2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Deformed State Lattice Planning

Zhongqiang Ren[1], Chaohui Gong[2] and Howie Choset[2]

***Abstract*— Search-based planning that uses a state lattice has been successfully applied in many applications but its utility is limited when confronted with complex problems represented by a lattice with many nodes and edges with high branching factor. However, in many seemingly complex problems, proper "form-fitting" can reduce the number of nodes and edges needed to represent the problems, provides a concise state lattice and therefore simplifies the computation. This paper proposes a planning framework which strikes to identify concise representations of problems, creates such "form-fitting" state lattice on which a more concise search can take place. In a sense, we take a conventional state lattice and map it onto a deformed space, and then the motion primitives and heuristics follow. Since the contribution of the paper is not the search approach but rather the means by which the lattice is deformed, any search-based planner can then be easily changed to a corresponding deformed version with no increase in time complexity. This paper demonstrate the benifits of the approach which includes 1) planned path can be followed with few changes in motion primitives and thus can provide global smoothness of planned path; 2) fewer states are expanded and thus shorter time to search solution in state space is required, and 3) fewer states are expanded and thus less memory is required to save the state lattice. We demonstrate the benefit of the proposed approach in illustrative toy examples, as well as robot experiments.**

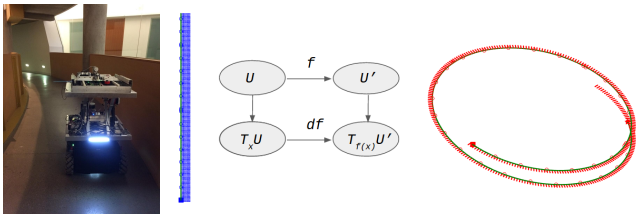
I. INTRODUCTION

Search-based motion planning methods, most notably A* [8] and its variants [6], [7] as well as state-lattice planning [2], have been successfully applied in many applications. However, many complex problems require a state lattice (see Section II for detailed description) with a fine spatial resolution, a large number of states and large branching factors to ensure enough states and edges lie in the free space, which in turn leads to rapid growth in planning time and memory consumption [10]. For example, as shown in figure 1, a naive state representation of a multi-layer elliptical helix corridor using a three-dimensional state lattice will occupy lots of memory and make it difficult for search-based planning algorithms.

To remedy the situation, great efforts have been invested in designing heuristics in order to guide the search along some low dimensional space. Rayner et al. [3] embeds state space into an Euclidean space by using manifold learning techniques, while Gochev et al. [4] plans with adaptive dimensionality.

[1] Zhongqiang Ren is with the Department of Mechanical Engineering and the Robotics Institute, Carnegie Mellon University, zhongqir@andrew.cmu.edu

[2] Chaohui Gong and Howie Choset are with the Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA, USA chaohuig@cmu.edu & choset@cmu.edu



(a) (b) (c) (d)

Fig. 1: Real robot implementation on an elliptical helix. (a) Mobile robot moveing on the elliptical helix corridor. (b) The regular state lattice. (c) Mapping diagram between the regular space and deformed space (d) The deformed state lattice and planned path.

Typically, for many robot applications, it is rather straightforward to define a state lattice based on a Euclidean parameterization of the free space. The approach described in this paper deforms such a state lattice representation to fit a proper submanifold of the free space, thereby directly producing a more concise representation of the environment. Using the aforementioned example, as shown in Figure 1, three-dimensional lattice representation of this space would require many nodes and edges to capture the free space of environment, i.e., the points on the two-dimensional helix. However, the surface of the elliptical helix is a 2 dimensional manifold and topologically equivalent to a rectangular space. One can therefore map a state lattice, defined on the two-dimensional rectangle, onto a state lattice that naturally embeds into the two-dimensional helix, which reduces computation burden and produces smoother paths for the robot to follow.

To formalize the idea, in this work, we propose a planning framework called Deformed State Lattice Planning (DSLP), which is composed of several steps: 1) find a proper low dimensional form-fitting state lattice representation of the environment, 2) deform state lattice, motion primitives, costs and heuristics and 3) perform a deformed search-based planner on the low dimensional space.

II. RELATED WORKS

A. State lattice planning

A state lattice [1], [2] is a set of states and connections between states, where the states are obtained by discretizing the configuration space and the connections between states are feasible paths generated by taking the nonholonomic constraints of the system into consideration and is sometimes called motion primitives. With state lattices, a motion

planning problem is converted to a graph search problem and the optimal path can then be found on the graph with a heuristic search planner like A* or its variants.

Although state lattice planning is resolution complete, it becomes inefficient when representing problems with large state lattices with fine resolution and large branching factor.

B. Heuristic design

To overcome the high resolution large state lattice representation problem, lots of work have been done to design effective heuristic to guide the search. Euclidean heuristics (EH) [3] and its variants [5] are among this class. EH embed the state space graph into an Euclidean space using manifold learning techniques and use the Euclidean distance as heuristics for the state space search to acquire faster speed and memory efficiency.

However, EH requires a lot of computational memory and time to embed states into an Euclidean space off-line, while DSLP amortizes the effort into every search iteration after having the mapping function and the time complexity of the planner remains the same.

III. Method

In this section, we first propose a general framework for DSLP and later illustrate concepts with several examples including the set of rigid body transformation, circular sector and elliptical helix. As we are primarily interested in mobile robots, we assumed the state space is either SE(3) or SE(2), but the proposed framework can be easily applied to configuration space with arbitrary number of dimensions.

To further clarify, we list our definitions here. We define a *regular state lattice (regular space)* to be a uniformly distributed state lattice as in previous work [1], which is exemplified in Figure 1b. We refer a *deformed state lattice (deformed space)* to be the set of discretized configurations and motion primitives that is mapped from regular state lattice, which is shown in Figure 1d. We use symbols without prime, for example $\mathbf{x}$, to indicate variable or vector that lies in regular space and primed symbols, for example $\mathbf{x}'$, to indicate variable or vector that lies in deformed space.

A. Deformation of a state lattice

Assume we have a smooth mapping function $f : \mathbf{x} \mapsto \mathbf{x}'$, where $\mathbf{x} \in U$, $\mathbf{x}' \in U'$, $U' \subseteq SE(3)$ and U is a diffeomorphism of U'. This function will map a state vector $\mathbf{x}$ in regular space U to a state $\mathbf{x}'$ in a deformed space U', which is a manifold, by $\mathbf{x}' = f(\mathbf{x})$.

f is required to be smooth, thus, the differential of f, also called Jacobian, is a map $df : T_xU \to T_{x'}U'$, where T_xU and $T_{x'}U'$ are the tangent spaces at $\mathbf{x}$ and $\mathbf{x}'$ in U and U' respectively, as shown in Figure 1c. And motion primitive is defined as a map $m : [0, 1] \to U$. Its time derivative $\dot{\mathbf{m}}(t) \in T_xU$, can be mapped to $\dot{\mathbf{m}}'(t) \in T_{x'}U'$, by $\dot{\mathbf{m}}'(t) = df\,(\dot{\mathbf{m}}(t))$.

Evaluating the cost of a path is essential in search based planning. Here we define the cost of motion primitive in regular space by a positive definite matrix M as

$$C = \int_0^T \boldsymbol{\xi}(t)^T M \boldsymbol{\xi}(t) dt \tag{1}$$

$$\boldsymbol{\xi}(t) = \left(\mathbf{m}(t)^{-1}\dot{\mathbf{m}}(t)\right)^{\vee} \tag{2}$$

where $\boldsymbol{\xi}(t) \in se(3)$ and $\vee$ is unhat operator.

After the deformation, the cost of deformed motion primitive becomes

$$C' = \int_0^T \boldsymbol{\xi'}(t)^T M \boldsymbol{\xi'}(t) dt \tag{3}$$

$$\xi'(t) = \left(f\left(\mathbf{m}(t)\right)^{-1} df\left(\dot{\mathbf{m}}(t)\right)\right)^{\vee}. \tag{4}$$

B. Deformation of nonholonomic constraints and heuristic

Many mobile robots are subject to physical constraints that limits the allowable directions of motion. These constraints can be handily implemented as Pfaffian constraints in the form of

$$\omega'(\mathbf{x}')\dot{\mathbf{x}}' = \mathbf{0}, \tag{5}$$

where ω' indicates that the constraints are in deformed space, which is the space where robot motions are executed. These constraints equations can also be mapped back to regular space correspondingly by $\omega'(f(\mathbf{x}))df(\dot{\mathbf{x}}) = 0$, which can be formulated in regular space as

$$\omega(\mathbf{x})\dot{\mathbf{x}} = 0 \tag{6}$$

$$\omega\left(\mathbf{x}\right) = \omega'\left(f\left(\mathbf{x}\right)\right) df. \tag{7}$$

If we want to design the motion primitives in regular space but executed in deformed space, then we can use equation (7) to design motion primitives in regular space to make sure that the nonholonomic constraints in deformed space is fulfilled. If we already have a set of motion primitives in regular space, we can use equation (5) to check if the motion primitives are valid. If not, then the state lattice will lose the connection between the two states connected by that motion primitive after deformation.

The heuristic also need to be mapped between two space. In previous work, a common choice of heuristic for a state lattice planning problem is the Euclidean distance between two states $h(\mathbf{x}) = \|\mathbf{x} - \mathbf{x_{goal}}\|$. In DSLP, we need to plan motion in deformed space and the heuristic becomes the Euclidean distance between two deformed states

$$h(f(\mathbf{x})) = \|f(\mathbf{x}) - f(\mathbf{x_{goal}})\|. \tag{8}$$

This heuristic is an underestimate for the optimal path cost from state $\mathbf{x}$ to $\mathbf{x_{goal}}$ and thus guarantee the optimality of the planner, but this heuristic may have modest guiding power. More discussion about heuristic can be found in later section.

C. Examples

1) Deformation on a circular sector: Nonlinear mapping function $f : (x, y, \theta) \mapsto (x', y', \theta')$ will map a rectangle state lattice to a circular state lattice. We define some deformation parameters:

x_c, y_c - the deformation center in rectangular state lattice,

β_s, β_e - circular starting and ending angle,
H - grids length,
r_c - the deformation radius,
β_c - the angle of deformation center,
γ - the angle resolution,
where

$$r_c = \sqrt{x_c^2 + y_c^2}, \quad \beta_c = \frac{\beta_s + \beta_e}{2}, \quad \gamma = \frac{\beta_e - \beta_s}{H - 1}. \tag{9}$$

Now we can give our deformation function $\mathbf{x}' = (x', y', \theta') = f(x, y, \theta), \ \mathbf{x}, \mathbf{x}' \in SE(2)$ based on these parameters

$$\begin{aligned} x' &= (r_c + x - x_c)\cos(\beta_c + (y - y_c)\gamma) \\ y' &= (r_c + x - x_c)\sin(\beta_c + (y - y_c)\gamma) \\ \theta' &= \theta + \arctan 2(y', x') = \theta + \beta_c + (y - y_c)\gamma. \end{aligned} \tag{10}$$

Now we can compute the deformation of motion primitives from a rectangular state lattice to a circular state lattice using equation (4), where df is.

$$df|_x = \begin{bmatrix} \cos(\beta_y) & -r_x \sin(\beta_y) & 0 \\ \sin(\beta_y) & r_x \cos(\beta_y) & 0 \\ 0 & \gamma & 1 \end{bmatrix} \tag{11}$$
$$\beta_y = \beta_c + (y - y_c)\gamma, \quad r_x = \gamma(r_c + x - x_c)$$

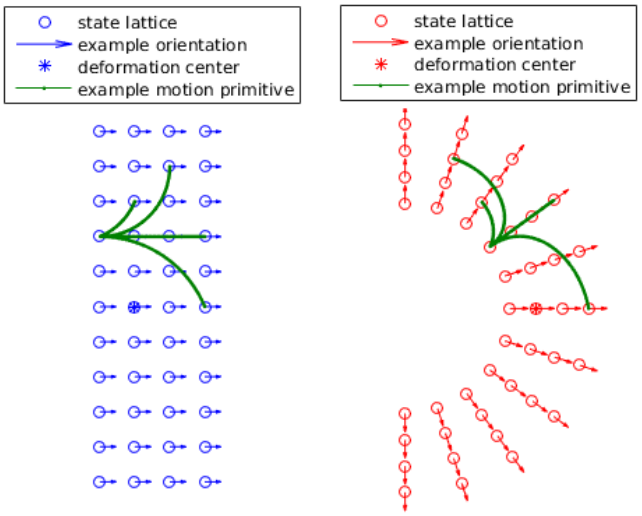


Fig. 2: Example of deformation of state lattice and motion primitives. The state lattice in red is the regular rectangular one with orientation of zero radian while the state lattice in blue is the corresponding deformed one with same orientation. The green curve in both sub-figures show some motion primitives before and after deformation. For a differential drive car, among the motion primitives shown in figure, only the straight move and turn in place motion primitives are still valid after deformation.

The Pfaffian constraint for differential drive car in deformed space is $\omega'(\mathbf{x}') = (\sin\theta', -\cos\theta', 0)$ and the one in regular space can be computed by multiplying it with $df|_x$. In this case, as shown in Figure 2, some pre-designed motion primitives in regular state lattice will become invalid after deformation.

D. Deformed A* Planner

The deformed A* planner is shown in Algorithm 1, which is same as the original A* planner with the only differences in $getCost()$ and $getHeuristic()$ functions. In original A* planner, $getCost()$ returns a constant cost related with the motion primitive by looking up a pre-computed table and takes constant time. Here, $getCost()$ uses Equation (3) and takes constant time by discretizing the integral into a sum of N parts, where N is constant. Similarly, in original A* planner, $getHeuristic()$ computes the heuristic using Equation (8) or 14 in later section. Both can be computed in constant time.

Algorithm 1 Deformed A* Planner

```
1:  procedure EXPAND(state)
2:      for each motion primitive neightbor n of state do
3:          g_new ← g(state) + getCost(state,n)
4:          if g_new < g(n) then
5:              g(n) ← g_new
6:              parent(n) ← state
7:          f(n) ← g(n) + getHeuristic(n, goalState)
8:
9:  procedure A* SEARCH(startState, goalState)
10:     parent(startState) ← ∅
11:     g(.) ← ∞
12:     f(.) ← ∞
13:     openList ← {startState}
14:     closeList ← ∅
15:     while openList not empty do
16:         state ← openList.pop()
17:         closeList.insert(state)
18:         if state is goalState then
19:             break
20:         EXPAND(state)
```

Because both changed functions still take constant time, the overall time complexity of the deformed A* planner remains the same as the original A* planner.

IV. Discussion

A. Constraints on motion primitives

If given a designed set of motion primitives in the regular space, equation (5) can help check the feasibility of the corresponding motion primitives in the deformed space. But for nonlinear deformation function f, this equation can rule out a lot of motion primitives and thus reduces the graph's connectivity so that no feasible path might exist from start to goal, as shown in Figure 4.

In our practice of mobile robotics, mapped motion primitives that do not satisfy differential constraints, in other words motion primitives that do not satisfy Equation (5) can actually also be closely followed if feedback control is applied to make the robot follow the path if the violation

is modest. Take differential drive car for example, if the mapped motion primitives have very little lateral velocity, then practically, this motion primitive can be executed with good accuracy with a line of sight controller [11].

Thus, changing nonholonomic constraints of the system to soft constraints can on one hand enrich the connectivity of the graph and make the robot move smoother because of more choice of motion primitives, and on the other hand, make the planner choose the optimal path with lowest cost by taking constraints into consideration.

Taking differential drive car for example, by setting matrix M as

$$M = \begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & c_\theta \end{bmatrix}, \tag{12}$$

where c_y is the cost or penalty for lateral velocity. Instead of setting it ∞, which is rigid constraint, we can set it to be a finite number. This will incorporate the system constraints as soft ones into the planning process.

B. Heuristic design

As mentioned in previous section, one naive way to get heuristic in deformed space is, given a state and a goal state in regular space, computing the mapped states in deformed space and then using the Euclidean distance between these two mapped states as the heuristic, as shown in equation (8). It is an underestimate of the true distance and thus guarantee the optimality of the planned path in deformed space, but the guiding power of such heuristic is reduced because of the relatively large difference between heuristic and true distance and this difference depends on the underlying manifold.

To achieve good guiding power on deformed space, a good heuristic is the length of the geodesic curve ${\tau_{\mathbf{G}}}'$ on the parametrized surface between the given point $\mathbf{p}'$ and goal point $\mathbf{p}'_{\mathbf{goal}}$, which exists if, by Hopf-Rinow theorem [9], the parametrized surface is complete and locally compact. Here point $\mathbf{p}$ is the translational part of the pose in $SE(3)$. The geodesic curve is an underestimate of the optimal path and thus guarantee the optimality of the planner. But this geodesic curve can be hard to compute given $\mathbf{p}$ and $\mathbf{p_{goal}}$.

One alternative heuristic is the length of the image curve ${\tau_{\mathbf{L}}}'(t)$ of the straight line connecting $\mathbf{p}$ and $\mathbf{p_{goal}}$ in regular space $\tau_{\mathbf{L}}(t)$. We use the integration of curvature along ${\tau_{\mathbf{L}}}'$ to measure how far ${\tau_{\mathbf{L}}}'$ deviates from being a geodesic ${\tau_{\mathbf{G}}}'$, which has zero curvature. The formula is

$$\epsilon = \int_0^T |\kappa({\tau_{\mathbf{L}}}'(t))| dt.$$

Such curve ${\tau_{\mathbf{L}}}'$ is an approximation of the optimal path in deformed space. It can overestimate the optimal cost and thus lead to sub-optimal solution. To limit the sub-optimality, we can use ϵ to check if ${\tau_{\mathbf{L}}}'$ is far away from geodesic. If ϵ exceed some thresholds, this heuristic should be rejected or discounted. To compute this heuristic, we first characterize the straight line in regular space as a parameterized curve $\tau_{\mathbf{L}}(t) = (x_L(t), y_L(t), z_L(t)), t \in [0, T]$ and $\tau_{\mathbf{L}}(0) = (x_L(0), y_L(0), z_L(0)) = (x_{state}, y_{state}, z_{state})$, $\tau_{\mathbf{L}}(T) = (x_L(T), y_L(T), z_L(T)) = (x_{goal}, y_{goal}, z_{goal})$. We can map each point along the curve from regular space to deformed space via f. Thus the length of the curve ${\tau_{\mathbf{L}}}'(t)$ in deformed space can be computed as

$$l({\tau_{\mathbf{L}}}'(t)) = \int_0^T \sqrt{\dot{x}'(t)^2 + \dot{y}'(t)^2 + \dot{z}'(t)^2} dt \tag{13}$$

and $l({\tau_{\mathbf{L}}}'(t))$ can be approximated by discretizating the integral

$$l({\tau_{\mathbf{L}}}'(t)) \approx \Sigma_{i=0}^{N} \frac{T}{N} \sqrt{\dot{x}_i'^2 + \dot{y}_i'^2 + \dot{z}_i'^2}. \tag{14}$$

where N can be chosen as a constant. This heuristic can better approximate the true distance than the previous heuristic, as shown in Figure 3, which visualize the heuristic value as the length of the curve between two states.

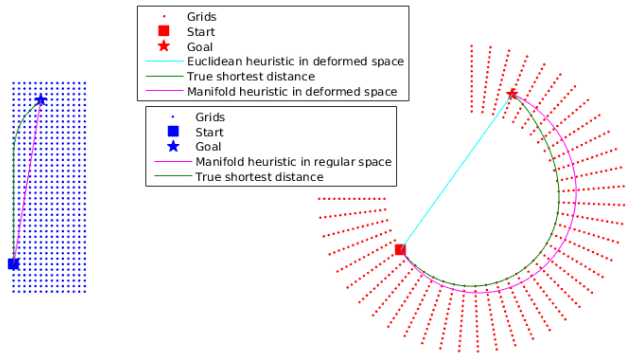


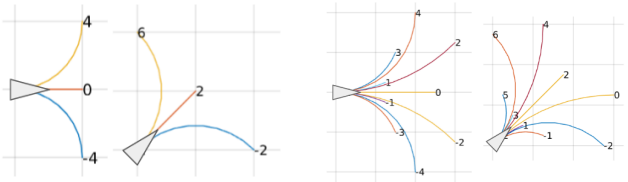
Fig. 3: Example of deformed heuristic. The green curves show the path with shortest cost between the given state and goal state planned with motion primitive sets illustrated on the Figure 4b. The magenta curves are the straight connecting line in regular space and its image in deformed space. The cyan straight line on the right side is the straight connecting line in deformed space.

V. Results

To evaluate the performance of DSLP framework, we compare it with A* on a regular state-lattice. Then we also implemented our framework on a real mobile robot and make it move on an elliptical corridor.

A. Comparison with regular state lattice planning

To numerically show the benefit of explicitly exploiting the form-fitting underlying structure of the environment, we do a comparison with normal search-based planning, where the planner used is A* planner. The environment we want to plan paths in is a semicircular plane with inner radius of 1.5m, outer radius of 2.5m. We choose resolution of 0.05m per grid for the traditional state-lattice planner. As shown in Figure 5, to cover the plane, with DSLP, we need a state lattice of size $X = 20, Y = 81, \Theta = 16$, where X is the width, Y is the height and Θ is the number of discretized angles. For the



(a) (b)

Fig. 4: Examples from two sets of motion primitives. (a)Motion primitives from the first set while (b)Motion primitives from second set. The numbers in the figure show the ending orientations of motion primitives, which range from 0 to 15 representing discrete angles from 0 to 2π. Both sets has turn in place motions which are omitted in the figure.

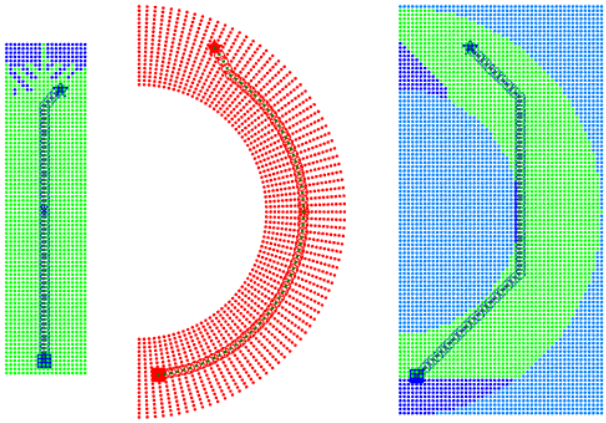


Fig. 5: Comparison with motion primitive set shown in Figure 4a. Only x, y are illustrated here, θ is omitted. The green part shows the states with at least one orientation explored before finding the cheapest path to goal (states searched: left 14718, right 14937). The start pose lies at the lower side of the graph while the goal pose lies at the upper side of the graph. Left figure shows the regular grids, middle one shows the deformed space and the right one shows the normal state lattice planning, where the light blue are obstacles and deep blue shows free space.

traditional planner, the size of the regular state lattice to cover the plane is at least $X = 50, Y = 100, \Theta = 16$, roughly three times as large as the previous one. If the environment gets larger, the deformation strategy will save more memory.

We have done the test with two sets of motion primitives, as shown in Figure 4. In the first set, we have only forward, turn in place and 90 degree arc motions and test results are shown in Figure 5, while in the second set, we have 12 different motions for each orientation and test results are shown in Figure 6.

As shown in the tests, another advantage of DSLP is, the output planned path is generally globally smoother, while the local smoothness is related with motion primitives. From regular state lattice to deformed state lattice, a straight line will be mapped to a curve.

Besides, as shown in Figure 7, we run planners in a narrow

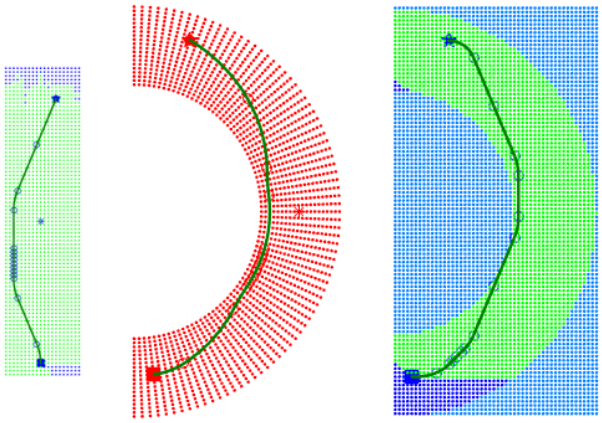


Fig. 6: Comparison with motion primitive set shown in figure 4b. Only x, y dimensions are illustrated here, θ is omitted. The green part shows the states with at least one orientation explored before finding the cheapest path to goal (states searched: left 11745, right 14014). The start pose lies at the lower side of the graph while the goal pose lies at the upper side of the graph. Left figure shows the regular grids, middle one shows the deformed space and the right one shows the normal state lattice planning, where the light blue are obstacles and deep blue shows free space.

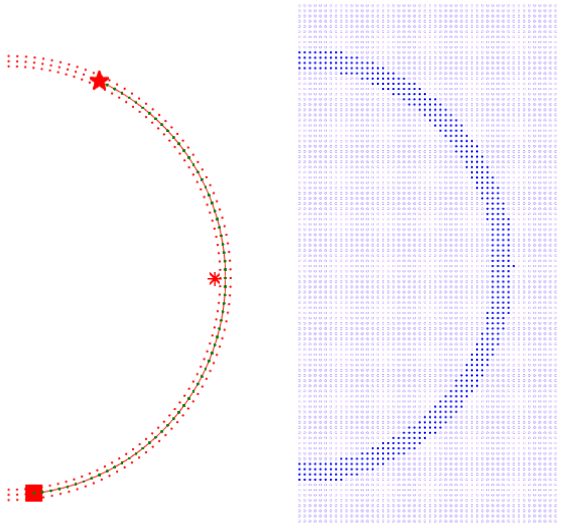


Fig. 7: Comparison between DSLP and normal state lattice planning in a narrow corridor. Taking robot size into consideration, DSLP find a solution while normal state lattice fails unless further increase the resolution of the state lattice.

curved corridor which is almost as wide as the robot size. The normal state lattice planner fails to find a solution unless the resolution of the state lattice increase, while DSLP find a solution after mapping straight motion primitives to curved ones.

B. Real robot implementation on an elliptical helix

To test our DSLP framework, we have also done a test on an elliptical helix narrow corridor in Gates Hillman Center in Carnegie Mellon University.

We use the same definition of deformation parameters r_c, β_c as the circular sector example and add a few more parameters, N, the number of complete layers of helix, z_s, z_e the starting and ending height of the helix. The angle resolution γ needed to be updated with

$$\beta_{s2} = \beta_s, \;\; \beta_{e2} = \beta_e + 2N\pi, \;\; \beta_c = \frac{\beta_{s2} + \beta_{e2}}{2}$$
$$\gamma = \frac{\beta_{e2} - \beta_{s2}}{H - 1}. \tag{15}$$

And we define phase φ of state lattice as $\varphi = \beta_c + \gamma(y - y_c)$. Then, the nonlinear map $f : (x, y, \theta) \mapsto (x', y', z', \theta')$ can be formulated as

$$\begin{aligned} x' &= (r_c + x - x_c)cos(\beta_c + (y - y_c)\gamma) \\ y' &= (r_c + x - x_c)sin(\beta_c + (y - y_c)\gamma) \\ z' &= \frac{\varphi - \beta_{s2}}{\beta_{e2} - \beta_{s2}}(z_2 - z_1) + z_1 \\ \theta' &= \theta + \beta_c + (y - y_c)\gamma \end{aligned} \tag{16}$$

And we can therefore compute the differential of f, $df|_g$

$$df|_x = \begin{bmatrix} \cos(\beta_y) & -r_x \sin(\beta_y) & 0 \\ \sin(\beta_y) & r_x \cos(\beta_y) & 0 \\ 0 & \frac{z_2 - z_1}{H} & 0 \\ 0 & \gamma & 1 \end{bmatrix} \tag{17}$$
$$\beta_y = \beta_c + (y - y_c)\gamma, \;\; r_x = \gamma(r_c + x - x_c)$$

The cost can also be deformed with matrix M as

$$M = \begin{bmatrix} c_x & 0 & 0 & 0 \\ 0 & c_y & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_\theta \end{bmatrix}. \tag{18}$$

The third column is all zero because the robot has no z component active body velocity and the z velocity in deformed space is enforced by the underlying manifold.

For the experiment, we first measured the size of the corridor by getting a point cloud with Velondyne LiDAR. With this point cloud, we estimated the major and minor axis of the ellipse by projecting elliptical helix onto horizontal plane. Then we tuned the parameters of the deformation function to make our deformed state lattice match the real corridor. Then we run the planner and executed the planned path on the corridor.

Because the deformed state lattice can not perfectly match the helix corridor and the initial pose where we start the robot is not exactly aligned with the start pose on state-lattice, especially the initial orientation will affect the whole trajectory drastically. To avoid collision with the wall, we employ a simple feedback controller based on a 2D Hokuyo laser, which can be used to measure the relative distance and orientation to the wall.

The experiment validates the correctness and efficiency of the DSLP and gives an illustrative example about how DSLP might be used for mobile robot planning in real world.

VI. Conclusion and future work

In this work, we proposed a planning framework called DSLP. By finding a nonlinear map that captures the proper low dimensional form-fitting of the environment, we can map state lattices, motion primitives and heuristics between regular space and deformed space. Based on this, we can easily adapt any state-of-the-art search-based planner to a corresponding DSLP planner with same time complexity by only changing getting cost and getting heuristic parts with the corresponding DSLP ones. Within this framework, we also dicuss how to address the case when the deformed state lattice loses lots of connectivity and how to design heueristic in deformed space. Based on DSLP framework, the planning process is facilitated, the planned path is globally smoother and less memory is required to save state lattice.

In the future, possible research directions are how to automatically identify the underlying form-fitting manifold from a map of the real environment, like a point cloud. This will make the DSLP framework more applicable to various types of environment.

References

[1] Pivtoraiko, Mihail, and Alonzo Kelly. "Generating near minimal spanning control sets for constrained motion planning in discrete state spaces." 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2005.

[2] Pivtoraiko Mihail, A. Knepper Ross and Alonzo Kelly, "Differentially constrained mobile robot motion planning in state lattices", Journal of Field Robotics, vol. 26, no. 3, pp. 308-333, 2009.

[3] Rayner, D. Chris, Michael H. Bowling, and Nathan R. Sturtevant. "Euclidean Heuristic Optimization." In AAAI. 2011.

[4] Gochev, Kalin, Benjamin Cohen, Jonathan Butzke, Alla Safonova, and Maxim Likhachev. "Path planning with adaptive dimensionality." In Fourth annual symposium on combinatorial search. 2011.

[5] Chen, Wenlin, Yixin Chen, Kilian Q. Weinberger, Qiang Lu, and Xiaoping Chen. "Goal-Oriented Euclidean Heuristics with Manifold Learning." In AAAI. 2013.

[6] Likhachev, Maxim, Geoffrey J. Gordon, and Sebastian Thrun. "ARA*: Anytime A* with provable bounds on sub-optimality." In Advances in Neural Information Processing Systems, p. None. 2003.

[7] Stentz, Anthony. "Optimal and efficient path planning for partially-known environments." In Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on, pp. 3310-3317. IEEE, 1994.

[8] Hart, Peter E., Nils J. Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths." IEEE transactions on Systems Science and Cybernetics 4, no. 2 (1968): 100-107.

[9] Hopf, Heinz, and Willi Rinow. "ber den Begriff der vollstndigen differentialgeometrischen Flche." Commentarii Mathematici Helvetici 3, no. 1 (1931): 209-225.

[10] Ferguson, Dave, and Anthony Stentz. "Anytime, dynamic planning in high-dimensional search spaces." Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, 2007.

[11] Lim, K. B., and G. J. Balas. "Line-of-sight control of the CSI evolutionary model: control." American Control Conference, 1992. IEEE, 1992.