Motion Control for Omni-Drive Servicerobots Under Kinematic, Dynamic and Shape Constraints

Timo Blender
University of Applied Sciences Ulm
Ulm, Germany
Email: blender@hs-ulm.de

Christian Schlegel
University of Applied Sciences Ulm
Ulm, Germany
Email: schlegel@hs-ulm.de

***Abstract*—In this paper, a fast reactive obstacle avoidance approach for omnidirectional driving is presented. The method is based on the dynamic window approach, but uses a cuboid instead of a window to limit the 3-dimensional search space accordingly to the dynamic constraints of the robot. Besides the kinematics and dynamics, the presented approach also considers the shape of a robot. To cope with the effort of the time consuming distance calculations, the remaining distance values are precalculated in an offline part and provided by a lookup table. This procedure is based on the Curvature Distance Lookup (CDL) approach which is extensively used in several real world robotic applications but which was so far only implemented for 2-DOFs. During the online phase, the extended approach enables the selection of a motion command from a wide range of curvatures (3-DOFs) within the current dynamic cuboid. The distance values are queried from the corresponding lookup table entries depending on the occupancy grid determined by latest sensor information. The reduced computational effort of the control loop allows to consider obstacle information from corresponding sources to the full extent and without preprocessing. Furthermore, complex heuristics can be implemented to evaluate a high number of omni-drive velocity triples in such a way that the driving behavior of the robot is influenced accordingly.**

I. Introduction

Robust navigation in open-ended environments is one of the key capabilities of a mobile robot in order to be able to perform complex tasks. The navigation problem can be solved in advance by a variety of global path planning methods that are able to compute a collision-free path to a specific goal location. However, such procedures are usually only reasonable in static environments with a-priori knowledge about the obstacle configuration since replanning in dynamic environments is often too slow. Hence, the contrary, or rather the complement to these global methods are the local ones which are also referred to as reactive collision avoidance. They are an essential component in many robotic systems since appropriate motion control in dependence of current sensor data is necessary for a safe navigation in an unknown and dynamic environment. In the past, comprehensive research in the field of mobile robotics led to the emergence of many different approaches for obstacle avoidance. A crucial feature in this domain is the consideration in what way a particular algorithm involves the three internal robot constraints - *kinematic, dynamic, shape* - in the decision-making process. An algorithm that does not consider kinematic constraints could provide a motion command which some robots could not physically execute. Equally unfavorable is the neglect of dynamic constraints in which case motions could be commanded that can not be reached by a robot with certain limited velocity or acceleration capabilities. Taking only an approximated shape of the robot into account leads to inaccuracies when calculating the remaining distance to obstacles. This could either cause a collision or restricts the maneuverability of the robot.

Despite this importance there are only a few number of approaches taking into account all three constraints. A major problem is the shape constraint because high computation effort is required for the distance calculations, especially when using a polygonal robot shape. However, the local nature of obstacle avoidance which runs in continuous cycles benefits from a procedure that determines an appropriate solution as fast as possible. The *Curvature Distance Lookup Approach (CDL)* [1] is based on the *Dynamic Window Approach (DWA)* [2] and is one of the rare approaches considering all three constraints. The CDL approach shifts the distance calculations to an offline part to reduce the effort within the control loop significantly. In contrast to DWA, the distance calculations can be performed with a polygonal shape because time is not a critical measure anymore.

The CDL approach was so far successfully implemented for kinematics with two degrees of freedom like differential-drive, synchro-drive or a tricycle kinematic [3]. This approach is extensively used in several real world robotic applications e.g. Collaborative Robot Butler Scenario, Intralogistics or RoboCup@Home [4]. This paper extends the CDL approach by adding a third degree of freedom in order to support omni-drive kinematics. This search space extension increases the agility of the robot significantly. The additional motions can be useful for different application scenarios e.g. docking or social navigation. In the first one, omni-drive motions can simplify approaching a goal location. In the second one, a lateral motion can be useful to express the robot intention (e.g. approaching the side of a hallway before traversing it). In general, the huge amount of possible movements allows to drive more sophisticated in situations where robots with 2-DOF kinematics would need to rotate in place to continue.

The paper is organized as follows: Section II gives an overview about established obstacle avoidance methods with different levels of constraints considerations. Section III and IV provides the functionality of the dynamic window approach and the CDL approach for 2-DOFs. These are the necessary basics for the omni-drive extension of the CDL approach which is presented in section V. Finally, in section VI, two experiments are presented showing the benefits of the new motion capabilities

978-1-4673-7929-8/15/$31.00

gained by the here proposed CDL omni-drive extension.

II. Related Work

The *vector field histogram (VFH)* [5], the *potential field method (PFM)* [6] and the *Nearness Diagram (ND)* [7] consider none of the three constraints. VFH uses an obstacle history in form of a certainty grid located in the near environment around the robot to compensate sensor uncertainty. A histogram provides probability values for the existence of an obstacle in a corresponding direction. Finally, a cost function is applied for every free area in order to determine the best steering direction with regard to certain properties, e.g. goal alignment. ND divides the environment of the robot in different sections and uses two diagrams to describe the obstacle configuration. These diagrams can be used to determine a safety level which defines one of five possible states that express the goal and obstacle configuration. Finally, a solution section can be computed based on the determined state. The authors show that their approach can cope with situations that constitute a problem for other local obstacle avoidance approaches. PFM considers a robot as a particle which is influenced by artificial forces from a potential field. The obstacles correspond to repulsive forces and the goal point constitutes the force of attraction. The application of a mathematical function will direct the robot to the goal while circumventing the obstacles. The potential field method treats the robot as a point with omnidirectional motion capabilities. Without further extensions none of the internal robot constraints are taken into account.
The *VFH+* [8] extension takes into account the kinematic of a robot in a simplified form by prohibiting all trajectories in form of circular arcs and straight lines that would cause a collision with an obstacle. A further approach that takes into account kinematic constraints is the *steering angle field* [9] method. It was tested with a rectangular robot and excludes all steering angles which cause collisions when using a non-holonomic kinematic. However arbitrary robot shapes are not possible and robot dynamics is not considered.
Approaches acting in the velocity space like the *curvature velocity method* [10] and its extension, the *lane curvature method* [11], consider the kinematics and dynamics of the robot. However, just like in the dynamic window approach, the robot shape is approximated by a circle.
A method that takes into account all three constraints is provided by Arras et al. [12]. They apply the dynamic window approach but restrict the search space by a dynamic line instead of a dynamic window. This step reduces the processing effort in such a way that polygonal robots can be applied. Minguez et al. [13] provides an extension in form of an abstraction layer wrapping the potential field method. The abstraction layer is responsible for the consideration of the three constraints. Their concept is generic which means that the abstraction layer can be used in combination with any other collision avoidance method that is designed for non-holonomic robots and which does not consider the robot constraints. First, the approach creates an arc reachable manifold that represents the obstacle configuration under consideration of the robot shape depending on trajectories assumed as circular paths. Based on the robots dynamics, non-admissible and reachable configurations can be determined when knowing the collision configurations from the arc reachable manifold. Finally, the manifold is transformed to a representation free of kinematic constraints where the robot is represented as a point. Now the potential field method can be applied to determine a motion command. The closest command which is reachable and admissible based on the computations from the abstraction layer is finally selected. The authors claim that their approach is faster than a number of other methods and in contrast to CDL, the obstacle space is not discretized. However, the procedure is restricted to non-holonomic robots. Hence, an extension of the CDL approach for omni-drives will provide a higher maneuverability while the accuracy of the computed solution is still appropriate.

III. The Dynamic Window Approach

The dynamic window approach [2] is a local reactive approach for collision avoidance under consideration of the dynamics and kinematics of a robot. The approach proceeds in the two-dimensional velocity space composed of the translational velocity v and the rotational velocity ω. The evaluation of various $\{v, \omega\}$ combinations (named as curvatures) within a control loop enables the determination of an appropriate velocity value for the next time step. Using a dynamic window allows to take the dynamics of the robot into account, thereby restricting the search space to those $\{v, \omega\}$ combinations which can be reached within the next time step. Further, a curvature $c = \{v, \omega\}$ is only admissible if the robot can come to a stop before a collision occurs. All reachable and admissible velocity combinations are finally evaluated by an objective function with respect to three properties: speed, distance, and orientation to a goal point. As already noted, the main drawback of the dynamic window approach is the approximation of the robot shape by a circle in order to reduce the required time for the distance calculations within the control loop. The computational effort for a polygonal robot would increase to $O(n^3)$ because the calculations must be performed for each segment [3].

IV. The Curvature Distance Lookup Approach

The CDL approach [1] represents an extension of the dynamic window approach. The main characteristic of the CDL approach is the usage of a lookup table to provide the remaining distance values for each curvature to the corresponding obstacle cells. These distance values are precalculated in an offline part and provided in a lookup table to avoid the time consuming distance calculations within the control loop (online part). This enables the usage of a polygonal robot shape because the offline part has no time critical requirements. The crucial foundation for the applicability of the CDL approach is the fact that the discretized $\{v, \omega\}$ combinations can be mapped to a limited number of curvatures. In other words, different $\{v, \omega\}$ tuples can be represented by the same curvature. This ensures that the size of the lookup table remains within reasonable limits. The distance calculations are performed in the local coordinate system of the robot. Therefore, the distance values are always independent from the position of the robot in the global coordinate system. Only grid cells in a certain vicinity to the robot (cartesian space) are considered. The environment can be evaluated by any source that provides information about occupied cells in a grid map (e.g. a laser scanner). During the online part, the distance to the closest obstacle cell for a curvature can be easily determined by applying a minimum operation over the corresponding lookup table entries.

A. Offline Part

1) Curvature indices: The curvature indices are the set of representative curvatures for which the distance calculation is applied in combination with the cells of the cartesian space. The curvature index i_c indicates the corresponding lookup table entry providing the distance value for curvature c to the obstacle cell $O(x, y)$, addressed by the corresponding index values i_x and i_y. A limited number of curvature indices is given by the outer cells of the discretized velocity space (Figure 1). Hence, the number of curvature indices is $(2 \cdot n_v) + (2 \cdot n_\omega) - 4$ instead of $n_v \cdot n_\omega$ where n_v and n_ω refer to the total number of discretized translational and rotational values. The representative value for a curvature index is given by its curvature line which can be described by the $atan2(v, \omega)$ function. An inner cell is associated with a curvature index whose curvature line constitutes the minimal deviation to the own curvature line of that inner cell.

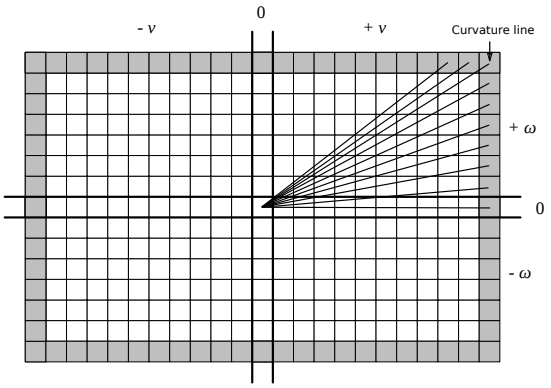

Fig. 1. The discretization window of the velocity space. The outer cells are used as curvature indices.

2) Lookup tables:

- $T_{index}[i_v, i_\omega] \Rightarrow i_c$: Contains the curvature index i_c for a $\{v, \omega\}$ tuple, addressed by the index values i_v and i_ω.
- $T_{dist}[i_x, i_y, i_c] \Rightarrow (d, \phi)$: Contains the distance d and the angular distance ϕ for the curvature index i_c to the obstacle cell $O(x, y)$, addressed by the index values i_x and i_y. The distance d and the angular distance ϕ are stored separately in this lookup table.
- $T_{acc,v}[i_c] \Rightarrow a_v$: Contains the maximum translational acceleration a_v for the curvature index i_c.
- $T_{acc,\omega}[i_c] \Rightarrow a_\omega$: Contains the maximum rotational acceleration a_ω for the curvature index i_c.

3) Distance calculation: When considering the robot with its center in the origin point of the cartesian space, the Instantaneous Center of Curvature (ICC) is located on the horizontal axis with the distance r given by the ratio v/ω of an arbitrary $\{v, w\}$ tuple. To check whether a curvature c overlaps a cartesian cell o under consideration of the robot contour, a circle k with center point $ICC(c)$ intersecting the cell o is defined. If k intersects a robot segment in a point p, then o is located on the curvature c and the remaining distance

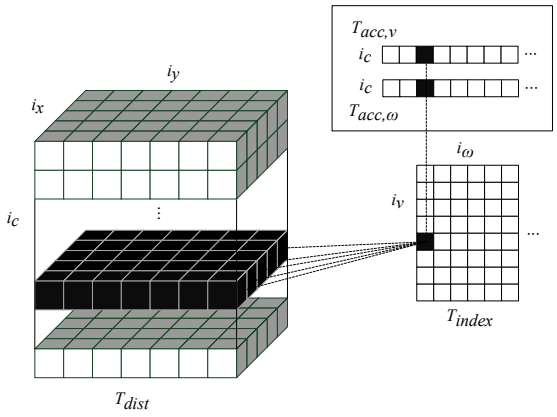

Fig. 2. The lookup tables of the CDL approach.

must be determined by initially calculating the angular distance α between p and o. The remaining distance is finally given by calculating $\alpha \cdot r(k)$ where $r(k)$ is the radius of circle k intersecting the robot contour.

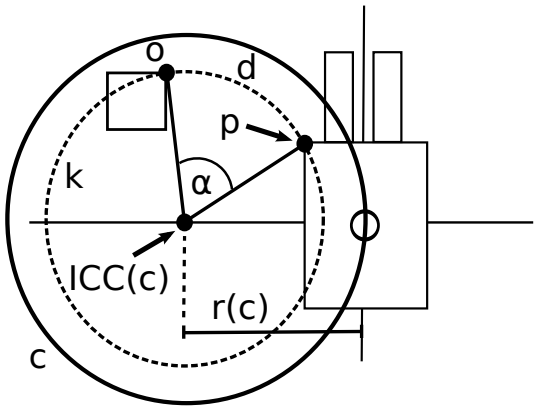

Fig. 3. The circle k intersects the robot shape. Hence, the remaining arc distance d between p and o must be calculated for a non-straight motion.

B. Online Part

The online part of the CDL approach follows the dynamic window approach but without the need to compute the remaining distance to obstacles. Instead, the values are queried from the corresponding lookup table entries. The following steps are performed during the control loop of the CDL approach:

1) Definition of the dynamic window.
2) Extraction of the relevant curvatures from the dynamic window.
3) Evaluation of the sensor readings and appropriate distance lookup for every occupied cell in combination with the curvatures extracted from the dynamic window.
4) Minimum distance determination for every curvature based on the information from the previous step.
5) Evaluation of all velocity combinations from the dynamic window which are admissible.
6) Commanding of the highest rated velocity combination.

V. OMNI-DRIVE EXTENSION

This section provides the necessary steps to extend the standard CDL approach to omni-drives. The new considerations affect mainly the offline part because the lookup table organization must be adapted in such a way that the distance values of the additional omni-drive specific motion types are included. Hence, it is necessary to determine how to compute the distance values of these new types and how to find a set of representative curvature indices. The problem of the curvature representation expands from two dimensions to three dimensions according to the number of degrees of freedom.

A. Offline Part

1) Motion types: Table I shows all motion types with 3-DOFs when considering all active ($\neq 0$) and not-active ($= 0$) combinations of the velocity triple $\{v_x, v_y, \omega\}$. Hence, there are 2^3 combinations. The possible motion types with 2-DOFs are included in table I (Nr. 1, 2, 3, 6) which means that the 2-DOF motions are a subset of the 3-DOF motions.

TABLE I. THE MOTION TYPES WITH 3-DOFS.

Nr.	v_x	v_y	ω	General	Description
1	$=0$	$=0$	$=0$	$(v_x \wedge v_y \wedge \omega) = 0$	No motion
2	$=0$	$=0$	$\neq 0$	$(v_x \wedge v_y) = 0 \wedge \omega \neq 0$	Rotation in place
3	$\neq 0$	$=0$	$=0$	$(v_x \vee v_y) \neq 0 \wedge \omega = 0$	Linear motion
4	$=0$	$\neq 0$	$=0$		
5	$\neq 0$	$\neq 0$	$=0$		
6	$\neq 0$	$=0$	$\neq 0$	$(v_x \vee v_y) \neq 0 \wedge \omega \neq 0$	Circular motion
7	$=0$	$\neq 0$	$\neq 0$		
8	$\neq 0$	$\neq 0$	$\neq 0$		

The distance values of the additional motion types must be covered by the new lookup table. Considering figure 3, the necessary information to perform the distance calculation for a corresponding curvature to a specific obstacle cell is the ICC location. For 2-DOF motions, the ICC location is fixed on the horizontal axis. Hence, the radius is the only measure to determine the exact location. This is not the case for 3-DOFs as can be seen in figure 4. Motion type 7 from table I is the reverse case to the classic non-straight motions with 2-DOFs (Nr. 6). Now, instead of the horizontal axis, the ICC location is fixed on the vertical axis depending on the radius of a curvature. For the *all active* case (Nr. 8), the ICC moves between the horizontal and vertical axis. In general, the ICC of a velocity triple is given by two measures:

- **Radius circle:** A circle with center point (0,0) and radius r given by equation 1.
- **ICC-line:** A line starting at point (0,0) with a slope depending on the ratio of the two translational velocities v_x and v_y.

The intersection of the radius circle and the ICC-line determines the ICC location of a corresponding velocity triple.

For 3-DOFs, the radius also depends on the ratio between the translational and the rotational velocity. However, the translational component is now considered as a vector $\vec{V}$ composed of v_x and v_y. The ratio between the magnitude of the vector $\vec{V}$ and the rotational velocity ω determines the radius of a curvature (equation 1).

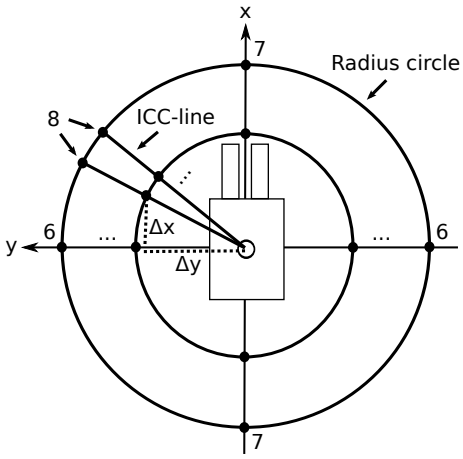

Fig. 4. The ICC locations of circular motions with 3-DOFs. The radius circle and the ICC-line are the two relevant measures. For motion type 6, the ICC-line value is located on the horizontal axis. For motion type 7, it is located on the vertical axis. The ICC-lines of motion type 8 lie somewhere between these axis depending on the v_x/v_y ratio.

$$r(v_x, v_y, \omega) = \frac{|\vec{V}|}{\omega} = \frac{\sqrt{v_x^2 + v_y^2}}{\omega} \quad (1)$$

When knowing the radius, the x and y coordinates of the ICC are finally given by equation 2 and 3.

$$ICC_x(v_x, v_y, \omega) = r(v_x, v_y, \omega) \cdot sin(atan2(v_y, v_x)) \quad (2)$$
$$ICC_y(v_x, v_y, \omega) = r(v_x, v_y, \omega) \cdot cos(atan2(v_y, v_x)) \quad (3)$$

2) Curvature indices: As already noted, the general idea of a curvature index is the representation of a set of velocity triples that summarizes the distance calculations to the obstacle cells of the cartesian space. This implies that the ICC locations of the triples must be as close as possible to the ICC location of the representative curvature index. In the case of 3-DOFs, velocity triples with a similar radius circle and a similar ICC-line should be grouped together. Finally a curvature index with the same properties should be assigned to each triple of this group. Figure 6 shows the distribution of ICC locations on different radius circles for a specific discretization. The velocity triples of the discretized search space got assigned a curvature index describing the radius circle next to their own. The set of curvature indices were derived by using the outer cells of the discretized velocity space given by $|\vec{V}|$ and ω. As a result, the number of ICC locations on each radius circle is quite high and the locations are dense distributed. This implies that a large number of curvature indices is necessary to cover all velocity triples sufficient enough. Hence, the total number of indices is given by a combination of indices defining the radius circles and indices defining the ICC-lines. As already noted, the first one uses the set of outer cells W_r given by the window contrasting $|\vec{V}|$ and ω. The second one uses the set of outer cells W_l given by the window contrasting v_x and v_y. The total number of curvature indices is then given by $|W_r| \cdot |W_l|$.

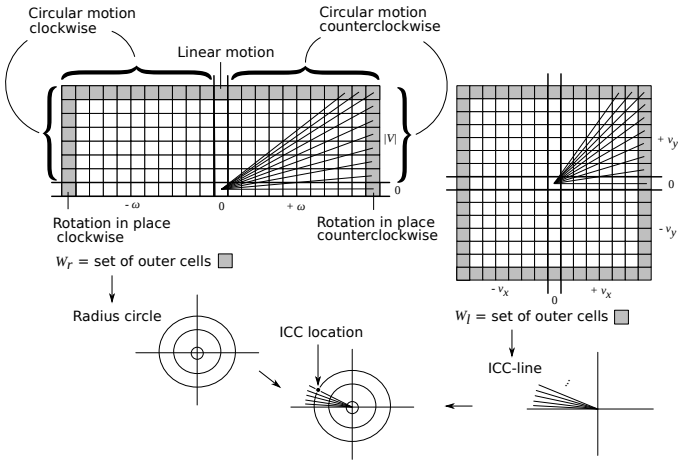

Fig. 5. The combination of the outer cells from both discretization windows serve as the curvature indices.

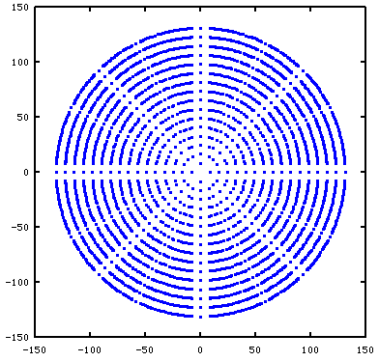

Fig. 6. The distribution of ICC locations on different radius circles.

3) Lookup table organization: The following lookup tables are used for the omni-drive extension:

- $T_{index,r}[i_{v,x}, i_{v,y}, i_\omega] \Rightarrow i_r$: Provides the curvature index i_r for a $\{v_x, v_y, \omega\}$ combination, addressed by the index values $i_{v,x}$, $i_{v,y}$ and i_ω.
- $T_{index,l}[i_{v,x}, i_{v,y}, i_\omega] \Rightarrow i_l$: Provides the curvature index i_l for a $\{v_x, v_y, \omega\}$ combination, addressed by the index values $i_{v,x}$, $i_{v,y}$ and i_ω.
- $T_{dist}[i_x, i_y, i_r, i_l] \Rightarrow d(i_r, i_l)$: Provides the remaining distance value d for the curvature index tuple $\{i_r, i_l\}$ to obstacle cell $O(x, y)$, addressed by the index values i_x and i_y.

B. Online Part

1) Search space restriction: The additional velocity component v_y expands the search space from two-dimensions to three. As a result, the classical two-dimensional dynamic window expands to a three-dimensional dynamic cuboid. The restrictions of the three velocity components depend on the current velocity ($\dot{v_x}$, $\dot{v_y}$, $\dot{\omega}$), the maximum acceleration ($\hat{a}_{v,x}$, $\hat{a}_{v,y}$, $\hat{a_\omega}$) and the time interval Δt (equation 4-6).

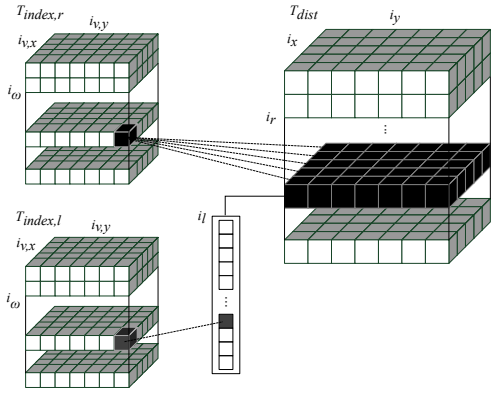

Fig. 7. The distance lookup table and the curvature index association tables of the CDL omni-drive extension.

$$DC_x = \{v_x \,|\, \dot{v_x} - \hat{a}_{v,x} \cdot \Delta t \leq v_x \leq \dot{v_x} + \hat{a}_{v,x} \cdot \Delta t\} \quad (4)$$
$$DC_y = \{v_y \,|\, \dot{v_y} - \hat{a}_{v,y} \cdot \Delta t \leq v_y \leq \dot{v_y} + \hat{a}_{v,y} \cdot \Delta t\} \quad (5)$$
$$DC_\omega = \{\omega \,|\, \dot{\omega} - \hat{a_\omega} \cdot \Delta t \leq \omega \leq \dot{\omega} + \hat{a_\omega} \cdot \Delta t\} \quad (6)$$

2) Minimum distance determination: The minimum distance must be determined for each velocity triple $\{v_x, v_y, \omega\} \in DC$. Each triple is represented by a curvature index tuple $\{i_r, i_l\}$. Now it is necessary to determine the minimum distance for each $\{i_r, i_l\} \in DC$. The associations between a velocity triple and the curvature indices i_r and i_l are made by the lookup tables $T_{index,r}$ and $T_{index,l}$. If an appropriate source provides the necessary obstacle information in a grid map, the minimum distance can be determined under consideration of all occupied obstacle cells in the cartesian space ξ (equation 7).

$$d(i_r, i_l) = \min_{\forall \{i_x, i_y\} \in \xi} T_{dist}\,[i_x, i_y, i_r, i_l] \quad (7)$$

3) Lookup for checking admissibility and evaluation: To determine whether a velocity triple is admissible, the information about the remaining distance and the maximum braking acceleration on a corresponding curvature must be known in order to get the value for the remaining braking distance. The minimum distance value is provided by $d(i_r, i_l)$ and the acceleration value by $T_{acc,V}$ respectively $T_{acc,\omega}$. Checking this condition for the corresponding curvature of a velocity triple allows to determine the admissibility of this combination. If the velocity triple is admissible, it represents a potential motion command for the next time step and it can be finally evaluated by an objective function to determine the benefit of the resulting maneuver. Different heuristics can be implemented to rank the velocity triples in such a way that a desired driving behavior is achieved. The experiment section provides two example heuristics for two different scenarios.

C. Resource Consumption (Offline Part)

This section gives an overview about the resource consumption of the omni-drive extension which is highly dependent on the number of used curvature indices. Equation 8

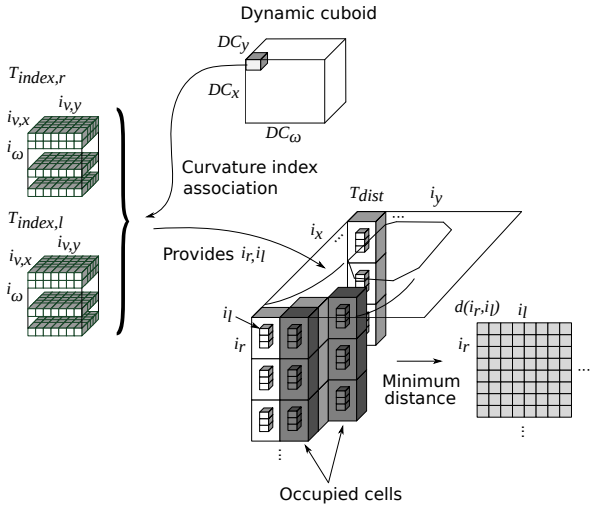

Fig. 8. Online part of the CDL extension: Step 1 - Determine minimum distance.

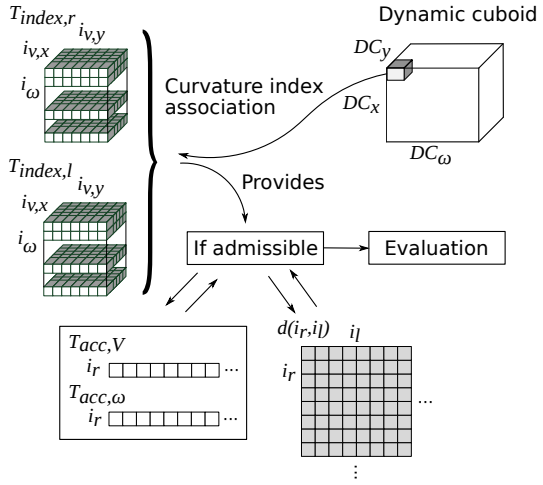

Fig. 9. Online part of the CDL extension: Step 2 - Lookup for checking admissibility and evaluation.

and 9 show the number of resulting curvature indices when using the outer cells of the discretization windows given by the example configuration from table IV. Based on these values, equation 10 determines the size of the lookup table depending on the number of cells in the cartesian space $(n_x \cdot n_y)$ which is given by the ratio of the cartesian space size and the cell discretization. To give an impression, table II shows the resource consumption of the offline part depending on the number of used curvature indices and the number of cells in the cartesian space. The time values were obtained with an Intel Core i7-3610QM processor and the distance values were stored as shorts (2 bytes). It can be seen that using the full number of outer cells as curvature indices results in a very large lookup table size (first two rows). However, a reduction of the ICC-line indices by at least half (last two rows) seems to be reasonable, since the resulting deviations of the computed distance values can be neglected. Note that this would correspond to $\kappa = 2$ for the ICC-line indices when considering equation 9. Hence κ is denoted as accuracy reduction factor if we want to keep the current velocity discretization but reduce the lookup table size at the expense of the accuracy in the computed distance values. Also note that the size of the lookup table is doubled because the arc distance and the angular distance are always stored separately. This could be optimized by only storing the angular distance values and computing the arc distance values online. In this case the lookup table size would be only half as large as given in table II.

A further possibility to reduce the lookup table size significantly is to provide only partial coverage and to compute no distance values for motion type 8 where $(v_x \wedge v_y \wedge \omega) \neq 0$ applies. Table III shows the resource consumption for two different numbers of obstacle cells with only such a limited coverage. It is obvious that this corresponds to a strong reduction of the memory consumption. However, in this case the search space is not a real cuboid and the commanding of a motion combination where $(v_x \wedge v_y \wedge \omega) \neq 0$ applies must be generally prohibited during the online phase which restricts the maneuverability of the robot. Note that for this partial coverage, $|W_r|$ is given by $(2v_s^{-1}\hat{v}) + (2v_s^{-1}|\check{v}|) + (2\omega_s^{-1}\hat{\omega}) + (2\omega_s^{-1}|\check{\omega}|)$ when we assume that $\hat{v} = \hat{v_x} = \hat{v_y}$ and $\check{v} = \check{v_x} = \check{v_y}$.

$$|W_r| = \left\lceil \kappa^{-1} \left(2 \cdot \left\lfloor \frac{\sqrt{\hat{v_x}^2 + \hat{v_y}^2}}{v_s} \right\rfloor + 2 \cdot \frac{\hat{\omega}}{\omega_s} + 1 \right) \right\rceil \quad (8)$$

$$|W_l| = \left\lceil \kappa^{-1} \left(2 \cdot \frac{\hat{v_x}}{v_s} + 2 \cdot \frac{|\check{v_x}|}{v_s} + 2 \cdot \frac{\hat{v_y}}{v_s} + 2 \cdot \frac{|\check{v_y}|}{v_s} \right) \right\rceil \quad (9)$$

$$S = |W_r| \cdot |W_l| \cdot \frac{s_x}{s_c} \cdot \frac{s_y}{s_c} \cdot sizeof(datatype) \cdot 2 \quad (10)$$

TABLE II. RESOURCE CONSUMPTION OF THE OFFLINE PART FOR THE EXAMPLE CONFIGURATION GIVEN IN TABLE IV.

$\|W_r\| \cdot \|W_l\|$	$n_x \cdot n_y$	Time (sec.)	Size (MB)
$423 \cdot 800$	$61 \cdot 101$	1 668.06	7 953.2
$423 \cdot 800$	$101 \cdot 101$	2 740.39	13 168.41
$423 \cdot 400$	$61 \cdot 101$	857.85	3 976.6
$423 \cdot 400$	$101 \cdot 101$	1 462.85	6 584.2

TABLE III. RESOURCE CONSUMPTION OF THE OFFLINE PART WITH PARTIAL COVERAGE.

$\|W_r\| + \|W_r\| + \|W_l\|$	$n_x \cdot n_y$	Time (sec.)	Size (MB)
$680 + 680 + 800$	$61 \cdot 101$	11.46	41.36
$680 + 680 + 800$	$101 \cdot 101$	18.86	68.49

TABLE IV. EXAMPLE CONFIGURATION.

Name	Value	Meaning
$\check{v_x}, \hat{v_x}$	-1000 mm/s, +1000 mm/s	Min/max x-trans. vel.
$\check{v_y}, \hat{v_y}$	-1000 mm/s, +1000 mm/s	Min/max y-trans. vel.
$\check{\omega}, \hat{\omega}$	-70 deg/s, +70 deg/s	Min/max rot. vel.
v_s, ω_s	10 mm, 1 deg	Trans./rot. discretization
s_x, s_y	5050 mm (3050 mm), 5050 mm	Cartesian space (cs) size
s_c	50 mm	Discretization of cs
n_x, n_y	101 (61), 101	Nr. cells in cs
$\hat{a}_{v,x}, \hat{a}_{v,y}$	400 mm/s^2, 400 mm/s^2	Max trans. acceleration
$\hat{a_\omega}$	30 deg/s^2	Max rot. acceleration
Δt	0.7 s	Time interval

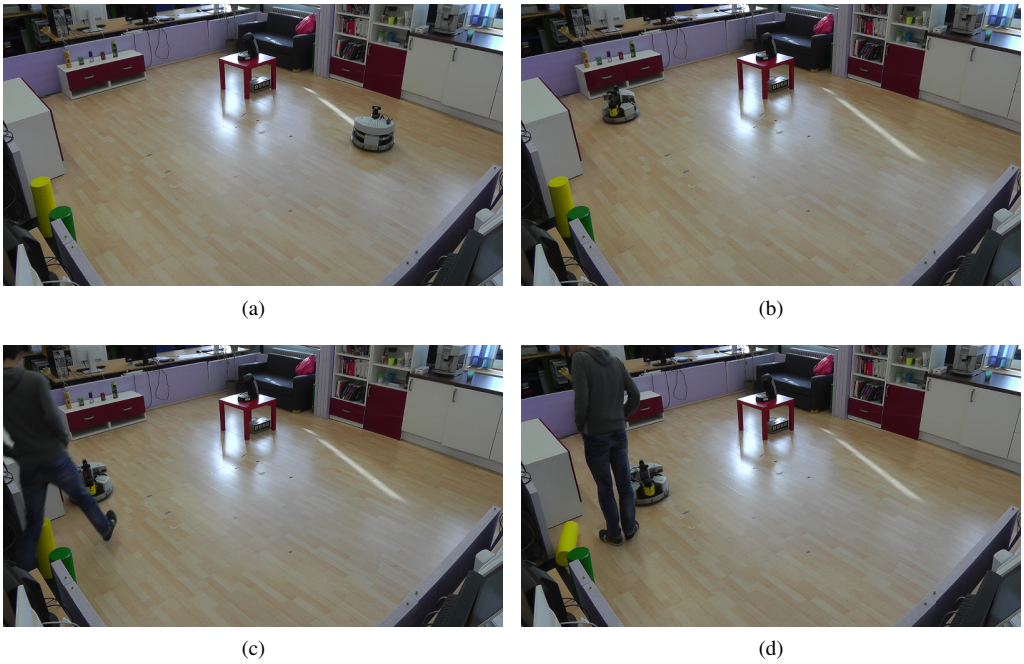

Fig. 10. A reactive scenario: (a)-(c) The robot traverses the room with classic 2-DOF motions (v_x, ω). (d) The robot avoids the suddenly appearing obstacle by a lateral movement. Such a motion is commanded if avoiding by a non-straight 2-DOF motion is no longer possible due to limited rotational dynamics.

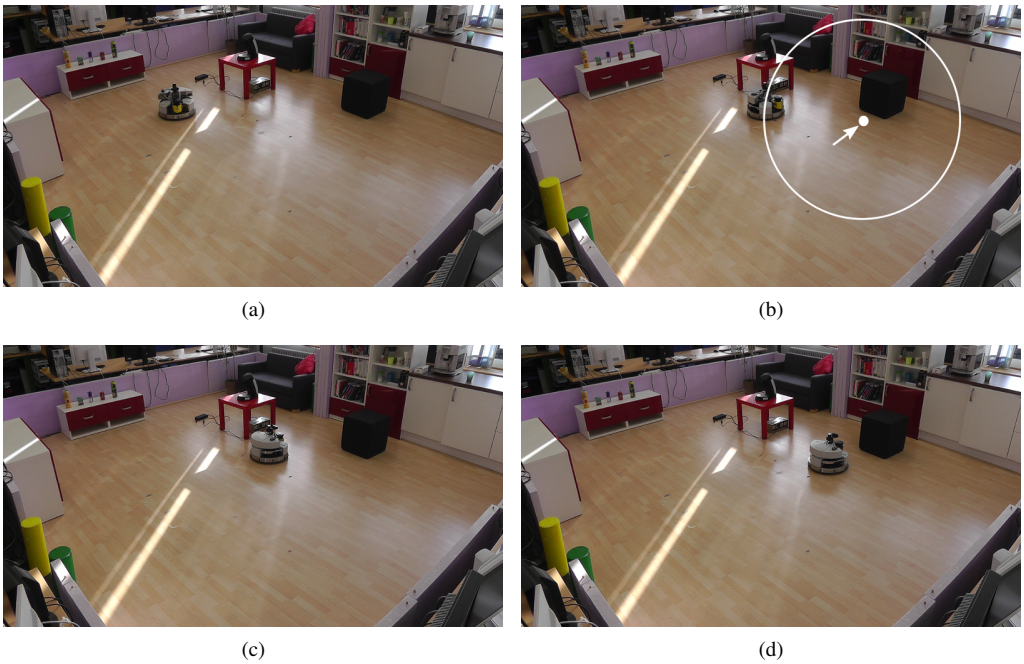

Fig. 11. A docking scenario: (a) The robot approaches the goal by arbitrary motions. (b) The robot has reached the docking area and should now rotate until the desired docking orientation is reached (indicated by the white arrow). (c) Desired docking orientation reached. (d) The robot approaches the goal by lateral motions.

VI. Experiments

The experiments were tested in real world and in simulation with the Robotino® 3 platform. Both partial coverage and full coverage lookup table versions were tested. Note, that the online performance does not depend on the number of used curvature indices but on the size of the velocity search space, the size of the cartesian space and the current obstacle density. The execution cycles of the CDL omnidrive extension running on the Robotino® 3 platform were fast enough to drive appropriate in the real world scenarios with the configuration given in table IV. In a simulation environment, when the motion execution runs on one core of an Intel Core i7-3610QM processor, the CPU utilization differs from below 10% (dynamic window) to 40% - 70% (dynamic cuboid) depending on the current environment. Note, that the used laser scanner provides an opening angle of 240 degrees. When using

a second laser scanner to achieve a view of 360 degrees, the execution cycle times will increase.

A. Reactive Scenario

Figure 10 shows a reactive heuristic without a given goal location. In order to keep it simple, classic 2-DOF motions where $v_y = 0$ applies are generally preferred and get a special bonus. From this set, linear motions with high velocities and curvatures with a large remaining distance are preferred. Equation 11 - 13 show the corresponding evaluation scheme for a $\{v_x, v_y, \omega\}$ triple which is represented by the curvature index tuple $\{i_r, i_l\}$. Note that ρ, σ and τ are normalization constants.

$$V_{heading} = 1 - (\rho^{-1}|0 - \omega|) \tag{11}$$

$$V_{speed} = 1 - (\sigma^{-1}(\hat{v} - (v_x^2 + v_y^2)^{0.5})) \tag{12}$$

$$V_{distance} = \tau^{-1} d(i_r, i_l) \tag{13}$$

The resulting driving behavior is shown in figure 10. Since 2-DOF motions are preferred, there is no need for the robot to make lateral motions at first. Hence, the room is traversed with linear motions along the wall until the remaining forward distance becomes small enough such that a curvature with a corresponding rotational value gets the preference in order to approach the next wall perpendicular to the current. However, in cases where no appropriate rotational velocity value can be selected, a lateral motion can help to prevent the robot from rotating in place. This situation can occur if a dynamic obstacle appears suddenly in front of the robot as shown in figure 10(c).

B. Docking Scenario

A simple algorithm for a docking scenario is formulated as follows:

1) A free region within a certain threshold around the final goal position (docking area) must be reached. In this step all movement types are allowed.
2) If the docking area is reached, the robot should rotate until a desired heading is achieved that is necessary to approach the goal with the correct orientation using linear motions.
3) If the desired heading is reached, the docking phase is applied. There, the robot approaches the final goal using only linear motions.

Figure 11 illustrates the robot behavior for such a procedure. Note that the overall navigation architecture [3] uses a path planner that generates intermediate waypoints depending on a given goal location and based on a-priori informations about the environment. The waypoints are finally approached by the objective function of the local obstacle avoidance method.

VII. Conclusion

An extension of the dynamic window approach to three dimensions (dynamic cuboid) was provided in this paper. The additional dimension allows the robot to drive omnidirectional (3-DOFs) through an unknown and dynamic environment. In addition to kinematic and dynamic considerations, the presented approach also considers the shape of the robot by using precalculated lookup tables providing the distance values for each curvature to each obstacle cell of the cartesian space. This procedure corresponds to an extension of the classical CDL approach for 2-DOFs which was so far extensively used in several real world robotic applications. Hence, the new omni-drive support extends this proven method by providing a significantly higher maneuverability. However, if available memory resources are very limited, an optimal trade-off configuration between maneuverability and memory consumption must be found. Note that an implementation of the CDL omni-drive extension can be found in our repository [14].
Future work deals with further optimization of the lookup table size and with an implementation of more powerful heuristics for complex scenarios taking into account the benefit of omnidirectional motions.

Acknowledgment: The research work presented in this paper is partially supported by the EU FP7 grant ECHORD++ MARS No. 601116.

References

[1] C. Schlegel, "Fast local obstacle avoidance under kinematic and dynamic constraints for a mobile robot," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, vol. 1, Victoria, Canada, 1998, pp. 594–599.

[2] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance." *IEEE Robotics and Automation*, vol. 4, no. 1, 1997.

[3] C. Schlegel, "Navigation and execution for mobile robots in dynamic environments: An integrated approach," dissertation, Universität Ulm, 2004. [Online]. Available: http://www.hs-ulm.de/users/cschlege/_downloads/phd-thesis-schlegel.pdf

[4] Service Robotics Ulm, "Youtube channel," http://youtube.com/user/RoboticsAtHsUlm.

[5] J. Borenstein, Y. Koren, and S. Member, "The vector field histogram - fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 278–288, 1991.

[6] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[7] J. Minguez and L. Montano, "Nearness diagram navigation (nd): A new real time collision avoidance approach," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000, pp. 2094–2100.

[8] I. Ulrich and J. Borenstein, "Vfh+: Reliable obstacle avoidance for fast mobile robots," in *Proc. IEEE International Conference on Robotics and Automation*, 1998.

[9] W. Feiten, R. Bauer, and G. Lawitzky, "Robust obstacle avoidance in unknown and cramped environments," in *Proc. IEEE International Conference on Robotics and Automation*, 1994, pp. 2412–2417.

[10] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *Proc. of the IEEE International Conference on Robotics and Automation*, 1996, pp. 3375–3382.

[11] N. Y. Ko and R. Simmons, "The lane-curvature method for local obstacle avoidance," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 1998, pp. 1615–1621.

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

[13] J. Minguez and L. Montano, "Extending collision avoidance methods to consider the vehicle shape, the kinematics, and dynamics of a mobile robot," *IEEE Transaction on Robotics*, vol. 25(2), pp. 367–381, 2009.

[14] Service Robotics Ulm, "Website," http://www.servicerobotik-ulm.de.