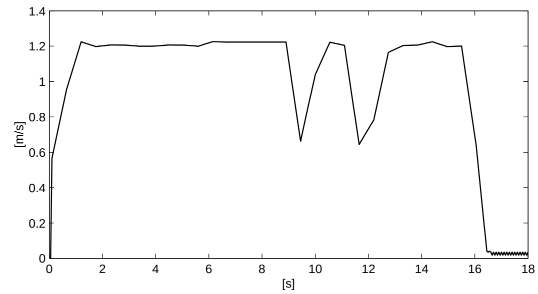

Fig. 8. The robot velocity. Note how the speed is limited by $v_{max} = 1.2m/s$ and how it decreases twice during the narrow passage.

VI. Conclusions

In this paper we have first presented the well known Dynamic Window Approach to fast and safe obstacle avoidance in an unknown environment. We then recast the approach in a continuous nonlinear control framework suggested by [3]. With a few changes to the basic scheme we were able to prove convergence to the goal position. This is significant since the earlier scheme could be subject to limit cycles and even divergence.

References

[1] D. Fox, W. Burgard and S. Thrun. The Dynamic Window Approach to Collision Avoidance, *IEEE Robotics and Automation Magazine*, March 1997.

[2] O. Brock and O. Khatib. High-Speed Navigation Using the Global Dynamic Window Approach, *IEEE International Conference on Robotics and Automation* , Detroit, Michigan, May 1999.

[3] J. A. Primbs, V. Nevistic and John C. Doyle. Nonlinear Optimal Control: A Control Lyapunov Function and Receding Horizon Perspective. *Asian Journal of Control*. Vol 1, No. 1, pp. 14-24, March 1999.

[4] E. Rimon and D. Koditschek. Exact Robot Navigation Using Artificial Potential Functions. *IEEE Transactions on Robotics and Automation*. pp. 501-518, Vol. 8, No. 5, October 1992.

[5] P. Ögren and N. Leonard. A Provably Convergent Dynamic Window Approach to Obstacle Avoidance, *IFAC World Congress*, Barcelona, Spain, July 2002.

[6] P. Ögren and N. Leonard. A Tractable Convergent Dynamic Window Approach to Obstacle Avoidance, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, October 2002.

[7] C. Stachniss and W. Burgard. An Integrated Approach to Goal-directed Obstacle Avoidance under Dynamic Constraints for Dynamic Environments, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, October 2002.

[8] P. Althaus and H. Christensen. Behavior Coordination for Navigation in Office Environments *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, October 2002.

[9] J. Lawton, B.Young and R. Beard. A Decentralized Approach To Elementary Formation Maneuvers. To appear in *IEEE Transactions on Robotics and Automation*.

[10] Latombe J. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991. ISBN 0-7923-9206-X

[11] Dijkstra, E.W., A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1 (1959), 269-71.

[12] Arkin, R. C. (1998).*Behavior-Based Robotics*. MIT Press, Cambridge, MA.

[13] Arras, K.O.; Persson, J.; Siegwart, R.; Tomatis, N. Real-time obstacle avoidance for polygonal robots with a reduced dynamic window *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, October 2002.

[14] Nak Yong Ko and Simmons, R.G. The lane-curvature method for local obstacle avoidance *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, October 2002.

[15] J.A. Benayas, J.L. Fernández, R. Sanz, A.R. Diéguez. The Beam-curvature Method: A New Approach For Improving Local Real-time Obstacle Avoidance *IFAC'02 15th International World Congress* , Barcelona, Spain, July, 2002.

[16] J. Minguez and L. Montano. Robot Navigation in very complex, dense and cluttered indoor/outdoor environments *IFAC'02 15th International World Congress* , Barcelona, Spain, July, 2002.

[17] Khatib, O., Minguez, J. and Montano, L. Reactive collision avoidance for navigation with dynamic constraints *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, October 2002.

[18] Iniguez, P.and Rosell, J.
A hierarchical and dynamic method to compute harmonic functions for constrained motion planning *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, October 2002.

[19] Eluagar, A. and Hussein, A.M. Motion planning using Maxwell's equations *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, October 2002.

[20] P. Jensfelt and S. Kristensen Active Global Localisation for a Mobile Robot Using Multiple Hypothesis Tracking. To appear in the IEEE Transactions on Robotics and Automation