Motion planning for a reversing general 2-trailer configuration using Closed-Loop RRT

Conference Paper · October 2016

DOI: 10.1109/IROS.2016.7759544

3 authors:

Niclas Evestedt
Linköping University
9 PUBLICATIONS **199** CITATIONS
SEE PROFILE

Oskar Ljungqvist
AB Volvo
33 PUBLICATIONS **287** CITATIONS
SEE PROFILE

Daniel Axehill
Linköping University
90 PUBLICATIONS **843** CITATIONS
SEE PROFILE

Motion planning for a reversing general 2-trailer configuration using Closed-Loop RRT

Niclas Evestedt[1], Oskar Ljungqvist[1], Daniel Axehill[1]

***Abstract*— Reversing with a dolly steered trailer configuration is a hard task for any driver without extensive training. In this work we present a motion planning and control framework that can be used to automatically plan and execute complicated manoeuvres. The unstable dynamics of the reversing general 2-trailer configuration with off-axle hitching is first stabilised by an LQ-controller and then a pure pursuit path tracker is used on a higher level giving a cascaded controller that can track piecewise linear reference paths. This controller together with a kinematic model of the trailer configuration is then used for forward simulations within a Closed-Loop Rapidly Exploring Random Tree framework to generate motion plans that are not only kinematically feasible but also include the limitations of the controller's tracking performance when reversing. The approach is evaluated over a series of Monte Carlo simulations on three different scenarios and impressive success rates are achieved. Finally the approach is successfully tested on a small scale test platform where the motion plan is calculated and then sent to the platform for execution.**

I. INTRODUCTION

During the last decade Advanced Driver Assistance Systems (ADAS) have been introduced broadly on the passenger car market. Historically the focus has been to increase safety with systems like Lane Keep Assist (LKA), Adaptive Cruise Control (ACC) and Automatic Breaking, however in recent years systems that help the driver to perform complex tasks, such as parallel parking and reversing with a trailer, have been introduced [1]. Reversing with a trailer is known to be a task that needs a fair amount of skill and training to perfect and an inexperienced driver will have problems already performing simple tasks such as reversing in a straight line or make a simple turn around an obstacle. To relieve the driver in such situations, trailer assist systems have been developed that stabilize the trailer around a reference that the driver can specify from a control knob. The trailer assist systems have been released to the passenger car market but an even greater challenge arises when reversing a truck with a dolly steered trailer. This introduces another degree of freedom making it virtually impossible for a driver, without extensive training, to control.

In this paper we present a novel motion planning and control scheme for a reversing general 2-trailer system that can be used to plan complex manoeuvres in parking scenarios or challenging obstacle avoidance situations. A Rapidly Exploring Random Tree (RRT) is used for the motion planning and a closed loop stabilised model of the general 2-trailer configuration is used for tree extensions. A cascaded control scheme using a Linear Quadratic (LQ) controller, based on the work in [2], is used to stabilize the vehicle configuration around an equilibrium point and then a pure-pursuit path tracking controller is used to make the vehicle configuration follow a piecewise linear reference path as in [3].

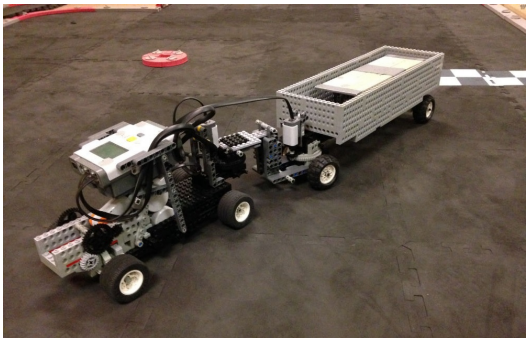

Fig. 1: Truck and trailer system used as a test platform for evaluation experiments. The truck has been built with LEGO NXT and fitted with angle sensors for dolly and hitch angles.

To the best knowledge of the authors this work presents the first motion planning framework for a reversing general 2-trailer configuration where the off-axle hitch connection is considered. The main contributions are the use of CL-RRT using a stabilised system model of the general 2-trailer to get the ability to account for the off-axle hitch and also to include the controller limitations explicitly in the motion plan. The properties of the algorithm are also tested in simulations and the generated paths are validated through real world experiments on a small scale test platform.

A. Related work

The nonlinear dynamics of a standard trailer configuration with the hitch connection in the center of the rear axle are well understood and the derivation of the equations for an n-trailer configuration can be found in [4]. For the 1-trailer configuration an exact local motion planner where path segments consisting of rotational, translational and transition segments with constant steering angle and speed are connected to produce a path from an initial configuration to a goal configuration was presented in [5]. In [6] it is shown that η^4-splines can be used to model the kinematics for the 1-trailer case and these splines are used for a local planner. Trajectory generation for the n-trailer is studied in [7] where the kinematic equations are first transformed to chained form and steered from an initial configuration

*The research leading to these results has been carried out within the iQMatic project funded by FFI/VINNOVA.

[1]Division of Automatic Control, Linköping University, Sweden, (e-mail: {niclas.evestedt, oskar.ljungqvist, daniel.axehill}@liu.se)

to a goal configuration before a transformation back to the original coordinates are performed. One and two trailer planning simulations are presented but obstacles are omitted. Impressive results on global planning with obstacles for a 2-trailer is presented in [8]. However, the approaches presented above consider the case with on-axle hitching despite that most practical applications have off-axle hitching making the kinematics more challenging and the system equations more complicated. Kinematically feasible paths are found by these methods but even though they are kinematically feasible there are no guarantees that there exists a controller that can track the path and in reverse the system is even unstable making the tracking more problematic.

Stabilisation of the reversing trailer system has received a lot of attention. In [9] and [10] the flatness property of the on-axle hitched system is used and controllers using feedback linearization are designed, the feasibility of the controllers are also demonstrated using some 1-trailer lab experiments. However, the assumption that the hitch connection is longitudinally centered at the rear axle center does not hold for passenger cars nor for the truck that will be used in this work. The nonlinear dynamics of the general n-trailer system, where no assumptions are made on the position of the hitching point are derived in [11]. Input-output linearization is used in [1] to derive a controller for the 1-trailer system with off-axle hitching that stabilizes the trailer's driving curvature and a path tracking controller is also considered and tested with good results on a real car test platform. Although these results are very encouraging it is shown in [12] and [13] that trailer configurations with more than one trailer are not input-output linearizable. An LQ based approach for reversing the general 2-trailer system with off-axle hitching is presented in [2]. The system is linearized around an equilibrium point and the linearized system is used to design an LQ control scheme. The work focuses on the stabilization of the internal angles but does not look into path tracking.

To include the complicated dynamics of the general 2-trailer model and the stabilising controller in a motion planning framework this work has focused on sampling based motion planners which offers a simple integration as long as the models can be forward simulated. The use of lattice planners would quickly become intractable due to the huge state lattice needed to include both controller and model dynamics. An increase in the interest for sampling based motion planning algorithms, particularly the ones based on Rapidly Exploring Random Trees (RRT), has been seen during the last decades. LaValle introduced the RRT as a new tool for motion planning in [14]. Later it was used in [15] for kinodynamic planning where fixed time step simulations with random inputs are applied to the model and used for tree construction. A big advantage of this approach is the easy implementation and that it is generally applicable since the only requirement on the model is the possibility of forward simulation. The original RRT algorithm is proved to be probabilistically complete, meaning it will converge to the sampling distribution in the limit, but it has no guarantees on finding the optimal solution and it is even very unlikely that it will. To overcome this Karaman et al. [16] introduced RRT* which is not only probabilistically complete but also probabilistically optimal. However, when used with non-holonomic systems a two point boundary value problem needs to be solved in the re-wire step which is a hard problem by itself and can limit the practical applicability of the algorithm for many systems.

This work is based on results from [17] where inspiration is drawn from the work with Closed-Loop RRTs (CL-RRT) in [18]. Using the results from [3] where an LQ-controller is used to stabilize the internal angles of the system and a pure pursuit path tracking controller is used for path tracking the input to the pure-pursuit controller is sampled with an CL-RRT algorithm to produce feasible paths that include both the kinematics of the model and the dynamics of the controller. However, with this approach optimality and completeness guarantees cannot be formally guaranteed [19] and instead convincing practical results from simulations and some experiments on a small scale platform are provided which demonstrate the applicability of the algorithm even without these formal guarantees.

The outline of the remainder of the paper is as follows: In Section II the RRT framework is presented before the system model and stabilising controllers are presented in Section III and IV, respectively. Section V describes the integration of the system model into the RRT framework before the small scale test platform is described in Section VI. Finally, Section VII and Section VIII present the results and conclusions, respectively.

II. RRT-FRAMEWORK

Applying the original RRT framework, where the sampling takes place in the input space of the model directly, to unstable system models, such as the reversing general two-trailer model, is usually not meaningful since the trajectories quickly diverge from useful ones. To overcome this problem Kuwata et al. [18] introduced Closed-Loop RRT (CL-RRT) where the system model is first stabilised by an inner control loop and then the sampling is done in the reference space of the controller enabling for longer and smoother simulation paths as the system model is stabilised by the controller and not the sampling. A system model $\mathbf{f}(\mathbf{x},\mathbf{u})$, a stabilizing controller $\mathbf{g}(\mathbf{x})$, a representation of the free space $\mathscr{X}_{free}$ and a collision check function, $c(x,\mathscr{X}_{free})$, that tests a state x against $\mathscr{X}_{free}$ and returns true if $x \in \mathscr{X}_{free}$ and false if $x \notin \mathscr{X}_{free}$, needs to be developed as part of the CL-RRT framework. The system model, $\mathbf{f}(\mathbf{x},\mathbf{u})$ and the stabilizing controller $\mathbf{g}(\mathbf{x})$ used for this work is further presented in Sections III and IV.

A. Tree expansion

The CL-RRT algorithm incrementally builds a tree, T, of feasible paths originating from the current root state while attempting to reach a goal state specified by a user or some higher level mission planner. In every iteration of the algorithm a sample, $\mathbf{s} = [s_x, s_y, s_\theta]^T$, is drawn in the controller

input space and a connection attempt is done from the best node, q_{best}, in T. To determine q_{best} the tree nodes are sorted according to a connection heuristic function, $h(\mathbf{s},q)$, to create a sorted list, Q, and the lowest cost node is selected as q_{best}. Instead of creating incremental ε-steps of motions we employ a greedy extend function motivated in [20]. A reference is formed from q_{best} to the sample $\mathbf{s}$ and the system is simulated along the reference either until the neighborhood of $\mathbf{s}$ is reached or an obstacle is hit. If the extension is successful it is added to the tree and the procedure is repeated with a new sample, if the extension fails an attempt to connect to the second best node in Q is performed until the maximum number of connection attempts, n_{max}, is reached or a feasible connection is found. After a successful connection, a greedy goal connection attempt is performed to see if the goal is directly reachable from the newly added node. This greatly lowers the time for a first solution in open environments or when the tree search has entered the surroundings of the goal area. Since an exact steering method is not available we will not reach the goal exactly and the resulting simulation state is instead tested against the goal state and if it is within a specified goal interval it is stored as a solution. As more iterations are performed more and more solution paths are found. The available solution paths are then scored according to a cost function and the minimum cost solution is returned. As shown in [19] probabilistic completeness is not guaranteed if best input selection is used instead of random input. This is also true for our case and probabilistic completeness is here abandoned in favour of the faster exploration achieved by not using random input selection. The expansion algorithm is summarized in Algorithm 1 and further implementation details including the general 2-trailer model can be found in Section V.

Algorithm 1 Tree expansion

```
1:  function EXPANDTREE
2:      N ← ∅
3:      s ← drawSample()
4:      Q ← sortNodes(T, h(s,q))
5:      for each node q in Q do
6:          r ← formReference(s,q)
7:          x(t), q_new ← expand(f, g, q, r)
8:          if x(t) ∈ X_free(t) then
9:              q_new.cost = calculateCost(q, x(t))
10:             N.push(q_new)
11:             addToTree(x(t), q_new)
12:             break;
13:     s_goal ← drawGoalSample()
14:     for each node q in N do
15:         r ← formReference(s_goal, q)
16:         x(t), q_goal ← expand(f, g, q, r)
17:         if x(t) ∈ X_free(t) then
18:             q_goal.cost = calculateCost(q, x(t))
19:             addToTree(x(t), q_goal)
20:             break;
        return
```

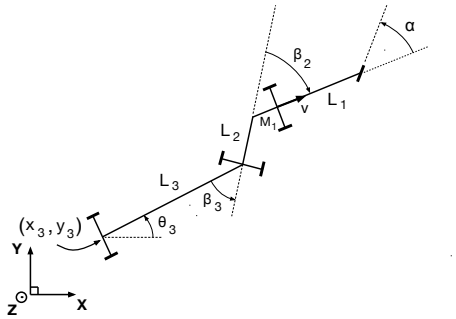

Fig. 2: Schematic view of the configuration used to model the general two-trailer system.

III. System dynamics

In this section we present the model used for forward simulations and controller design for the general 2-trailer system with an off-axle hitching on the pulling vehicle. A schematic overview of the configuration is shown in Fig 2. The generalized coordinates used to model the system are, $\mathbf{p} = [x_3, y_3, \theta_3, \beta_3, \beta_2]^T$ where x_3, y_3 are the position of the rear axle center of the trailer, θ_3 is the heading of the trailer, β_3 is the relative angle between the trailer and the dolly and β_2 is the relative angle between the dolly and the truck. The parameters L_3, L_2, L_1 are the distances between the axle center of the trailer to the axle center of the dolly, the axle center of the dolly to the off-axle hitch connection of the truck and the distance between the axle centers of the truck, respectively. M_1 is the off-axle hitch length for the truck and α is the steering angle. The dynamic model for a general n-trailer system was derived in [11] and the following equations for the special 2-trailer case was presented in [2]:

$$\dot{x}_3 = v\cos\beta_3\cos\beta_2\left(1+\frac{M_1}{L_1}\tan\beta_2\tan\alpha\right)\cos\theta_3 \quad (1)$$

$$\dot{y}_3 = v\cos\beta_3\cos\beta_2\left(1+\frac{M_1}{L_1}\tan\beta_2\tan\alpha\right)\sin\theta_3 \quad (2)$$

$$\dot{\theta}_3 = v\frac{\sin\beta_3\cos\beta_2}{L_3}\left(1+\frac{M_1}{L_1}\tan\beta_2\tan\alpha\right) \quad (3)$$

$$\dot{\beta}_3 = v\cos\beta_2\left(\frac{1}{L_2}\left(\tan\beta_2-\frac{M_1}{L_1}\tan\alpha\right)- \frac{\sin\beta_3}{L_3}\left(1+\frac{M_1}{L_1}\tan\beta_2\tan\alpha\right)\right) \quad (4)$$

$$\dot{\beta}_2 = v\left(\frac{\tan\alpha}{L_1}-\frac{\sin\beta_2}{L_2}+\frac{M_1}{L_1L_2}\cos\beta_2\tan\alpha\right) \quad (5)$$

where v is the longitudinal velocity at the rear axle of the truck. The model is valid under the no slip condition and since our application only concerns manoeuvres at lower speeds this is a feasible assumption. To account for the dynamics of the steering mechanism during forward simulation the steering angle rate is limited by $\dot{\alpha}_{max}$ and a first order system with time constant T_d is used to model the time delay.

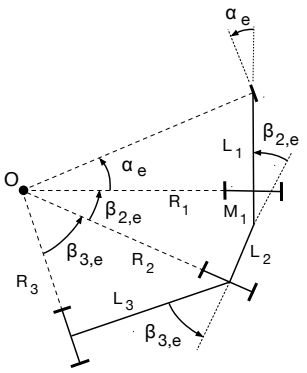

Fig. 3: Stationary equilibrium point for steering angle α_e. The system will travel in a circular path with a radius determined by the geometry and α_e.

IV. Stabilization and Path Tracking

When driving forward, the system is stable but in reverse $(v<0)$ the system becomes unstable. To be able to simulate the system without entering a jack-knife state when reversing the cascaded control approach presented in [3] is used. For the low level stabilization of β_2 and β_3 an LQ-controller with gain scheduling is used and as a higher level path tracker a pure pursuit control law is employed.

A. LQ-controller

The full details of the linearization steps and control design is found in [3] but for clarity the main steps are presented here. By considering the subsystem (4)-(5) and defining $\boldsymbol{\beta} = [\beta_3, \beta_2]^T$, this system can be linearized around circular equilibrium points giving the linearised model

$$\dot{\boldsymbol{\beta}} = \bar{A}(\boldsymbol{\beta} - \boldsymbol{\beta_e}) + \bar{B}(\alpha - \alpha_e) \tag{6}$$

where

$$\bar{A} = \left.\frac{\partial f}{\partial \boldsymbol{\beta}}\right|_{\boldsymbol{\beta_e},\alpha_e}, \quad \bar{B} = \left.\frac{\partial f}{\partial \alpha}\right|_{\boldsymbol{\beta_e},\alpha_e} \tag{7}$$

and $\boldsymbol{\beta_e}, \alpha_e$ are the equilibrium points for a stationary circular movement given a constant steering angle α_e as seen in Fig. 3. Given the linearised model (6) standard LQ design techniques can be used for controller design. Using (6) for designing the controller around the equilibrium configuration the following structure is obtained

$$\alpha = \alpha_e - L(\alpha_e)\left(\boldsymbol{\beta} - \boldsymbol{\beta}_e(\alpha_e)\right) \tag{8}$$

where α_e is the desired linearization point, $L(\alpha_e)$ is the controller gain at this point, $\boldsymbol{\beta}$ the current measured angles and $\boldsymbol{\beta}_e$ is the steady state values for β_2 and β_3 at the equilibrium point for a given α_e. Given a linearization point α_e and the linearized model in (6), the LQ design method finds the optimal gain, $L(\alpha_e)$ minimizing the cost function

$$\mathscr{J} = \int_0^\infty \left(\bar{\boldsymbol{\beta}}^T Q \bar{\boldsymbol{\beta}} + \bar{\alpha}^2\right) dt \tag{9}$$

where $\bar{\boldsymbol{\beta}} = \boldsymbol{\beta} - \boldsymbol{\beta}_e$, $\bar{\alpha} = \alpha - \alpha_e$ and Q is a design parameter. To account for model variations due to the choice of equilibrium point, α_e, a gain scheduled controller gain, $L(\alpha_e)$, is obtained by solving the problem for different linearization points. For the high level path follower, that will be further explained in the next section, we want to be able to control β_3 instead of α. To achieve this β_{3e} is introduced as the reference to the controller by deriving a pre-compensation link from β_{3e} to α_e giving

$$\alpha_e = \arctan\left(\frac{L_1 \operatorname{sign}(\beta_{3e})}{\sqrt{L_3^2\left(1+\frac{1}{\tan^2\beta_{3e}}\right)+L_2^2-M_1^2}}\right) \tag{10}$$

B. Path tracking

When the internal angles β_2 and β_3 are stabilized by the LQ-controller a pure pursuit path follower is used as a high level controller to stabilize the system around a reference path. In forward motion the controller has direct control of the steering angle α and the anchor point of the look-ahead circle is set at the rear axle center of the pulling vehicle. In backwards motion however the look-ahead circle anchor point, P^*, is set in the center of the rear axle of the last trailer and the pure pursuit controller gives the reference β_{3d} to the low level stabilizing controller as depicted in Fig. 4. A piecewise linear reference path is given as input and by calculating the intersection of the look-ahead circle with radius, L_r, and the line segment between two points on the reference path, the error heading, θ_e, can be calculated. The control law for β_{3d} that will drive the vehicle along a circle to the look-ahead point can now be found using basic geometry giving

$$\beta_{3d} = -\arctan\left(\frac{2L_3\sin\theta_e}{L_r}\right) \tag{11}$$

The only tunable parameter is the look-ahead distance, L_r, where a shorter look-ahead gives a more aggressive tracking but can cause instability while a longer look-ahead gives a smoother tracking but causes bigger tracking offset. This path tracker can now be used in the RRT framework by sampling a piecewise linear reference path for the input to the pure

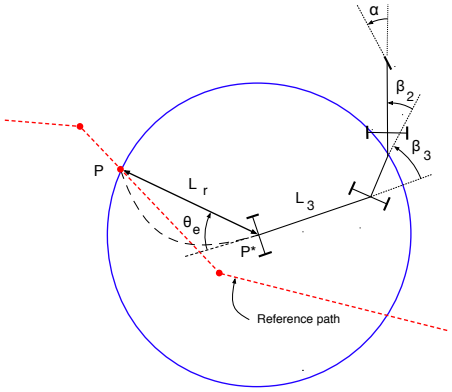

Fig. 4: Geometry of the pure pursuit control law: L_r look-ahead distance, θ_e angle to look-ahead point and α steer angle.

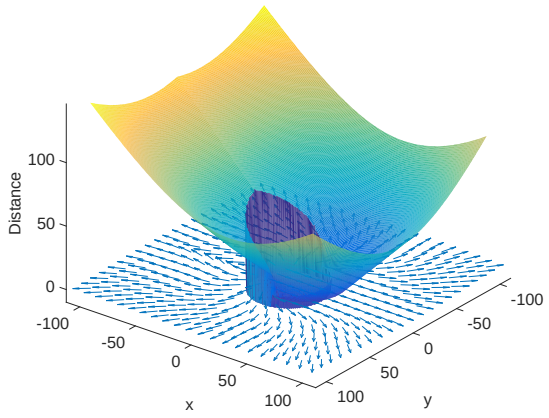

Fig. 5: Pre-calculated lookup table for fast cost estimation. The arrows represent the resulting heading when simulated from the origin to every grid cell and the z-axis represents the distance covered to reach this grid cell.

pursuit controller and then perform a forward simulation of the system along the reference.

V. RRT-INTEGRATION

With the model and controllers presented in the previous section a further presentation of the incorporation of the 2-trailer system in the RRT framework can now be given.

A. Node connection heuristic

The heuristic function, $h(\mathbf{s},q)$, is used to evaluate the cost of connecting the sample $\mathbf{s}$ to a node q in the tree. A natural choice for holonomic systems is the euclidean distance as it is possible to move in all directions. For non-holonomic systems there are other natural choices, e.g. in [18] the Dubins path length is used for a robotic car. To do fast connection cost estimations for the stabilised 2-trailer model, system simulations are performed off-line to create a lookup table that can be quickly accessed at runtime. A 200 m by 200 m grid with a discretization of 0.5 m is created and a reference from the origin to every grid cell is calculated and then simulated. The distance travelled and the resulting heading is stored in the lookup table, $M(\mathbf{s_q})$, as seen in Fig. 5. During runtime the sample $\mathbf{s}$ is transformed to the local frame at q, giving $\mathbf{s}_q$, and the estimated cost can easily be checked in the table by a lookup at the closest grid cell. If the sample would be outside the range of the table the euclidean distance from the table edge to the sample is added to the cost. The resulting heuristic cost is then calculated as

$$h(\mathbf{s},q) = M_d(\mathbf{s}_q) + K_\theta |s_\theta - M_\theta(\mathbf{s}_q)| \tag{12}$$

where $M_d(\mathbf{s}_q)$ is the distance value in the table, $M_\theta(\mathbf{s}_q)$ is the resulting heading in the table, and K_θ is a weighting parameter between travelled distance and heading error.

B. Goal evaluation

As stated earlier the goal will never be reached exactly because an exact steering method is not available and instead the resulting end state of every simulation will have to be checked to see if it is within the specified goal region. In this application we check that the following goal requirements are fulfilled

$$|d_e| < \varepsilon_d \tag{13}$$
$$|\theta_e| < \varepsilon_\theta \tag{14}$$
$$|\beta_{2e}| < \varepsilon_{\beta_2} \tag{15}$$
$$|\beta_{3e}| < \varepsilon_{\beta_3} \tag{16}$$

where d_e, θ_e, β_{2e} and β_{3e} are the position, heading, β_2 and β_3 errors respectively and the ε specifies a error tolerance for every requirement. The values used for our experiments were ε_d = 2.0 m and $\varepsilon_\theta = \varepsilon_{\beta_2} = \varepsilon_{\beta_3}$ = 5.0 degrees.

C. Cost function

When multiple feasible solution paths are found a cost function is used to determine which path that is the best one. The cost for this application is based on the path length and a goal cost that evaluate how close to the final goal the solution has come. The cost of reversing is chosen to be twice the cost of travelling forward to prefer paths with forward motion rather than reversing motion. The cost, c_q, for a node is calculated as

$$c_q = d_{q,root} + K_{goal} c_{goal} \tag{17}$$

where $d_{q,root}$ is the path length from the root to the node q, K_{goal} is a weighting parameter between the path length and the goal deviation and c_{goal} is the goal deviation specified as

$$c_{goal} = d_e^2 + K_\theta \theta_e^2 \tag{18}$$

where K_θ is a weighting parameter. For our experiments we have used K_{goal} = 25 and K_θ = 10 and the values have been found through experiments.

VI. EXPERIMENTAL PLATFORM

Algorithmic properties will be evaluated in simulation but some real world experiments are included to demonstrate the feasibility of the complete approach. The experiments have been carried out on the test platform shown in Fig. 1. The platform consists of a small scale truck with Ackerman steering and an off-axle hitching which is connected to a trailer with a turntable dolly. A LEGO NXT control brick is used as the on board computer and the angles β_2 and β_3 are measured using two HiTechnic angle sensors. The LQ-controller and the pure pursuit controller is running onboard the NXT control brick with an update frequency of 100 Hz and 10 Hz respectively. A high accuracy Qualisys Oqus motion capture system is used for positioning and the information is relayed to the NXT via a Bluetooth connection. Overall control of the system is done through a laptop computer running the RRT framework where reference paths can be created and then sent via Bluetooth to the vehicle.

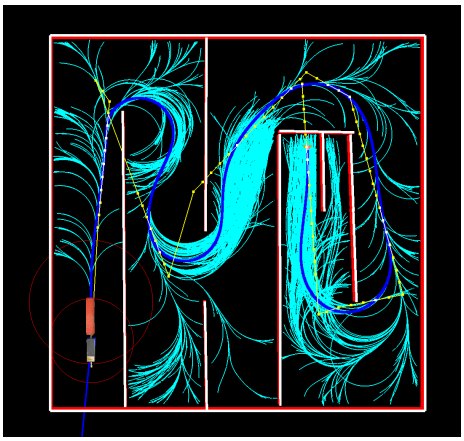

Fig. 6: Complex environment together with the search tree shown in turquoise. The best solution in the tree is shown in dark blue.

A. Parameters

By measuring the distances between the wheel axles the following parameters can be found for our vehicle, $L_1 = 19.0$ cm, $L_2 = 14.0$ cm, $L_3 = 34.5$ cm, $M_1 = 3.6$ cm and the steering angle is limited to ± 44 degrees.

VII. RESULTS

The performance of the algorithm is evaluated through a series of experiments run on a standard laptop computer with an Intel Core i7-4810MQ@2.8GHz CPU. Since the algorithm has a probabilistic nature different performance is expected even when presented with the same problem so in the first two experiments two different scenarios are presented to the algorithm and multiple solution attempts with different random seeds are performed and solution performance data is stored after each attempt. The algorithm is allowed to run for a maximum of T_{max} seconds and when the first solution is found the run time, solution cost and number of expanded nodes are stored. When the maximum planning time is reached solution cost and expanded nodes are stored again to see if the solution has been improved by the extra calculation time. A third scenario taken from the reversing manoeuvre test that has to be performed as part of the Swedish driving test to get a licence for heavy trucks with dolly steered trailer is also included. Finally the concept is tested on a small scale test platform.

A. Maze

As a first scenario a complicated maze like environment is constructed and to get a good statistical basis the scenario is run 10000 times with a maximum planning time of 60 s. The scenario is quite unrealistic and most drivers will never be faced with such complex environments when reversing but it is included to demonstrate the capabilities of the planning algorithm. The scenario is shown in Fig. 6 together with the search tree for one of the solutions. The results from the 10000 runs are presented in the histograms in Fig. 7. The sampling is done uniformly over the environment with 80% of the samples as reverse motions and 20% forward motions. Even though the algorithm has no formal completeness guarantees only nine of the 10000 planning attempts failed to find a solution within the 60 s planning time, giving a success rate of 99.91%. As illustrated in Fig. 7 the majority of the solutions are found within ten seconds and the peak of the histogram is centred around 2.77 s. The number of nodes expanded when the first solution is found is also shown in Fig. 7 and as seen most solutions expand less than 10000 nodes while an average of 23187 nodes are expanded when the maximum planning time has elapsed. The total solution cost according to the cost function presented in Section V is shown in the lower histogram of Fig. 7. It is seen that given the extra computation time from first solution until the maximum planning time, the cost of the solution is reduced and an improved solution can be found. However the improvement is not huge and most of the improvement come from a reduced goal error while the main part of the solution is unchanged.

B. Three point turn

A more realistic scenario, that a driver might encounter, and that will be presented to the algorithm is to perform a three point turn in order to turn the vehicle around when the road in front is too narrow or have become obstructed. The scenario setup is shown in Fig. 8 where the vehicle must reverse into the gap on the side to be able to drive forward and turn the vehicle around. A uniform sampling over the environment with 80% of the samples as reverse motions and 20% forward motions is used and the scenario is solved 2000 times with the maximum planning time, T_{max}, set to 30

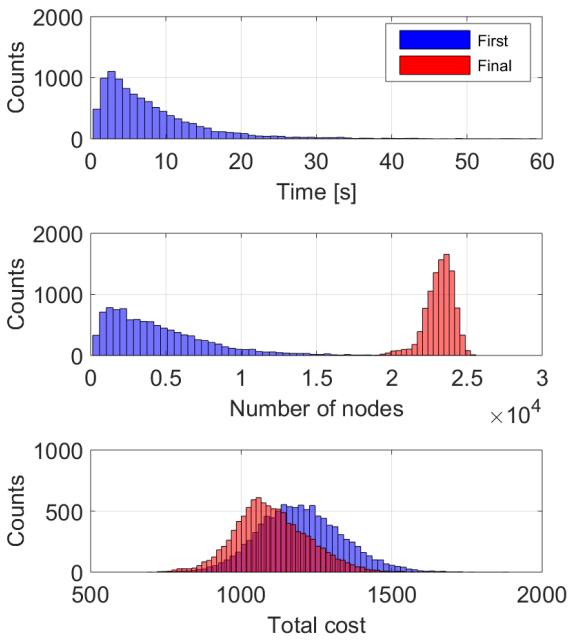

Fig. 7: Histogram over the solutions for the complex environment. Time, number of nodes and total cost of first solution and final solution are shown in the top, middle and lower histogram, respectively.

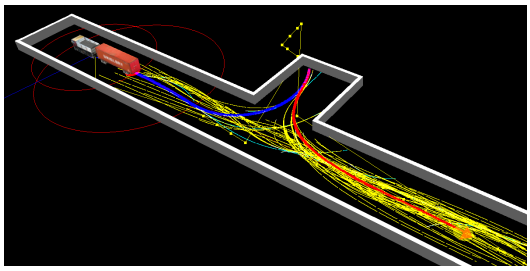

Fig. 8: Three point turn scenario where the configuration has to be turned around by reversing into the gap before a forward motion can take the vehicle to the goal. Turquoise paths represent backward motions stored in the tree and yellow forward motions. Dark blue and red paths are the reverse and forward motion representing the best solution in the tree.

s. In this scenario a success rate of 99.30% is achieved where 14 of the 2000 attempts did not return a solution within the maximum planning time, however most of the solutions that were found were found within 5 s as seen in Fig. 9. Most of the first solutions expanded less than 2000 nodes while an average of 13382 nodes were expanded when the maximum planning time had elapsed. As seen in the lower histogram in Fig. 9 the cost of the solution is improved when given extra computation time and again most of the improvement comes from a reduced goal error however a quite large tail is seen in the histogram giving a broad range of costs. This tail can be explained because some solutions will not find a straight reverse into the gap and then a forward motion to the goal but instead it will add extra forward backward manoeuvres before it can turn the vehicle around. As it is known that this problem can be solved by only one backward and one forward motion this can cause undesired unintuitive solutions that extra computation time can not improve in many cases since no optimality guarantees exist.

C. Driver test

The last scenario is the manoeuvre test that has to be completed as part of the Swedish driving test for heavy trucks with dolly steered trailers. The test simulates the scenario when a driver has to park the vehicle configuration in a narrow parking space as depicted in Fig. 10. The starting position has to be at least two vehicle configuration lengths away from the entrance to the lot and the lot should have an angle between ± 45 degrees relative to the heading of the vehicle. The driver is then given a maximum of 10 minutes to manoeuvre the configuration into the parking area without hitting the cones that specifies the parking area. This scenario was given to the algorithm and run 2000 times with a maximum planning time of 30 s. The initial position of the vehicle was randomized in every iteration according to the scenario specifications except that the relative angle was increased to ± 90 degrees to give an even greater challenge. Uniform sampling was used but only reversing motions were sampled for this scenario giving a success rate of 97.0% with 60 failed attempts within the 30 s given. The results of the runs are summarized in Fig. 11 and as seen when a

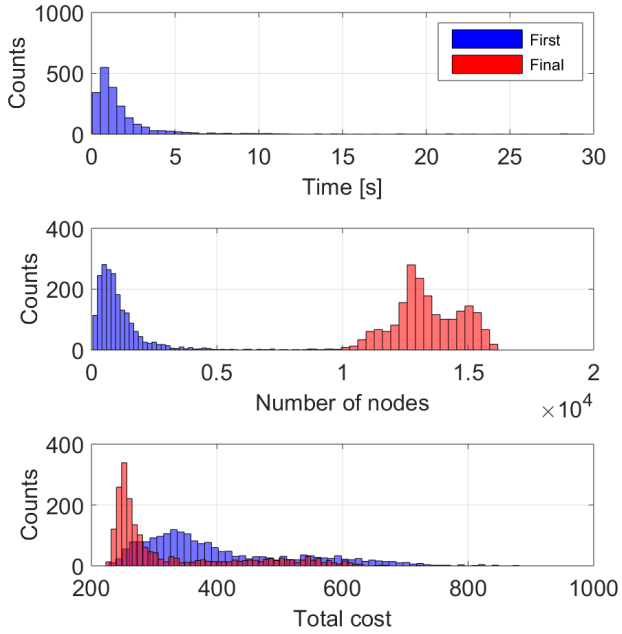

Fig. 9: Histogram over the solutions for the three point turn scenario. Time, number of nodes and total cost of first solution and final solution are shown in the top, middle and lower histogram, respectively.

solution is found it is often found within the first second and only expanding a few hundred nodes. Given more time the solution is improved and again most of the improvement comes from a better end pose that decreases the goal error. Even though this scenario seems much less complex with only a reverse into the parking lot the scenario has the lowest success rate of the three scenarios tested. This shows one of the problems with the simulation based expansion that is performed by the forward simulations, even though the topology of the scenario is very simple the precision required to enter the narrow parking area is hard to achieve with only random sampling and forward simulations and is similar to the well known bug trap problem that RRTs are facing.

D. Real world

For a thorough analysis of the tracking performance on the small scale test platform the reader is referred to [3] and

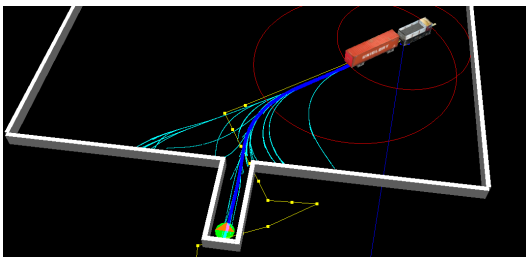

Fig. 10: One instance of the driver test scenario. The vehicle is started with an angle to the parking spot and has to be reversed into the spot. Turquoise lines shows the search tree and the dark blue path represent the best solution in the tree.

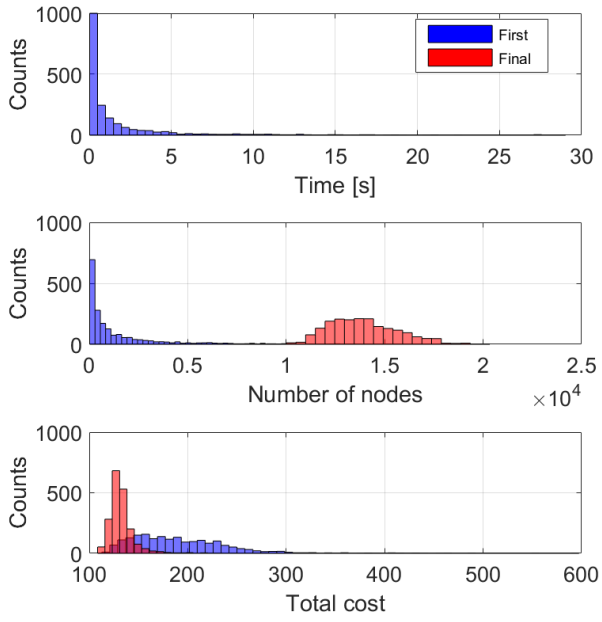

Fig. 11: Histogram over the solutions for the driver test scenario. Time, number of nodes and total cost of first solution and final solution are shown in the top, middle and lower histogram, respectively.

a further analysis will not be given here. A mean tracking error of a few centimetres and slightly larger maximum errors have been achieved in the lab. Further experiments with the planning framework and the small scale test platform can be seen at `https://youtu.be/qVeYNmG157k`.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper a motion planning and control framework for the general 2-trailer vehicle configuration is presented. A cascaded control structure, with an LQ-controller that stabilises the internal angles and a pure pursuit path tracking algorithm for reference following, is used for forward simulations within a motion planning framework based on Closed-loop Rapidly Exploring Random Trees. The algorithm has no formal completeness guarantees but an average success rate of 98.7367 % was achieved in Monte Carlo simulations for three test scenarios. Since the approach is firstly meant to be used as assistance for a driver the failed cases poses no danger as the driver can not start the execution if no path exists and given the probabilistic nature of the algorithm there is a good chance a path will be found if the solver is restarted. Two big drawbacks with the method is the extensive sampling needed to achieve an acceptable goal pose and the possibility for undesired unintuitive solutions. Therefore current and future work is focused on finding a steering method that can connect two poses exactly but so far this has proved to be too time consuming to be feasible within the RRT framework. If an efficient steering method can be found it could be used within an RRT* algorithm and probabilistic completeness and optimality can be guaranteed.

The approach was also tested successfully on a small scale test platform in a lab environment while tests on a full scale truck and trailer configuration is on going work and at the moment a reliable method for estimation of β_3 and β_2 is under development.

ACKNOWLEDGMENT

We gratefully acknowledge the Royal Institute of Technology for providing us with the opportunity to perform experiments at their facilities at the Smart Mobility Lab.

REFERENCES

[1] M. Werling, P. Reinisch, M. Heidingsfeld, and K. Gresser, "Reversing the general one-trailer system: Asymptotic curvature stabilization and path tracking," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 15, no. 2, pp. 627–636, 2014.

[2] C. Altafini, A. Speranzon, and K. H. Johansson, "Hybrid control of a truck and trailer vehicle," in *Hybrid Systems: Computation and Control*. Springer, 2002, pp. 21–34.

[3] N. Evestedt, O. Ljungqvist, and D. Axehill, "Path tracking and stabilization for a reversing general 2-trailer configuration using a cascaded control approach." arXiv:1602.06675, 2016.

[4] O. J. Sørdalen, "Conversion of the kinematics of a car with n trailers into a chained form," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*. IEEE, 1993, pp. 382–387.

[5] P. Švestka and J. Vleugels, "Exact motion planning for tractor-trailer robots," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 3. IEEE, 1995, pp. 2445–2450.

[6] F. Ghilardelli, G. Lini, and A. Piazzi, "Path generation using-splines for a truck and trailer vehicle," *Automation Science and Engineering, IEEE Transactions on*, vol. 11, no. 1, pp. 187–203, 2014.

[7] D. Tilbury, R. M. Murray, and S. Shankar Sastry, "Trajectory generation for the n-trailer problem using goursat normal form," *Automatic Control, IEEE Transactions on*, vol. 40, no. 5, pp. 802–819, 1995.

[8] S. Sekhavat, J.-P. Laumond, and M. Overmars, "Multi-level path planning for nonholonomic robots using semi-holonomic subsystems."

[9] D.-H. Kim and J.-H. Oh, "Experiments of backward tracking control for trailer system," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 1. IEEE, 1999, pp. 19–22.

[10] M. Sampei, T. Tamura, T. Kobayashi, and N. Shibui, "Arbitrary path tracking control of articulated vehicles using nonlinear control theory," *Control Systems Technology, IEEE Transactions on*, vol. 3, no. 1, pp. 125–131, 1995.

[11] C. Altafini, "The general n-trailer problem: conversion into chained form," in *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, vol. 3. IEEE, 1998, pp. 3129–3130.

[12] R. M. Murray, "Nilpotent bases for a class of nonintegrable distributions with applications to trajectory generation for nonholonomic systems," *Mathematics of Control, Signals and Systems*, vol. 7, no. 1, pp. 58–75, 1994.

[13] P. Rouchon, M. Fliess, J. Levine, and P. Martin, "Flatness and motion planning: the car with n trailers," in *Proc. ECC93, Groningen*, 1993, pp. 1518–1522.

[14] S. M. LaValle, "Rapidly-exploring random trees a new tool for path planning," 1998.

[15] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[16] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[17] O. Ljungqvist, "Motion planning and stabilization for a reversing truck and trailer system," Master's thesis, 2016.

[18] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *Control Systems Technology, IEEE Transactions on*, vol. 17, no. 5, pp. 1105–1118, 2009.

[19] T. Kunz and M. Stilman, "Kinodynamic RRTs with fixed time step and best-input extension are not probabilistically complete," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 233–244.

[20] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 995–1001.