2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

On the Design of Deformable Input- / State-Lattice Graphs

Martin Rufli Roland Siegwart

Autonomous Systems Lab, Institute for Robotics and Intelligent Systems, ETH Zurich
Tannenstrasse 3, CH-8092 Zurich, Switzerland

martin.rufli@mavt.ethz.ch rsiegwart@ethz.ch

***Abstract*— In this paper we describe a novel and simple to implement yet effective lattice design algorithm, which simultaneously produces input *and* state-space sampled lattice graphs. The presented method is an extension to the ideas suggested by Bicchi *et al.* on input lattices and is applicable to systems which can be brought into (2,n) chained form, such as kinematic models of unicycles, bicycles, differential-drive robots and car-like vehicles (pulling several trailers).**

We further show that a transformation from chained form to path coordinates allows the resulting lattice to be bent along any C^1 continuous path. We exploit this fact by shaping it along the skeleton of arbitrary structured environments, such as the center of road lanes and corridors. In our experiments in both structured (i.e. on-road) and unstructured (i.e. parking lot) scenarios, we successfully demonstrate for the first time the applicability of lattice-based planning approaches to search queries in arbitrary environments.

***Index Terms*— Non-holonomic motion planning, deformable input- / state-lattice**

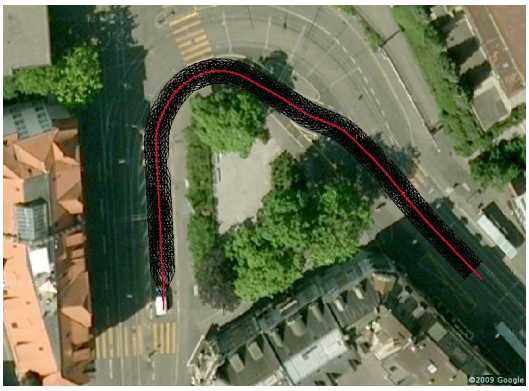

Fig. 1. Google maps view of Haldenegg curve, Zurich. Our novel lattice is bent along the centerline of the right lane. Using this approach, solutions on the lattice are naturally aligned with the road's direction.

I. INTRODUCTION

Global real-time trajectory planning has traditionally been achieved by reducing the high-dimensional state-space to a simpler more approachable one, such as a 2D grid. While moving obstacles and arbitrary robot shape can be treated within the reduced-dimensional planning setup (i.e. via frequent re-planning [1], [2] or motion prediction [3], and conservative obstacle inflation, convolution approaches [1], [4], respectively), the resulting plan's violation of (nonholonomic) vehicle constraints either necessitates the addition of a path smoothing layer, which negates any previously obtained optimality guarantees, or an exceedingly robust and consequently slow controller implementation.

Recently it has thus been realized, that by incorporating vehicle kinematic (and dynamic) constraints into the planning stage, the load on the path following and controller modules may be reduced dramatically, and impassable and / or potentially dangerous (dynamic) regions may be identified and avoided before encountering them directly. Unfortunately, the incorporation of such constraints increases the complexity of the motion planning problem dramatically (for a kinematic car-like vehicle this would amount to the addition of heading, steering angle, velocity and time dimensions in addition to 2D position), resulting both in a substantially larger and more complex search space.

At the same time, applications in dynamic urban environments demand real-time planning and re-planning capability (i.e. at 10Hz). The tradeoff between higher-dimensional (and thus more precise) search queries and low execution times is thus a delicate one, that has led to several environment and application specific approaches. Primarily we distinguish between cases where the structure can be deduced from the environment (such as on-road, and in-door), and cases, where such structure is largely lacking (off-road, parking lots).

In the former case, Dolgov *et al.* employ a low-dimensional search followed by a smoothing step, where the smoothing minimizes the orientation mismatch between the path and the external structure [5]. They argue that the low-dimensional solution typically returns a motion close to the optimal high-dimensional one, so that the smoothing step then finds the local high-dimensional optimum in the vicinity (which is often the global one). While this approach may work well in static cases, it fails in dynamic scenes where the low-dimensional search ignores vital non-convex elements of the high-dimensional cost map. Howard *et al.* solve two-point boundary problems between the current vehicle position and a point some fixed distance ahead along the structured environment (such as a road) [1]. In areas where such structure is lacking, it is difficult to assess whether a potential goal state lies along the globally optimal trajectory, however.

In the latter case, lattice based planner have emerged as the solution of choice, as they may yield (sub-) optimality guarantees on the underlying lattice graph at a low compu-

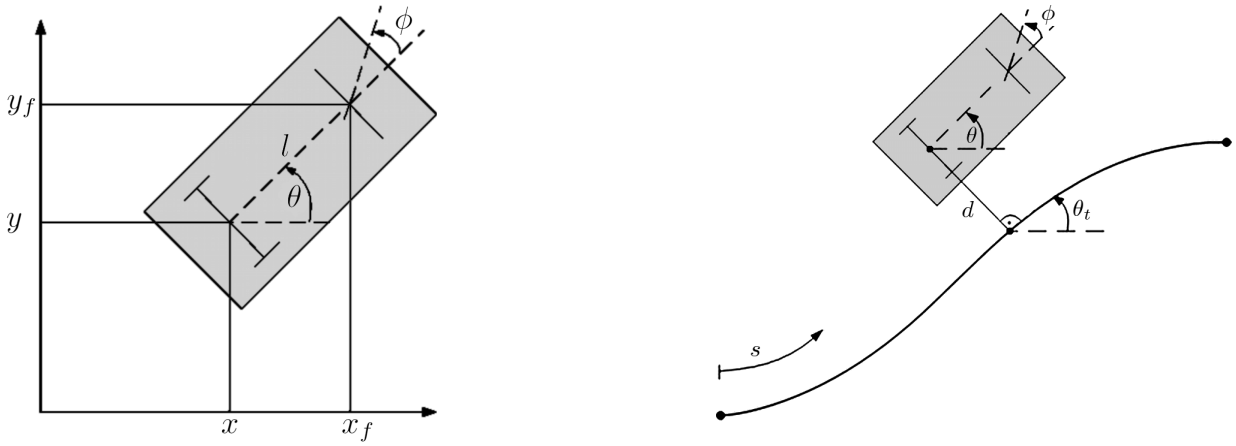

Fig. 2. **Left:** Vehicle kinematics in *physical form*. (x, y) denote the 2D position of the rear- and $(x_\mathrm{f}, y_\mathrm{f})$ of the front-axle center. Heading (θ) is measured counter-clockwise with respect to the x-axis and steering angle (ϕ) with respect to the vehicle orientation. The wheelbase of the vehicle is described with l. **Right:** Vehicle kinematics in *path form*. The reference path s is fully defined trough its curvature $c(s)$. (s, d) denote the distance along, and perpendicular to the path, respectively. $\theta_\mathrm{p} = \theta - \theta_\mathrm{t}$ marks the difference in heading between path and vehicle and ϕ is the steering angle.

tational cost. Bicchi *et al.* suggest an input sampled lattice, which is obtainable for a wide range of vehicle models via a coordinate transformation into chained form coordinates [6]. For certain input sets they prove that the transformed state space spans a lattice. Later, Pivtoraiko *et al.* developed an algorithm to generate lattices in state space directly [7]. Their method is more involved but generates a minimal edge-representation for the chosen state space discretization. Nonetheless, the globally fixed state-space discretization (in heading dimension) has so far prohibited the employment of lattices in structured environments.

This concern is resolved by our main contribution. It has its foundation in a principled yet simple method for the simultaneous generation of an input *and* state lattice that is applicable to vehicle models which can be brought into (2,n) chained form (such as the kinematic model of a car). A coordinate transformation from chained form into so called path coordinates then allows for the bending of this lattice along arbitrary C^1 continuous paths and thus for the first time enables its use in structured environments.

The remainder of this paper is organized as follows: in Sections II & III we review the kinematic model of a car-like vehicle, and then analyze its reachable set. Section IV details our novel lattice generation algorithm. Finally, in Section V we provide a comparison between regular and shaped lattices in hybrid structured-unstructured environments, where previously lattice planners could not be employed.

II. KINEMATIC MODEL OF A CAR-LIKE VEHICLE

In this section we review the kinematics of a slowly-steered car-like vehicle consisting of four states that span the configuration space: 2D position (x, y), heading (θ) and steering angle (ϕ). These kinematics may be described in various equivalent coordinate frames (treated in Sec. II-B & II-C) to facilitate some of the arguments in Sec. III & IV.

When referring to a *system*, throughout this paper we mean it in the context of control theory and modelling [6].

Definition 1. *"A* system *is a quintuple (X, T, U, Ω, A), where X denotes the configuration set, T an ordered time set, U a set of admissible (possibly configuration dependent) input symbols, Ω a set of admissible input words formed by symbols in U and A a state transition map $A : T \times \Omega \times X \to X$" [6].*

A. Physical Form Coordinate Frame

An easily approachable way of deriving the kinematic representation of a (rear-wheel driven) car-like vehicle involves the four states (x, y, θ, ϕ) as introduced in Fig. 2, left. We will henceforth refer to this representation as *physical form*. The vehicle is subject to two non-holonomic constraints, one each for the lumped together front $(x_\mathrm{f}, y_\mathrm{f})$ and rear (x, y) wheels. They specify that the velocity of each wheel must never have any lateral component (roll and no-slip condition).

$$\begin{aligned} 0 &= \dot{x}_\mathrm{f} \sin(\theta + \phi) - \dot{y}_\mathrm{f} \cos(\theta + \phi) \\ 0 &= \dot{x} \sin\theta - \dot{y} \cos\theta \end{aligned} \tag{1}$$

Furthermore, the front axle coordinate and velocity components are related to their rear axle counterparts through the rigid body assumption (with l the vehicle wheelbase)

$$\begin{aligned} x_\mathrm{f} &= x + l\cos\theta \\ y_\mathrm{f} &= y + l\sin\theta \end{aligned} \quad \Longrightarrow \quad \begin{aligned} \dot{x}_\mathrm{f} &= \dot{x} - \dot{\theta}\, l \sin\theta \\ \dot{y}_\mathrm{f} &= \dot{y} + \dot{\theta}\, l \cos\theta \end{aligned} \tag{2}$$

Finally, the means by which we act on the vehicle are specified via forward speed v_car (throttle) and the front wheels' steering velocity $\dot{\phi}$ (through the steering wheel). We refer to these inputs as

$$\begin{aligned} v_1 &= v_\mathrm{car} = \sqrt{\dot{x}^2 + \dot{y}^2} \\ v_2 &= \dot{\phi} \end{aligned} \tag{3}$$

By combining Eqns. 1-3 and solving for the state variables $[x, y, \theta, \phi]^T$, the kinematic state-space representation in physical form is obtained (System 1).

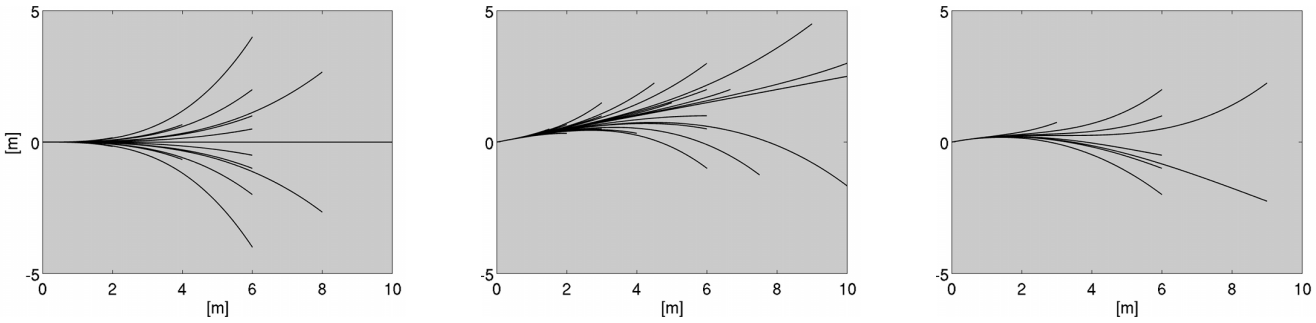

Fig. 3. 24-directional input and state lattice displayed for several initial conditions. Heading levels are unevenly spaced. Steering angle discretization is heading dependent. $(.)_\mathrm{i}$ denotes initial values. **Left:** $(\theta_\mathrm{i}, \phi_\mathrm{i}) = (0, 0)$, **Center:** $(\theta_\mathrm{i}, \phi_\mathrm{i}) = (\mathrm{atan}\, 0.25, 0)$, **Right:** $(\theta_\mathrm{i}, \phi_\mathrm{i}) = (\mathrm{atan}\, 0.25, -0.39)$.

System 1 (Physical Form)**.**

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \tan\phi/l \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v_2$$

B. Chained Form Coordinate Frame

The (2,4) single chain form (as given in System 2) describes the kinematics of of a car-like vehicle in an alternate coordinate frame, whose states x_2 and x_3 do not exhibit direct physical meaning. The representation's usefulness is mainly predicated on its close to linear structure (which renders it interesting for controller design), and the existence of a coordinate transformation to System 1. It was first introduced by Murray *et al.* [8] as a tool to steer vehicles.

System 2 (2,4 Chained Form - Continuous Time Notation)**.**

$$\begin{aligned} \dot{x}_1 &= u_1 \\ \dot{x}_2 &= u_2 \\ \dot{x}_3 &= x_2\, u_1 \\ \dot{x}_4 &= x_3\, u_1 \end{aligned}$$

We readily verify that System 1 may be brought into $(2, 4)$ chained form using the change of coordinates and input transformations given in Eqn. 4 [8].

$$\begin{aligned} x_1 &= x \\ x_2 &= \tfrac{1}{l} \sec^3\theta \tan\phi \\ x_3 &= \tan\theta \\ x_4 &= y \\ v_1 &= u_1 / \cos\theta \\ v_2 &= u_2\, l\, \cos^3\theta \cos^2\phi\, - \\ &\quad\; 3\, \tfrac{u_1}{l} \sin\theta \sin^2\phi \cos^2\theta \end{aligned} \tag{4}$$

C. Path Coordinate Frame

The third coordinate frame relevant to this paper, commonly referred to as path coordinate frame (consult System 3 and Fig. 2, right for a description of the variables used), has been popular in vehicle control applications (see i.e. [9]). In this formulation, the problem of stabilizing a robot onto a reference path s (which is fully defined through its curvature $c(s)$) is reduced to the regulation of the states $[d, \theta_p, \phi]^T$ to zero, irrespective of the path's shape.

System 3 (Path Form)**.**

$$\begin{bmatrix} \dot{s} \\ \dot{d} \\ \dot{\theta}_\mathrm{p} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \frac{\cos\theta_\mathrm{p}}{1 - d\, c(s)} \\ \sin\theta_\mathrm{p} \\ \frac{\tan\phi}{l} - \frac{c(s)\cos\theta_\mathrm{p}}{1 - d\, c(s)} \\ 0 \end{bmatrix} \cdot v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \cdot v_2$$

The inputs (v_1, v_2) of System 3 again take the form of $(v_\mathrm{car}, \dot{\phi})$. Similar to the case with physical coordinates, path coordinates may also be transformed into (2,4) chained form via the state and input transformations given in Eqn. 5 [9].

$$\begin{aligned} x_1 &= s \\ x_2 &= \tfrac{(1 - d\, c(s))^2 \tan\phi}{l \cos^3\theta_\mathrm{p}} - c'(s)\, d \tan\theta_\mathrm{p}\, - \\ &\quad\; c(s)\, (1 - d\, c(s)) \tfrac{1 + \sin^2\theta_\mathrm{p}}{\cos^2\theta_\mathrm{p}} \\ x_3 &= (1 - d\, c(s)) \tan\theta_\mathrm{p} \\ x_4 &= d \\ v_1 &= \tfrac{1 - d\, c(s)}{\cos\theta_\mathrm{p}}\, u_1 \\ v_2 &= \alpha_2 (u_2 - \alpha_1 u_1) \\ \alpha_1 &= \tfrac{\delta x_2}{\delta s} + \tfrac{\delta x_2}{\delta d} (1 - d\, c(s)) \tan\theta_\mathrm{p}\, + \\ &\quad\; \tfrac{\delta x_2}{\delta \theta_p} \left(\tfrac{\tan\phi\, (1 - d\, c(s))}{l \cos\theta_\mathrm{p}} - c(s) \right) \\ \alpha_2 &= \tfrac{l \cos^3\theta_p \cos^2\phi}{(1 - d\, c(s))^2} \end{aligned} \tag{5}$$

Note that Eqn. 5 strongly depends on the selected path and its derivative. c(s) is thus required to be at least C^1 continuous.

III. REACHABLE SETS OF THE KINEMATIC SYSTEMS IN CHAINED AND PHYSICAL FORM

The starting point of this section is constituted by the work of Bicchi *et al.* on input lattices [6]: specifically, we draw on their important findings that the reachable set of the chained form System 2 is discrete and forms a lattice in state-space iff the input set fulfills certain conditions (detailled in Theorem 1). By computing the systems's reachable set explicitly, we obtain a relation between the innput set's sampling levels and the chained form system's state-space discretization (Sec. III-A). Further, by applying the transformation between chained and physical form coordinates from the preceding section (Eqn. 4), we arrive at a mapping between discretization levels in physical form state-space and sampling levels in chained form input space (Sec. III-B). Our lattice design algorithm (Sec. IV) builds on this result.

Fig. 4. **Top Left:** Google Street View scene close to Bellevue, Zurich. **Bottom Left:** corresponding Google Maps image. Impassable areas are shaded gray. **Bottom Center:** 2D heuristic map. Color indicates cost to goal. **Right:** plan from start- (green) to goal-configuration (black). The lattice shape (at $d = 0$) is displayed in red. Note that the curvature is set to zero in the parking lot area due of lack of structure in close vicinity. Instead, the direction is aligned with the dominant orientation of the lot area. The solution on the deformable lattice is represented in blue, the one on the fixed lattice in yellow.

A. Reachability of the Chained Form System 2

Definition 2. *The* reachable set *of a system S at initial configuration q_i is comprised of the set of configurations $q \in \mathcal{Q}$ which are attained by incrementally applying a set of allowable inputs $u \in \mathcal{U}$ to q_i and its successor configurations. The reachable set may form a tree (continuous) or a lattice (discrete) in the state-space of S.*

Theorem 1. *The reachable set of System 2 is discrete and forms a lattice in state-space iff the input set $\mathcal{U}$, with $u = [u_1, u_2]^T \in \mathcal{U}$ piecewise constant, is selected as*

$$\mathcal{U} = \{diag(\lambda)\, Ws,\ s \in S\} \subset \mathbb{R}^2$$

with $W \in \mathbb{Q}^{2x2}$ an invertible matrix, $\lambda \in \mathbb{R}^2$ and $S \subset \mathbb{Z}^2$.

Compiled from [6], Theorem 9, Defs. 4 & 8.

The reachable set of System 2 may therefore only form a lattice, if all control inputs $[u_1, u_2]^T$ assume piecewise constant values. Hence, we may restate System 2 in discrete, exact unit-time sampled notation [6].

System 4 (2,4 Chained Form - Discrete Time Notation)**.**

$$\begin{aligned} x_1^+ &= x_1 + u_1 \\ x_2^+ &= x_2 + u_2 \\ x_3^+ &= x_3 + x_2 u_1 + \tfrac{u_1 u_2}{2} \\ x_4^+ &= x_4 + x_3 u_1 + \tfrac{x_2 u_1^2}{2} + \tfrac{u_1^2 u_2}{6} \end{aligned}$$

Incremental step size in the reachable set of System 4 turns out to be a function of the greatest common divisor of all $u \in \mathcal{U}$, which we denote as u_d. We may thus specify $u_\mathrm{d} = [u_{1\mathrm{d}}, u_{2\mathrm{d}}]^T$ directly by selecting λ, W, and S

$$\lambda = \begin{bmatrix} u_{1\mathrm{d}} \\ u_{2\mathrm{d}} \end{bmatrix}, \qquad W = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad S = \mathbb{Z}^2,$$

so that u_d is itself part of the set $\mathcal{U}$. The reachable set of System 4 is then easily computed (Eqn. 6).

B. Reachability of the Physical Form System 1

Through Eqn. 4 we are able to relate the reachable set in chained form (Eqn. 6)

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \in \begin{bmatrix} \mathcal{X}_1 \\ \mathcal{X}_2 \\ \mathcal{X}_3 \\ \mathcal{X}_4 \end{bmatrix} \subset \begin{bmatrix} k_1\, u_{1\mathrm{d}} \\ k_2\, u_{2\mathrm{d}} \\ k_3\, \frac{u_{1\mathrm{d}}\, u_{2\mathrm{d}}}{2} \\ k_4\, \frac{u_{1\mathrm{d}}^2\, u_{2\mathrm{d}}}{6} \end{bmatrix} \forall k_i \in \mathbb{Z} \qquad (6)$$

to the much more complex reachable set in physical coordinates, given by Eqn. 7.

$$\begin{bmatrix} x \\ y \\ \theta \\ \phi \end{bmatrix} \in \begin{bmatrix} \mathcal{X} \\ \mathcal{Y} \\ \mathcal{T} \\ \mathcal{P} \end{bmatrix} \subset \begin{bmatrix} k_1\, u_{1\mathrm{d}} \\ k_4\, \frac{u_{1\mathrm{d}}^2\, u_{2\mathrm{d}}}{6} \\ \mathrm{atan}(k_3\, \frac{u_{1\mathrm{d}}\, u_{2\mathrm{d}}}{2}) \\ \mathrm{atan}(k_2\, u_{2\mathrm{d}}\, l \cos^3\theta) \end{bmatrix} \qquad (7)$$

Unfortunately, the transformation between physical coordinates and chained form (Eqn. 4) is subject to some restrictions: for uniformly sampled x_3 in chained form (System 2), the heading discretization in physical coordinates (System 1) becomes denser towards $\pm\pi/2$, where the transformation is not defined due to a singularity. Previous work with chained form systems has therefore been limited to problems where the heading could be constrained to the interval $(-\pi/2, \pi/2)$, as in autonomous vehicle parking applications [10], [11]. For generic motion planning problems, such a restriction cannot be imposed, however.

In order to overcome this issue, and thus expand the reachable set of Eqn. 7 to heading levels outside the interval $(-\pi/2, \pi/2)$, we introduce a second *physical form* coordinate system, rotated counter-clockwise by $\pi/2$ with respect to the original one. Variables in the rotated frame are denoted with $(.)_\mathrm{rot}$. The challenge then lies in *merging* the two distinct reachable sets via mutually reachable states. For the heading levels, this is enforced via Eqn. 8 for some $m, n \in \mathbb{Z}$.

$$\theta \overset{!}{=} \tfrac{\pi}{2} - \theta_\mathrm{rot} \Rightarrow \mathrm{atan}(m\, \tfrac{u_{1\mathrm{d}} u_{2\mathrm{d}}}{2}) \overset{!}{=} \tfrac{\pi}{2} - \mathrm{atan}(n\, \tfrac{u_{1\mathrm{d}} u_{2\mathrm{d}}}{2}) \qquad (8)$$

Lemma 2. $\operatorname{atan}\kappa + \operatorname{atan}(1/\kappa) = \pi/2 \;\; \forall \kappa > 0$
Proof : $\frac{d\operatorname{atan}\kappa}{d\kappa} + \frac{d\operatorname{atan}(1/\kappa)}{d\kappa} = \frac{1}{\kappa^2+1} - \frac{1}{(\kappa^{-2}+1)\,\kappa^2} \equiv 0$
$\pi/2$ is obtained through insertion.

By rearranging Eqn. 8 appropriately, the same structure as in Lemma 2 is obtained. We conclude that $\forall\theta \neq 0$, $\theta \in \mathcal{T}$ iff $\frac{n\,u_{1\mathrm{d}}u_{2\mathrm{d}}}{2} = \frac{2}{m\,u_{1\mathrm{d}}u_{2\mathrm{d}}}$ for some $m, n \in \mathbb{Z}$ and artificially construct a merged reachable set which is comprised of states that are mutually reachable from both coordinate frames only. It follows from Eqn. 7 that the number of heading levels in this merged reachable set is a function of $u_{1\mathrm{d}}\,u_{2\mathrm{d}} = u_{\mathrm{comb}}$.

A similar argument holds for the number of mutual steering angles ϕ: likewise to heading, the discretization in steering angle ϕ is more densely sampled towards $\theta = \pm\pi/2$, and thus becomes configuration dependent. In cases where steering angle saturation needs to be considered, it is thus of interest to compute the minimal number of discrete values obtainable over all heading levels. By inserting the largest attainable steering angle $\phi_{\max}$ into Eqn. 4:

$$\phi_{\max} \overset{!}{\geq} \operatorname{atan}\left(x_2\, l \cos^3\theta\right), \tag{9}$$

it can be shown that this number always occurs at $\theta = 0$.

IV. DEFORMABLE INPUT- / STATE-LATTICE DESIGN PROCEDURE

Now that we have introduced all essential auxiliary parts, let us focus on the development of our main contribution, an algorithm for the simple yet effective design of deformable input- and state-sampled lattice graphs. The procedure draws heavily from the preceding sections: it builds on the work of Bicchi *et al.* [6] on input sampled lattices for chained form systems (Theorem 1) and on Eqn. 7 which relates the reachable set in physical coordinates to the design parameters u_{d} in chained form. Essentially, we add an additional layer on top of the approach of Bicchi *et al.* [6], so that their algorithm allows us to build a lattice in physical form coordinates based on physical form specifications directly.

A. Lattice Design Algorithm

In Bicchi's method [6], a lattice spanning the four-dimensional chained form configuration space is constructed by specifying a two-dimensional input set $\mathcal{U}$ only. The approach is thus clearly overconstrained with respect to the obtainable state space discretization: in fact we are restricted to the direct specification of two states' discretization levels in chained form only. The resolution of the remaining two states emerges from the design procedure.

For practical reasons, we are typically interested in specifying a discretization in physical coordinates directly, however. To this end we apply Eqn. 7 to specify $\mathcal{U}$ based on a desired physical form state-space discretization. This forms the starting point of our lattice design algorithm.

As a means of illustration, we specify a discretization in heading (θ) and steering angle (ϕ) directly, although any other combination of two states is equally feasible. (x, y)-discretization then emerges from the design procedure, and needs to be verified a posteriori.

$\mathbf{u}_{\mathrm{comb}}$	# Head.	Heading Discr. Lvls in 1^{st} Octant
2	8	$\{0, \operatorname{atan}1\}$
1	16	$\{0, \operatorname{atan}\frac{1}{2}, \operatorname{atan}1\}$
1/2	24	$\{0, \operatorname{atan}\frac{1}{4}, \operatorname{atan}\frac{1}{2}, \operatorname{atan}1\}$
1/3	32	$\{0, \operatorname{atan}\frac{1}{6}, \operatorname{atan}\frac{1}{3}, \operatorname{atan}\frac{1}{2}, \operatorname{atan}1\}$

TABLE I
HEADING LEVELS AS A FUNCTION OF THE PARAMETER $\mathrm{u}_{\mathrm{comb}}$

Algorithm 1 (Input- and State-Lattice Design)**.**

1) *Gather vehicle specific information, notably vehicle wheelbase (l) and maximal steering angle ($\phi_{\max}$).*
2) *Decide on a desired number of heading levels θ in physical coordinates and consult Table I or Eqn. 8 to obtain the corresponding value of $u_{\mathrm{comb}} = u_{1\mathrm{d}}\,u_{2\mathrm{d}}$.*
3) *Decide on a minimal number of steering angle levels. As shown above, this value will be encountered at $\theta = n\,\pi/2,\ \forall n \in \mathbb{Z}$. Through Eqn. 9, a discretization in x_2, and thus in u_2 is obtained.*
4) *The discretization of x_1 is obtained by inserting u_2 into u_{comb} and solving for u_1. The discretization in x_4 results from Eqn. 6.*
5) *For each initial chained form state of the chosen discretization $[x_{1\mathrm{i}}, x_{2\mathrm{i}}, x_{3\mathrm{i}}, x_{4\mathrm{i}}]^T \in [\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3, \mathcal{X}_4]^T$ (with $(x_{1\mathrm{i}}, x_{4\mathrm{i}}) = (0, 0)$ due to translational invariance), apply all inputs $u \in \mathcal{U}$ that are part of the input sampling to arrive at a set of configuration dependent outgoing edges. They form the lattice.*
6) *Remove edges from the lattice which start or end at heading / steering angle levels, that are not part of both the original and the rotated coordinate frame.*

Via Algorithm 1 we designed a 24-directional lattice with at least seven discrete steering angles (occurring at $\theta = 0$), and an (x,y)-discretization of $\frac{1}{12}$m for a 2008 Toyota Prius ($l = 2.7\,\mathrm{m}$, $\phi_{\max} = 0.6\,\mathrm{rad}$, displayed in Fig. 3).

Despite the apparent focus on car-like vehicles, it should be noted that the theory developed in this paper equally applies to all other systems which can be transformed into (2,n) chained form. A sufficient algorithm is described in [8].

B. Geometrical Properties of the Lattice

Our lattice design procedure requires piecewise constant inputs (u_1, u_2) in chained form. Thus x_1 assumes piecewise linear and x_4 piecewise cubic values. By referring to the transformation between chained form and physical form coordinates (Eqn. 4), we observe an equivalence between (x_1, x_4) and (x, y). It follows trivially that lattice segments in physical coordinates are composed of piecewise cubic polynomials.

$$y(x) = ax^3 + bx^2 + cx + d$$

Remarkably, Algorithm 1 manages to generate lattice segments which satisfy the six constraints (difference in 2D position, initial and final heading and curvature) imposed through System 2 by using cubic polynomials alone.

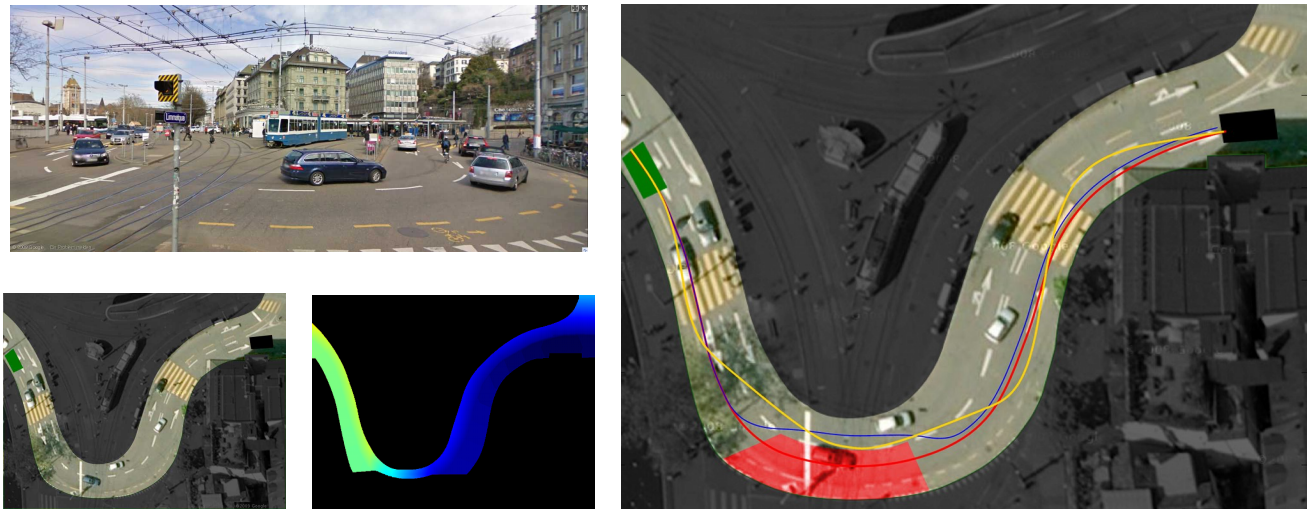

Fig. 5. **Top Left:** Google Street View scene of Central, Zurich. **Bottom Left:** corresponding Google Maps image. Impassable areas are shaded gray. **Bottom Center:** 2D heuristic map. Color indicates cost to goal. Note, that moving along the left lane is costlier. **Right:** plan from start- (green) to goal-configuration (black). A vehicle with break-down (shaded red) needs to be overtaken. The lattice shape (at $d = 0$) is displayed in red, the optimal solution on the deformable lattice in blue, and on the globally fixed lattice in yellow.

C. Lattice Deformation along Path c(s)

It is well known that lattice graphs are ill suited to applications in structured environments, as their heading discretization remains globally fixed. Rather small heading discretization levels are thus required to avoid the typical swerves in directions that are not part of the lattice. These additional heading levels affect search speed dramatically.

The concept of path coordinates (as compiled in Sec. II) is easily misappropriated to solve the above mentioned issue, however. Lattice segments (constructed with Algorithm 1) may be transformed from chained form into path coordinates via Eqn. 5, and the *specification of a reference path c(s)*. The key point is then to shape c(s) along the local structure of the environment. Notably, this result allows us for the first time to freely align the orientation of the graph's *heading* levels with the environment (provided such information can be extracted). In structured environments, sensible choices include the center lines along road lanes and corridors. In unstructured environments, the scene's local principal directions could be chosen. Once designated, the reference path $c(s)$ then constructs the lattice segments with $d \equiv 0$. Segments with $d \neq 0$ are constructed so that d remains perpendicular on the path (illustrated in Fig. 2, right).

As a consequence to the liberty in choosing $c(s)$, several auxiliary steps need to be performed during graph search in order to ensure compliance with vehicle constraints, however.

- During edge expansion, every successor edge is required to be checked for maximal curvature violation (corresponding to the maximal allowed steering angle $\phi_{\max}$). If a violation occurs, the edge may not be expanded.
- During expansion, any edges with distance from path $d(s) > 1/c(s)$ may not be expanded (degenerate case).
- Due to distortion effects $c(s)$ imposes on lattice edges, cost-map cells traversed by edges cannot be precomputed. Their calculation in real-time is thus required.

V. EXPERIMENTAL RESULTS

In this section we apply our novel deformable lattice to two difficult outdoor environments: the Zurich *Bellevue* scene contains both structured (road) and unstructured (parking lot) parts, the *Central* scene is comprised of a curved double lane segment, where a lane change needs to be performed due to an immobile car (see Fig. 5). Both scenes have been prepared manually from Google Maps images: static obstacle areas are shaded dark, lane-centers are marked in red. In both scenarios we compare our deformable lattice to the same lattice with globally fixed heading discretization. We employ an adapted version of the A* graph search algorithm to return minimal cost paths on the lattice graph. The underlying cost function attains low values on driveable terrain with slowly growing values towards the road boundaries. Obstacle areas are assigned infinite cost, and are thus impassable.

A. Bellevue Environment

Fig. 4 shows an overview of a scene close to Bellevue, Zurich and describes the different steps taken before planning: masking of obstacles and untraversable areas, generation of a 2D heuristic, labeling of unstructured environments (the parking lot is shaded in red) and centerline extraction. In blue color, the solution obtained via our novel deformable lattice is displayed. During on-road operation, we shape the lattice along the center of the lane. Once the parking lot is reached, it is shaped along the lot's dominant direction. On the road segment, the solution follows the lane center perfectly, as no obstacles need to be avoided. In the parking lot, it eventually departs the lattice centerline to park the vehicle safely in the designated spot. Contrary to that, the solution obtained with the globally anchored lattice (in yellow color) produces an excessively curved path which is typically not well aligned with nearby structures.

B. Central Environment

Fig. 5 provides an overview for the experiment at Central, Zurich. This scenario exemplifies how our deformable lattice performs on-road, where the preferred lane needs to be departed (in the present case due to a vehicle with breakdown, shaded red in Fig. 5, right). The robot is thus required to move to the left lane, overtake the stationnary vehicle and then eventually return to the right lane. The time of change, and the duration spent on the left lane is not chosen by the lattice shaping a priori, but rather by the graph search through a minimal cost solution. In the same environment, we performed a search using the globally fixed lattice. The result is a meandering path, which just barely manages to remain within the lane boundaries.

C. Discussion

With the Bellevue and Central experiments we confirmed that in curved structured environments, and in narrow environments where the orientation is not well aligned to a lattice heading, globally anchored lattices perform poorly: failure to compute a solution or, in case of success, excessively curved paths tend to emerge. By contrast, we were able to demonstrate that deformable lattices can not only be employed in unstructured environments, but in virtually any scenario (provided that, through i.e. distance transforms, the local structure or general orientation can be gathered), as they allow for the dynamic adaptation of the heading dimension during search. Compared to an anchored lattice, superior performance can thus be achieved with far fewer discrete heading levels, thereby reducing the graph's branching factor and increasing search speed. We believe it to be the first time, that high-dimensional graph search on a state lattice was employed as a universally applicable planning algorithm. Through our contribution the need for a higher-level reasoning and switching module thus vanishes. By removing the superfluous host of disparate, environment specific planning solutions, the navigation system becomes simpler, more robust and easier to optimize and operate.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a novel algorithm to generate input-/state-lattices comprised of cubic splines with a user specified discretization in state space. The method is applicable to vehicles whose kinematic model is mathematically equivalent to the (2,n) chained form, and thus includes unicycles, bicycles, differential-drive robots and cars. Furthermore we presented a solution to bend such lattices along any C^1 smooth path, thereby allowing us to align the heading discretization of the lattice dynamically along the skeleton of arbitrary structured environments. We then demonstrated for the first time the applicability of this planning approach to both structured and unstructured environments.

In the future, we would like to investigate whether our lattice design approach could be extended to dynamic vehicle models and non-planar environments.

ACKNOWLEDGMENT

This work has been partially supported by the Swiss National Science Foundation NCCR 'MC3' and by the European Project 'EUROPA' under contract number FP7-231888.

REFERENCES

[1] D. Ferguson, T. Howard, and M. Likhachev. Motion Planning in Urban Environments: Part I. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008.

[2] T. Howard D. Ferguson and M. Likhachev. Motion Planning in Urban Environments: Part II. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008.

[3] A. Kushleyev and M. Likhachev. Time-bounded lattice for efficient planning in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[4] M. Likhachev and D. Ferguson. Planning long dynamically-feasible maneuvers for autonomous vehicles. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2008.

[5] D. Dolgov and S. Thrun. Detection of principal directions in unknown environments for autonomous navigation. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2008.

[6] A. Bicchi, A. Marigo, and B. Piccoli. On the reachability of quantized control systems. *IEEE Transactions on Automatic Control*, 4(47):546–563, April 2002.

[7] R. A. Knepper M. Pivtoraiko and A. Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(1):308–333, March 2009.

[8] R. Murray and S. S. Sastry. Nonholonomic motion planning: Steering using sinusoids. *IEEE Transactions on Automatic Control*, 38:700–716, 1993.

[9] A. De Luca, G. Oriolo, and C. Samson. Feedback control of a nonholonomic car-like robot. *Springer, Berlin*, 1998.

[10] S. Pancanti, L. Pallottino, D. Salvadorini, and A. Bicchi. Motion planning through symbols and lattices. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[11] M. Rufli. Driver-in-the-loop path control for a non-holonomic vehicle. *Bachelor Thesis, ETH Zurich*, 2006.