2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

# Generic Trajectory Representation and Trajectory Following For Wheeled Robots

Morten Kjærgaard[1], Nils A. Andersen[1] and Ole Ravn[1]

*Abstract*— **This article presents the work towards a purely generic navigation solution for wheeled mobile robots motivated by the following goals: Generic: Works for different types of robots. Configurable: Parameters maps to geometric properties of the robot. Predictable: Well defined where the robot will drive. Safe: Avoid fatal collisions.**

**Based on a survey of existing methods and algorithms the article presents a generic way to represent constraints for different types of robots, a generic way to represent trajectories using Bëzier curves, a method to convert the trajectory so it can be driven in a smooth motion, a method to create a safe velocity profile for the robot, and a path following controller.**

## I. INTRODUCTION

The most fundamental task for wheeled robots is navigation. In the simplest form the task consist of moving the robot from one position to another. In practice, navigation includes quite complex situations, such as deciding a path, following it safely, detecting and handling unexpected obstacles or following a moving target. In addition, the navigation algorithm must take the geometric and dynamic constraints of the robot into account.

Due to this large complexity, the navigation task is usually divided into a set of subproblems such as localizing, trajectory planning, trajectory following, obstacle avoidance, and learning maps.

Mobile robot navigation has been the subject of research for decades. An early example is a team at the LAAS laboratory in Toulouse, France who already in 1977 investigated the design and control of mobile robots and built the Hilare robot. The robot platform was the base for early work in autonomous navigation and obstacle avoidance such as [Cha82][Mor80].

Today, almost 30 years later, many consider the planning, mapping and navigation problems for indoor robots somehow solved, and have moved onto more challenging research areas such as robot swarms and aerial robots. Advancements in robotics, sensor technology and computer hardware, means that much more advanced algorithms based on probability theory is used for navigation today. Implementations for localization and mapping algorithms such as FastSlam [Hae+03] or GMapping [GSB05][GSB06] are available at *openslam.org*.

Yet, no generic and ready-to-use implementation of robot navigation for an arbitrary wheeled robot is available. The scope of this work is to provide a generic navigation implementation for Ackermann and differential drive robots.

[1]Department of Electrical Engineering, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark `mkj,naa,or@elektro.dtu.dk`

## II. GENERIC CONSTRAINT MODELING

The kinematics constraints of both an Ackermann and a differential drive robot does not allow for sideways motion. It is assumed that no skid is taking place, thus $\dot{y} = 0$.

Therefore the robot velocity can be expressed using only a forward and an angular velocity $(\dot{x}, \dot{\theta})^T$. With such a velocity the robot will perform a curved motion corresponding to following a circle of radius $r$ given by (1). The inverse of the circle radius is the curvature $\kappa$ defined in (2).

$$r = \frac{\dot{x}}{\dot{\theta}} \tag{1}$$

$$\kappa = \frac{\dot{\theta}}{\dot{x}} \tag{2}$$

### A. Polar Representation of Velocity Vector

As seen from (2), the curvature $\kappa$ has the unfortunate property of becoming infinite when the robot is rotating on the spot and the forward velocity is zero. Rotation on the spot is a perfectly valid motion for a differential drive robot. To overcome this limitation, the velocity vector of $(\dot{x}, \dot{\theta})^T$ is instead represented in polar coordinates $(\rho, \phi)^T$.

$$\rho = \sqrt{\dot{\theta}^2 + \dot{x}^2} \tag{3}$$

$$\phi = \operatorname{atan2}(\dot{\theta}, \dot{x}) \tag{4}$$

While the value of $\rho$ is independent of the curvature of the robot motion, the value of $\phi$ relates directly to it with the following expression:

$$\kappa = \tan(\phi) \tag{5}$$

### B. Driving Velocity Constraint

The maximum velocity of the robot will usually be artificially limited due to safety concerns or dynamic properties. Assuming the maximum forward and reverse velocity of the robot is constrained by $\dot{x}_{max,fwd}$ and $\dot{x}_{max,rvs}$ respectively, as defined in (6)-(8). Figure 1 illustrates how this constraint affects the available velocity space.

$$-\dot{x}_{max,rvs} \leq \dot{x} \leq \dot{x}_{max,fwd} \tag{6}$$

$$\dot{x}_{max,rvs} \geq 0 \tag{7}$$

$$\dot{x}_{max,fwd} \geq 0 \tag{8}$$

Given these limits, the maximum allowed magnitude of $\rho$, during a motion with a given curvature represented by $\phi$ can be found with the function in (9).

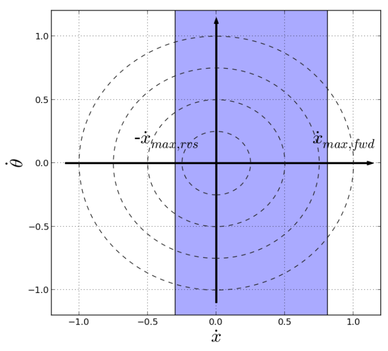

Fig. 1. Example of how the driving velocity constraint affects the available velocity space. $\dot{x}_{max,fwd} = 0.8\frac{m}{s}$ and $\dot{x}_{max,rvs} = 0.3\frac{m}{s}$

$$f_{\rho_{\max,\text{drive}}}(\phi) = \begin{cases} \frac{\dot{x}_{max,fwd}}{\cos\phi} & \text{when } \phi \in [-\frac{\pi}{2}; \frac{\pi}{2}] \\ \frac{\dot{x}_{max,rvs}}{\cos\phi} & \text{otherwise} \end{cases} \quad (9)$$

### C. Centrifugal Force Constraint

The robot will be exposed to a centrifugal force acting sideways on the robot when it is driving in a curve. This force can make the wheels slide sideways, or even worse make the robot fall over or drop a load it is carrying.

The maximum centrifugal force the robot can handle depends on many factors such as the wheels, floor friction or the weight of the current load, and it is assumed that the maximum value has been determined. The centrifugal acceleration $a_c$ acting on a robot with a given velocity vector is:

$$a_c = \dot{x}\,\dot{\theta} \quad (10)$$

The maximum allowed magnitude of $\rho$, during a motion with a given curvature represented by $\phi$ can be found with the function in (11).

$$f_{\rho_{max},a_c}(\phi) = \sqrt{\frac{2a_c}{\sin(2\phi)}} \quad (11)$$

Figure 2 illustrates how the centrifugal force constraint affects the available velocity space.

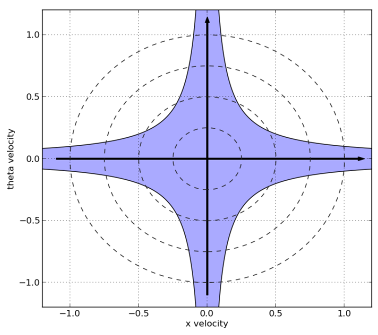

Fig. 2. Example of how the centrifugal force constraint affects the available velocity space. $a_c = 0.1\frac{m}{s^2}$

### D. Steering Angle Constraint (Ackermann)

Due to mechanical constraints an Ackermann wheel configuration will have a limited steering range defined as: $\beta \in [-\beta_{max}, \beta_{max}]$. The steering angle constraint results in a range of curvatures that the robot is incapable of following. It is independent on the driving speed of the robot, thus it cannot be obeyed simply by limiting the driving speed of the robot. Instead it should be considered at the planning level, only creating trajectories with feasible curvatures.

If $L$ denotes the distance between the front and rear axis, the constraint maps into limits for $\rho$ using the following function:

$$f_{\rho_{\max},\text{steering}}(\phi) = \begin{cases} 0 & \text{when } \phi > \phi_{\beta,max} \\ \infty & \text{when } \phi \leq \phi_{\beta,max} \end{cases} \quad (12)$$

$$\phi_{\beta,max} = \tan^{-1}\left(\frac{\tan(\beta_{max})}{L}\right) \quad (13)$$

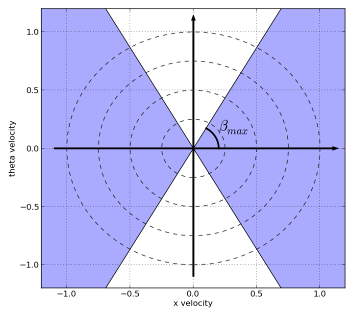

Fig. 3. Example of how the steering angle constraint affects the available velocity space.

### E. Drive Wheel Constraint (Differential Drive)

For a differential drive robot each of the drive wheels are controlled individually. Define the maximum allowed angular velocity of each wheel as $\varphi_{max}$.

$$-\varphi_{max} \leq \quad \varphi_r \quad \leq \varphi_{max} \tag{14}$$
$$-\varphi_{max} \leq \quad \varphi_l \quad \leq \varphi_{max} \tag{15}$$

If $r$ denotes the radius of the drive wheels, and $d$ denotes half the distance between them, then the drive wheel constraint limits the magnitude of $\rho$ given by (16).

$$f_{\rho_{\max},\text{wheel}}(\phi) = \text{abs}\left(\frac{r\,\varphi_{max}}{d\,\sin\phi + \cos\phi}\right) \tag{16}$$

Figure 4 shows an example of how this constraint limits the velocity space of a robot.

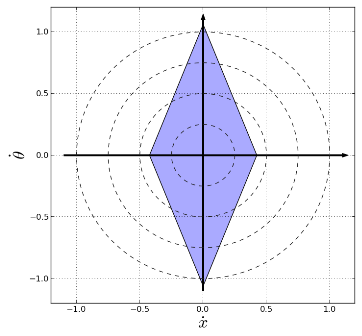

Fig. 4. Example of how the drive wheel constant limits the available velocity space.

### F. Joint Constraint Function

Given a specific robot type, all these constraints can be combined into a single constraint function. Below is a combined constraint function for a differential drive robot (17) and for an Ackermann robot (18).

$$f_{\rho_{\max},\text{differential}}(\phi) = \min \begin{cases} f_{\rho_{\max},\text{drive}}(\phi) \\ f_{\rho_{max},a_c}(\phi) \\ f_{\rho_{\max},\text{wheel}}(\phi) \end{cases} \tag{17}$$

$$f_{\rho_{\max},\text{Ackermann}}(\phi) = \min \begin{cases} f_{\rho_{\max},\text{drive}}(\phi) \\ f_{\rho_{max},a_c}(\phi) \\ f_{\rho_{\max},\text{steering}}(\phi) \end{cases} \tag{18}$$

## III. TRAJECTORY REPRESENTATION

The simplest form for trajectory segments consists of straight line segments between a series of intermediate poses. Many path planners for mobile robots such as the one found in [Lik+05] returns a path based on straight line segments. Straight lines can only interpolate the position between the poses, not the orientation. Rotation only happens in the actual knot points, thus following such a trajectory requires stopping in each knot point.

In [FBT97] the trajectory is modeled with circular arcs, where each segment has a constant curvature. Such a trajectory appears smooth, but all the knots that connect segments of different curvature will have non-continuous curvature.

The authors in [MF07] and [Sah+07] use cubic Bézier splines to model the trajectory. These can be adjusted to provide first order continuity (velocity) in the knots. Curvature also depends on the second order derivative (acceleration), thus curvature continuity cannot be guaranteed using cubic Bézier splines.

The authors in [LSB09] uses fifth order Bézier splines to model the trajectory segment. They present a method and a series of heuristics that makes it possible to match both the first and second order derivative in the knots, by only adjusting the two neighboring segments. The beginning trajectory is a sparse series of positions without orientation that are found to be collision-free when connected by a straight line. The method iteratively optimizes the trajectory into a curved trajectory, while not straying too much from the straight line shape.

The proposed method in this paper adopts part of the method from [LSB09], with some modifications to increase generality:

(1) Extended to be able to model rotation-on-the-spot segments. (2) Input is a collision-free, possibly curved trajectory with pose information continuous in position and orientation. (3) Knots are curvature-matched without the need for the second derivative to be the same value. (4) Allow knots to be non-contious in curvature when a full stop is required anyway.

## IV. CREATING SMOOTH CURVATURE

The curvature should generally be continuous along the trajectory for the robot to be able to follow it in a smooth motion. There are two special situations where continuity is not required: (1) The knot connecting a segment with forward motion, and a segment with reverse motion. The robot is at a complete halt in the knot, so continuous curvature is not required. (2) A rotation-on-the-spot (ROTS) segment will have infinite curvature, thus a connecting segment will be non-continuous. The robot is required to come to a complete halt in the knot. Table 5 lists the different types of segments and whether they must be connected by continuous curvature or by putting the robot in a complete halt.

| | **Forward** | **Reverse** | **ROTS** |
|---|---|---|---|
| **Forward** | Continuous | Halt | Halt |
| **Reverse** | ▨ | Continuous | Halt |
| **ROTS** | ▨ | ▨ | Continuous ($\infty$) |

Fig. 5. Connecting Knots

This article proposes a method to convert a trajectory consisting of 3rd order Bézier curves with continuous position and orientation, into a trajectory of 5th order Bézier curves with continuous curvature.

### A. Initial Trajectory

It is assumed that the initial trajectory has been found and is:

- Collision Free.
- Continuous in the position and orientation.

- Modeled as 3rd order Bézier curves. (Except ROTS segments)

The scope is not to find the actual trajectory, it is assumed to be planned beforehand. If the initial curve is not modeled with 3rd order Bézier curves, literature within computer graphics provides methods for generating an approximation with 3rd order Bézier curves [Gla90].

The initial trajectory, denoted $Q^{(3)}$ (19), consists of a series of segments $\hat{Q}_i^{(3)}$. Each segments is either a curve segment, modeled by the four points describing a 3rd order Bézier curve (20a), or a ROTS segment, modeled as a point, a start and an end orientation (20b). The superscript "(3)" denotes that it is using third order Bézier curves.

$$Q^{(3)} = < \hat{Q}_0^{(3)}; \hat{Q}_1^{(3)}; \ldots ; \hat{Q}_n^{(3)} > \tag{19}$$

$$\hat{Q}_i^{(3)} = \begin{cases} < P_{i,0}^{(3)}; P_{i,1}^{(3)}; P_{i,2}^{(3)}; P_{i,3}^{(3)} > & \text{Curve (a)} \\ < P_i; \theta_{i,0}; \theta_{i,1} > & \text{ROTS (b)} \end{cases} \tag{20}$$

### B. Curvature Matching Method

The method from [LSB09] matches two segments by setting both the first and second order derivative to the same value. Because the initial trajectory is continuous in the orientation, the direction of the first derivative is continuous in the knots, while the magnitude may be different. The velocity at which the trajectory is driven is independent of this magnitude, so this discontinuity is not a problem. In this method, the value of the first derivative is conserved, and only the second derivative is modified to curvature-match the two segments. This maintains the shape of the original curve better.

Denote the curvature in the start-point of segment $\hat{Q}_i$ as $\kappa_i^{begin}$, and similar in the end-point of segment $\hat{Q}_{i-1}$ as $\kappa_{i-1}^{end}$. An average value for the curvature is found by a weighted average, inverse proportional to the length $l$ of each of the segments respectively. The weighted average is used because the second derivative of longer segments can be adjusted more without too much difference in the original shape.

$$\kappa_{avg} = \frac{l_{i-1}\, \kappa_i^{begin} + l_i\, \kappa_{i-1}^{end}}{2(l_{i-1} + l_i)} \tag{21}$$

The value of the second derivative of the two initial segments are denoted $\mathbf{a}_{i-1}^{end}$ and $\mathbf{a}_i^{begin}$. Since the second derivative is a two dimensional vector, first a unit vector for the average direction is found by:

$$\mathbf{a}_{unit} = \frac{\mathbf{a}_{i-1}^{end} + \mathbf{a}_i^{begin}}{||\mathbf{a}_{i-1}^{end} + \mathbf{a}_i^{begin}||} \tag{22}$$

The length of the second derivative that will result in a curvature of $\kappa_{avg}$ is found by the following expression:

$$a_{len} = \frac{||\mathbf{v}||^2\, \kappa_{avg}}{\frac{\mathbf{v}}{||\mathbf{v}||} \cdot \mathbf{a}_{unit}} \tag{23}$$

where $\mathbf{v}$ denotes the first derivative in the respective end-point.

### C. Continuous-Curve Trajectory

The continuous curvature trajectory $Q^{(5)}$ is similar to the initial counterpart, except the curved segments is modeled by six points describing a 5th order Bézier curve. The ROTS segments are not changed.

$$Q^{(5)} = < \hat{Q}_0^{(5)}; \hat{Q}_1^{(5)}; \ldots ; \hat{Q}_n^{(5)} > \tag{24}$$

$$\hat{Q}_i^{(5)} = \begin{cases} < P_{i,0}^{(5)}; \ldots ; P_{i,5}^{(5)} > & \text{Curve (a)} \\ < P_i; \theta_{i,0}; \theta_{i,1} > & \text{ROTS (b)} \end{cases} \tag{25}$$

The two control points in each end of the Bézier curve is selected to imitate the initial curve as closely as possible.

$$P_0^{(5)} = P_0^{(3)} \tag{26}$$

$$P_1^{(5)} = \frac{2}{5} P_0^{(3)} + \frac{3}{5} P_1^{(3)} \tag{27}$$

$$P_4^{(5)} = \frac{3}{5} P_2^{(3)} + \frac{2}{5} P_3^{(3)} \tag{28}$$

$$P_5^{(5)} = P_3^3 \tag{29}$$

$$\tag{30}$$

With this procedure the value of the first derivative in the end points is not changed.

The two middle control point can be placed to control the value of the second derivative denoted $\mathbf{a}_0$ and $\mathbf{a}_1$ in the two end points.

$$P_2^{(5)} = \frac{1}{20}\mathbf{a}_0 + 2P_1^{(5)} + P_0^{(5)} \tag{31}$$

$$P_3^{(5)} = \frac{1}{20}\mathbf{a}_1 + 2P_4^{(5)} + P_5^{(5)} \tag{32}$$

Using the above expression all the segments in the initial trajectory is converted into a curvature-continuous set of fifth order Bézier segments.

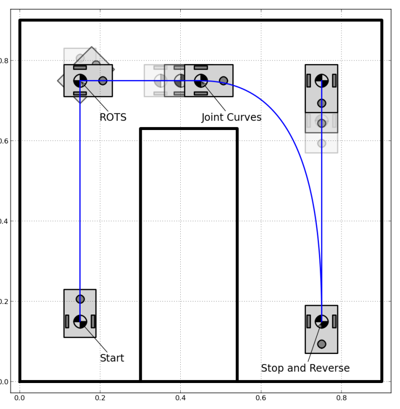

Fig. 6. Example Trajectory 1

### D. *Example Trajectory 1*

Figure 6 shows a trajectory for a differential drive robot. The trajectory is represented by third order Bézier segments and one ROTS segments, thus includes all the possible types of segments and knots.

The curvature of the initial trajectory is illustrated in figure 7 as a blue curve. The knot connecting the second forward segment and the curved segment is the only one where the robot does not have to come to a complete halt, but from the figure one can see that the curvature is non-continuous in this knot. Using the proposed method, the trajectory is modified to apply continuous curvature in this knot by modifying the two connecting segments. The curvature of the modified trajectory is illustrated by the green line in the figure, where one can see how the two connecting segments have been modified, and the curvature is now continuous.

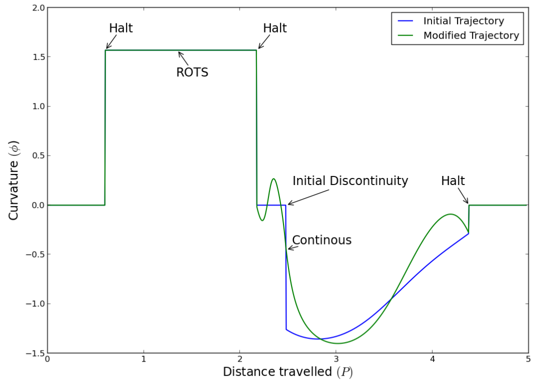

Fig. 7. Curvature of Example Trajectory 1

### E. *Example Trajectory 2*

Figure 8 shows a trajectory for an Ackermann robot driving slalom around a series of round obstacles. The last turn is too sharp to perform in a simple motion due to the steering angle constraint. Instead the robot stops, reverses shortly, and continues around the obstacle.

The blue line in the figure illustrates the initial trajectory defined by 3rd order Bézier curves. The curvature of the initial trajectory is illustrated by the blue curve in figure 9. From the figure one can see the trajectory has several places with non-continuous curvature.

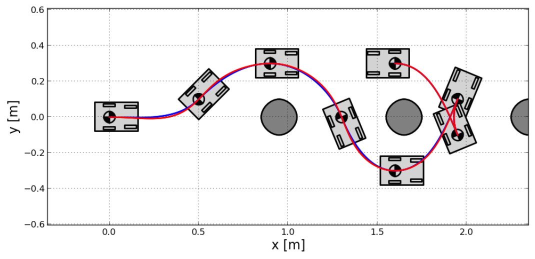

Fig. 8. Example Trajectory 2

Therefore the above method is used to convert the trajectory into a continuous curvature version with 5th order Bézier curves. The red line in figure 8 shows the modified trajectory, and how it has been slightly changed.

The green curve in figure 9 illustrates the curvature for the modified trajectory. It shows that is continuous, except in the two places where the robot changes between forward and reverse motion and is at a complete halt.

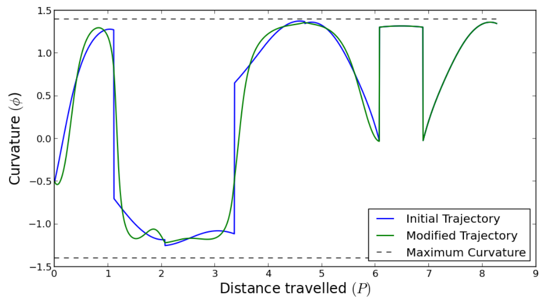

Fig. 9. Curvature of Example Trajectory 2

## V. VELOCITY PROFILE

The trajectory so far is purely geometrical. It has been modified to be able to drive with only the required stops, but it contains no information about the actual speed the robot should have.

When following the trajectory, the robot must obey the velocity and acceleration constraints, and also brake in proper time before full stops and curves. To be able to handle this, the trajectory is analyzed in a three step process, resulting in a velocity profile (33) of the maximum allowed velocity $\rho_{max}$ the robot can have at any location $P$ on the trajectory. The maximum velocity is defined as a maximum length $\rho$ of the velocity vector (3), thus limiting both the translational and the rotational velocity.

$$\rho_{max} = f_{vel}(P) \tag{33}$$

For simplicity, each trajectory segment is split into 20 sub-segments in which constant acceleration is assumed. A maximum velocity is calculated for each boundary point between two sub-segment. The three step process is:

1) Find the maximum velocity based on only the curvature at each boundary point, using the constraint function (17) or (18).
2) Iterate forward through the trajectory. Find the maximum obtainable velocity from the previous boundary point and using maximum acceleration. Limit it to the value from step 1.
3) Iterate backwards through the trajectory. Find the maximum allowed velocity limited by the future sub-segment boundary and with maximum deceleration. Limit it to the value from step 1.

## A. Example Trajectory 1

The velocity profile for trajectory 1 in the previous section has been calculated using the above method. The velocity profile after step 1 is illustrated by the red curve in figure 10. It shows how it follows the magnitude of the curvature. The reason why the value of $p$ is high in the areas with high curvature is, that it also has a $\dot{\theta}$ component that makes it larger even if the forward velocity $\dot{x}$ is lower. The green curve shows the profile after step 3, where the velocity is limited before and after the robot has to drive slowly or come to a complete halt.

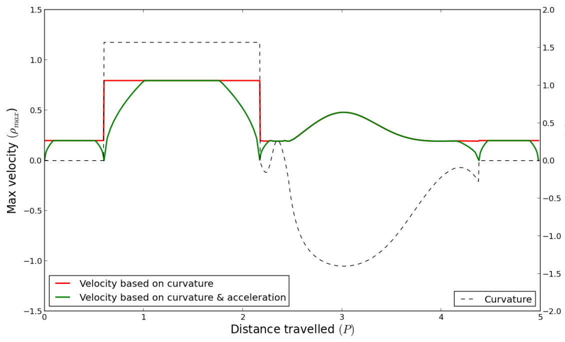

Fig. 10. Velocity Profile for Example Trajectory 1

## B. Example Trajectory 2

The velocity profile for trajectory 2 is illustrated in figure 11 and shows a similar behavior. The maximum velocity after step 1 follows the magnitude of the curvature, since it is purely derived from this curvature. The velocity profile after step 3 now has a few places where the robot is configured to drive slower, including the beginning acceleration, the end stop, and the three-point turn.

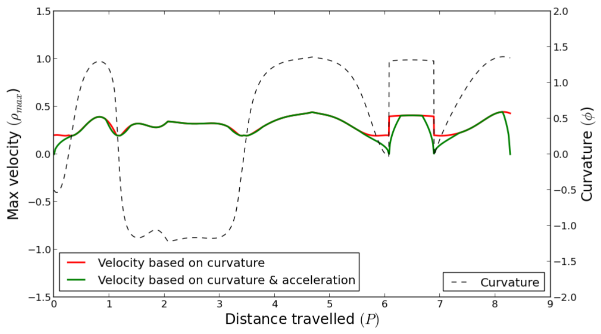

Fig. 11. Velocity Profile for Example Trajectory 2

## VI. TRAJECTORY FOLLOWING CONTROLLER

Trajectory following controllers are addressed in a lot of literature. A linear control law that drives a robot towards a certain reference pose, including the orientation, is found in [SNS11]. Based on the position error $\rho$, the angle towards the target position $\alpha$ and the orientation error $\beta$ the control laws in (34) and (35) drives the robot towards the goal position.

$$\dot{x} = k_\rho \, \rho \tag{34}$$
$$\dot{\theta} = k_\alpha \, \alpha + k_\beta \, \beta \tag{35}$$

A different approach is found in [Mic+93] where the authors derive a control law for unicycle-type and two-steering-wheels robots. It controls the angular velocity $\dot{\theta}$ to make the robot follow the path. The error input is the current sideways distance to the path, and assumes that the forward velocity $\dot{x}$ is decided elsewhere. This controller is extended by [L L03] taking modeling uncertainties into account, and using a virtual target that moves with a predefined velocity along the path as controller reference point. In [GK08] the control law is used to guide a wheelchair, where smooth and comfortable motions are important.

[VAP08] presents a set of control laws originally designed for guiding marine vehicles in formation, but in [CAG08] it is generalized for wheeled robots. One controller moves the virtual target along the trajectory to make sure the vehicle can keep up, and the second guides the vehicle towards the virtual target. A noticeable difference compared to the earlier control law is that input to the second controller includes the current velocity of the virtual frame. The control law corrects only the positional error.

Unfortunately, none of the trajectory following controllers handles the situation where the robot is reversing. In addition, even if they are designed to handle curved trajectories, most only use the position error, and thus cannot handle rotation on the spot situations. To support a generic navigation implementation, and to perform the experimental tests in this article, a control law with both of these properties was needed.

Fortunately most of the control laws adhere to the same interface. They take the current robot pose and a moving virtual reference target, and output a desired robot velocity.

Therefore the solution was to design a simple control law that supported reversing and rotation on the spot, and use it to show that a robot can follow a trajectory based on the calculated velocity profile. It should have the following properties:

1) It should correct both the position and orientation errors.
2) Be simple and tunable. It is only supposed to work as a starting point and to show that following the trajectory based on the velocity profile is possible.
3) Adhere to the common interface, where input is the robot pose and a virtual target frame, and output is a robot velocity.

A control law was designed based on a mixture between the equations from (34) and (35) and the more complex control law in [CAG08].

The input to the controller is the desired pose reference $\xi_{ref}$ given in the same reference frame as the pose of the robot. The reference velocity $\dot{\xi}_{ref}$ is the local velocity of the reference point on the trajectory.

$$\xi_{ref} = \begin{pmatrix} x_{ref} \\ y_{ref} \\ \theta_{ref} \end{pmatrix} \tag{36}$$

$$\dot{\xi}_{ref} = \begin{pmatrix} \dot{x}_{ref} \\ 0 \\ \dot{\theta}_{ref} \end{pmatrix} \tag{37}$$

(38)

The error in the reference is transformed into the frame of the robot using its currently known orientation.

$$e = \mathbf{R}(\theta)(\xi_{ref} - \xi) \tag{39}$$

$$= \begin{pmatrix} e_x \\ e_y \\ e_\theta \end{pmatrix} \tag{40}$$

The controller is designed to handle small errors, that means when the robot is on track on the trajectory. It is not designed to guide the robot back on the trajectory if it has lost track.

The controller equations (41) and (42) consists of four elements.

- The reference velocity $\dot{\xi}_{ref}$ is mapped directly into the robot velocity.
- The correction of the positional $e_x$ error controller by a proportional gain $\kappa_\rho$.
- The correction of the sideways error $e_y$ controlled by the gain $\kappa_\alpha$. Since this error is corrected by turning the robot in the direction towards the trajectory, the correction is also proportional to the expected positional velocity $\dot{x}_{ref}$. This makes the robot turn towards the trajectory even when reversing, and also avoids over correction when the robot is driving slowly.
- The orientation error $e_\theta$ is corrected with a proportional gain $\kappa_\beta$.

$$\dot{x}_{control} = \dot{x}_{ref} + \kappa_\rho \, e_x \tag{41}$$

$$\dot{\theta}_{control} = \dot{\theta}_{ref} + \kappa_\alpha \, e_y \, \dot{x}_{ref} + \kappa_\beta \, e_\theta \tag{42}$$

In case of cumulative position errors, the robot can appear to be on track based on the current pose estimation but in reality be far from the trajectory. When the localizer corrects the pose, the error will become large, resulting in a desperate corrective velocity from the robot. To have more control of the motion that guides the robot back on track, the trajectory is modified when large localizer corrections occur with the current robot pose as the starting point. This is the same method used in [LSB09].

## VII. EXPERIMENTS

An experiment was performed to verify that it is possible to follow the continuous curvature trajectory with the analyzed velocity profile. The robot used for the experiment is a small differential drive robot called SMR (Small Mobile Robot) designed and used locally at the Department of Electrical Engineering. The robot has two powered rear wheels (diam $= 65mm$), and two free caster wheels in front. The dimensions of the robot is $28cm \times 32cm$ (w $\times$ l). The robot is seen in figure 12.

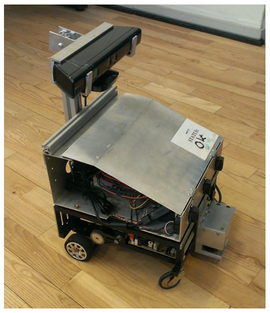

Fig. 12. The DTU SMR Robot used for the experiments

The proposed controller was used to make the robot follow the example 1 trajectory. The control loop was run at 100Hz on a local computer connected to the robot through a LAN cable. The gains used for the controller were $\kappa_\rho = 2.5$, $\kappa_\alpha = 10.0$ and $\kappa_\beta = 4.0$. Figure 13 shows the experiment results. It illustrates both the reference pose and the actual path of the robot.

It is seen from the figure that the robot nicely follows the path, although with slight oscillations. When following the curve, after approximately 15 seconds it is seen that the robot is slightly off track both in position and in orientation. These observations could be a result of the simple controller, non-optimal parameters for the controller, or high delays over the network. Despite these problems, the results show how the robot nicely follows all parts of the trajectory including rotation on the spot, full stops, reversing, and curved motion. The robot slows down before the full stops and it drives slower during the curved motion as the velocity profile dictates.

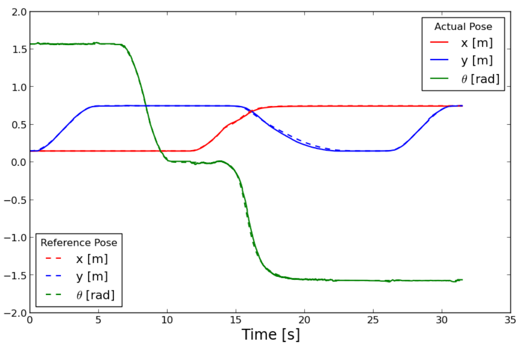

Fig. 13. Experimental results for the SMR following Example Trajectory 1

## VIII. CONCLUSION

In this article the initial work towards a generic navigation solution for wheeled mobile robots was performed.

The method to create a drivable trajectory assumes that a curved trajectory adhering to the geometrical constraints of the robot is already found. A navigation solution should also include the step of finding this trajectory.

The method to make the trajectory curvature-continuous slightly alters the trajectory. Even if the initial trajectory is feasible, the method could therefore potentially generate a trajectory with collision, or increase the curvature beyond the geometrical limits of the robot. One potential solution is to allow the method to also move the knot points slightly, to optimize the path and minimize the curvature in high-curvature areas of the trajectory. When the trajectory is modified, the method should perform collision checking. Moving the knots will also allow for other optimization schemes, such as with respect to driving speed or higher safety.

The controller for trajectory-following is designed as very simple. Further work is intended to be put into analyzing the performance of the controller, or potentially consider a more advanced version. It is also possible that less generic controllers should be designed for the different types of wheel combinations instead of a generic one. Doing so, will make it possible to take the respective robot model further into account, and potentially implement observers for detecting errors in the configured robot parameters.

## References

[CAG08] Ricardo Carona, A. Pedro Aguiar, and José Gaspar. "Control of Unicycle Type Robots - Tracking, Path Following and Point Stabilization". In: *Proceedings of IV Jornadas de Engenharia Electrotécnica e de Computadores*. 2008.

[Cha82] R. Chatila. "Path planning and environment learning in a mobile robot system". In: *Proceedings of ECAZ, Orsay, France*. 1982.

[FBT97] D. Fox, W. Burgard, and S. Thrun. "The Dynamic Window Approach to Collision Avoidance". In: *IEEERobotics & Automation Magazine* 4.1 (1997).

[GK08] Shilpa Gulati and Benjamin Kuipers. "High Performance Control for Graceful Motion of an Intelligent Wheelchair". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2008.

[Gla90] Andrew S. Glassner. *Graphics Gems*. Orlando, FL, USA: Academic Press, Inc., 1990. ISBN: 0122861655.

[GSB05] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. "Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2005.

[GSB06] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters". In: *IEEE Transactions on Robotics* (2006).

[Hae+03] D. Haehnel et al. "A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements". In: *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*. 2003.

[Lik+05] Maxim Likhachev et al. "Anytime dynamic a*: An anytime, replanning algorithm". In: *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*. 2005.

[L L03] A. Pascoal L. Laperre D.Soetanto. "Non-singular Path-Following Control of a Unicycle in the Precense of Parametric Modeling Uncertainties". In: *International Journal of Robust and Nonlinear Control* (2003).

[LSB09] Boris Lau, Christoph Sprunk, and Wolfram Burgard. "Kinodynamic Motion Planning for Mobile Robots Using Splines". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)*. 2009.

[MF07] Christian M and El Udo Frese. "Comparison of Wheelchair User Interfaces for the Paralysed: Head-Joystick vs. Verbal Path Selection from an offered Route-Set". In: *Proceedings of the 3rd European Conference on Mobile Robots, EMCR 2007*. 2007.

[Mic+93] Alain Micaelli et al. *Trajectory Tracking for Unicycle-Type and Two-Steering-Wheels Mobile Robots*. Tech. rep. 1993.

[Mor80] Hans Moravec. "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover". In: *tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University & doctoral dissertation, Stanford University*. 1980.

[Sah+07] A. Sahraei et al. "Artificial Intelligence and Human-Oriented Computing". In: Springer, 2007. Chap. Real-Time Trajectory Generation for Mobile Robots.

[SNS11] Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza. *Introduction to Autonomous Mobile Robots*. 2nd. The MIT Press, 2011. ISBN: 0262015358, 9780262015356.

[VAP08] Francesco Vanni, A. Pedro Aguiar, and Antonio M. Pascoal. "Cooperative path-following of underactuated autonomous marine vehicles with logic-based communication". In: *2nd IFAC Workshop Navigation, Guidance and Control of Underwater Vehicles 2nd IFAC Workshop Navigation*. 2008.