# An Exact Optimal Kinodynamic Planner Based on Homotopy Class Constraints

Basak Sakcak(✉), Luca Bascetta, and Gianni Ferretti

Dipartimento di Elettronica Informazione e Bioingegneria, Politecnico di Milano, Piazza L. da Vinci, 32-20133 Milano, Italy
{basak.sakcak,luca.bascetta,gianni.ferretti}@polimi.it

**Abstract.** This paper proposes a kinodynamic planning algorithm for vehicles with bounds on actuation. The proposed approach is based on identification of homotopy classes to decompose the global obstacle avoidance problem into several simpler subproblems. We formulate the homotopic trajectory planning problem in a multiple phase trajectory optimization scheme, such that at each phase different kinematic constraints are active. This novel formulation satisfies the collision-free constraints and homotopy constraints at the same time.

**Keywords:** Kinodynamic planning · Homotopy · Mobile robots

## 1 Introduction

Autonomous vehicles have gained considerable attention in the field of robotics as the number of applications, ranging from service robots to surveillance and to search and rescue missions, keep increasing. Motion planning is one of the most important factors that affect the functioning of an autonomous vehicle and deals with the problem of finding a collision-free trajectory that guides the vehicle from an initial configuration to a final one. For many vehicles, using a decoupled approach of first generating a collision-free path makes it difficult to adapt a feasible trajectory. Thus, it is necessary to address the kinematic and the dynamic constraints of the vehicle at once.

Kinodynamic planning, as first introduced by Donald et al. in [3], is the problem of finding a trajectory that is not only collision-free, but at the same time considers the underlying dynamics of the system. Even though it generates feasible collision free trajectories it is inherently more complex compared to planning only with kinematic constraints, since it involves higher dimensional search spaces and differential constraints.

In practical approaches, *search based* and *sampling based* methods represent the two most widespread classes of algorithms, avoiding the complexity of explicitly representing the search space by a collision check. For the former type, the state space is discretized and a graph of states is computed that is connected with feasible trajectories. The motion planning problem is then casted into finding the best sequence of motions by traversing this graph [9,14]. However, they are only resolution complete and suffer from branching for high dimensional state spaces. On the other hand, sampling based approaches iteratively sample the continuous configuration space and solve for a trajectory, which is then added to the graph if it is collision free. Due to their practicality there has been an extensive effort on sampling based planning with kinodynamic constraints. However, in order to have optimality guarantee on the solution, sampled states should be connected to the graph with an optimal trajectory which corresponds to solving a *two point boundary value problem* (TPBVP), a process which is not trivial and computationally expensive for the many nodes to be added to the graph, especially for systems with complex dynamics. Therefore, some research has been conducted to overcome this difficulty such as obtaining an analytical solution [18], approximating the solution [13] or avoiding an exact steering by using a shooting approach [6]. However, these methods are either too conservative or they do not have a proven guarantee on optimality.

Due to the complexity of a search in the state space, some approaches, especially the ones that have to generate online plans in response to dynamic environments, compute a coarse initialization and use trajectory optimization techniques to find the optimal one [15,16]. Most of these algorithms that are gradient based return solutions within the same class of trajectories due to the discontinuities imposed by the obstacles. In other words, they return a trajectory which is in the same homotopy class of the initial guess. A homotopy class in the context of trajectory planning is the set of all trajectories that can be morphed to each other without any discontinuity (Fig. 1). In other words, trajectories that belong to the same class avoid obstacles in the same way.

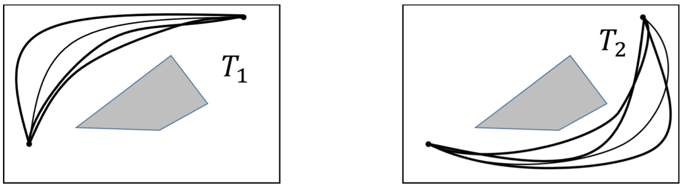


**Fig. 1.** $T_1$ and $T_2$ define two sets of trajectories that belong to two distinct homotopy classes. All the trajectories in $T_1$ belong to the same class, the same for $T_2$.

Approaches that take into account the topology of the search space can generate several locally optimal trajectories and select the best one among them or if given the information about the class that contains the optimal solution speed up the computation by focusing on this class. In order to compute distinct homotopy classes, Bhattacharya et al. [1] proposed calculating a class invariant signature, the homotopy classes are then identified by applying A* algorithm [12] on an arbitrary roadmap representing the environment. In [5] the authors capture the homotopy information by selecting a reference point and partitioning the search space by generating line segments that connect this point to the obstacles. They augment a number of well-known path planning algorithms with homotopy class constraints without considering the dynamics. Focusing only on vehicles that operate on 2D planes Kuderer et al. [7] and Park et al. [10] proposed to partition the search space using Voronoi cells and cell decomposition, respectively. In [7] the authors compute homotopy classes based on the vector of winding angles around obstacles, which is invariant for homotopic trajectories. They then compute local optimal trajectories optimizing the splines that interpolate the centers of the Voronoi cells. Park et al. [10] obtained the homotopy classes by means of cell sequences and used *Mixed Integer Quadratic Programming* (MIQP) to compute trajectories that follow the cell sequence and avoid the obstacles. Similar to [10] we also use the cell sequence information to get the topology information.

Other than allowing computation of distinct trajectories that are locally optimal, homotopy class constraints also restrict the trajectory to avoid the obstacles in a specific manner. This can interest a various number of robotics applications, ranging from favoring some areas for surveillance, restricting dangerous zones, shared autonomy where the user makes only high level decisions [7] or for tactical reasoning [4].

In this work we propose an optimal trajectory generation framework for vehicles with kinodynamic constraints that operate on 2D surfaces. The algorithm is based on partitioning the search space into convex cells and identifying distinct promising homotopy classes by means of cell-sequences. The main contributions of this work are twofolds. First, we propose a heuristic to select a small number of promising cell-sequences in order to have tractable number of problems. Consequently, we formalize the optimal homotopic trajectory planning as a nonlinear optimal control problem having different phases, such that shared boundaries between two consecutive cells trigger phase transitions and a set of linear inequalities that correspond to obstacle boundaries that are active in the corresponding phase provide collision avoidance. This problem is then efficiently solved using nonlinear programming.

## 2 Problem Definition

Let the compact sets $S \subset \mathbb{R}^d$ and $U \subset \mathbb{R}^m$, with $d \geq 2$ and $m \geq 1$, represent the state and the control spaces, respectively, for the following dynamical system.

$$\dot{\mathbf{s}}(t) = f\left(\mathbf{s}(t), \mathbf{u}(t)\right) \tag{1}$$

where $f$ is a continuously differentiable function, $\mathbf{u}(t) \in U$ is the vector of control inputs and $\mathbf{s}(t) \in S$ is the state vector.

Assuming the system has bounds on the actuation, the free control space $U_{free} = \{\mathbf{u}(t) \in \Omega \subset U\}$ is introduced, where $\Omega$ is the set of admissible controls.

In this work we assume that the vehicle is operating on a planar surface and obstacles are defined as 2D polygons. Let $\mathcal{P} \subset \mathbb{R}^2$ define the 2D plane that the vehicle operates and $\mathcal{P}_{obs} \subset \mathbb{R}^2$ is the obstacle region. Consequently, the obstacle-free region is defined as $\mathcal{P}_{free} := \mathcal{P} \setminus \mathcal{P}_{obs}$. Let $\kappa : S \to \mathbb{R}^2$ be a transformation that maps a state to the 2D plane, i.e., to the position component such that $\kappa(\mathbf{s}(t)) = \boldsymbol{\pi}(t)$, where $\boldsymbol{\pi}(t)$ denotes the position of the vehicle. Finally, the free state space, $S_{free}$, is defined as the set of states that are collision free, i.e., $S_{free} := \{\mathbf{s} \in S \mid \kappa(\mathbf{s}) \in \mathcal{P}_{free}\}$.

A trajectory of system (1) is defined by the tuple $\sigma = (\mathbf{s}(t), \mathbf{u}(t), T)$, where $T$ is the duration of the trajectory (i.e., the final time), and the states and control inputs are defined along this trajectory as $\mathbf{s} : [0, T] \rightarrow S$ and $\mathbf{u} : [0, T] \rightarrow U$, respectively.

The optimal kinodynamic motion planning problem with homotopy class constraints is then formalized as finding a set of states and controls of a dynamic system over a duration $[0, T^*]$, i.e., $\mathbf{s}^*(t) : [0, T^*] \rightarrow S$ and $\mathbf{u}^*(t) : [0, T^*] \rightarrow U$, such that the trajectory

– starts from an initial state $\mathbf{s}(0) = \mathbf{s}_0$;
– avoids collisions with obstacles, i.e., $\mathbf{s}(t) \in S_{free}\ \forall t \leq T^*$;
– fulfills the actuation bounds, i.e., $\mathbf{u}(t) \in U_{free}\ \forall t \leq T^*$;
– reaches the goal region, i.e., $\mathbf{s}(T^*) \in S_{goal}$, where $S_{goal}$ is the goal region;
– minimizes an objective function defined as follows

$$J(\mathbf{s}, \mathbf{u}, T) = \int_0^T g\left(\mathbf{s}(t), \mathbf{u}(t)\right) \mathrm{d}t \tag{2}$$

where $g(.)$ is a cost function;
– belongs to the desired homotopy class, i.e., $\sigma^* \in \mathit{\Sigma}$ where $\mathit{\Sigma}$ denote the set of all trajectories that belong to the desired homotopy class.

Note that, the optimal trajectory duration, $T^*$, is one of the unknowns of the problem.

Once the motion planning with homotopy class constraints is formalized, the preliminary step of the problem is to identify distinct homotopy classes. For that purpose, this step is defined as to identify a set of homotopy classes and represent them with a set of geometric constraints that can be used to impose the homotopy class constraints. Once a set of homotopy classes are identified, they are formulated as sub-problems as in the aforementioned problem definition. Denote $\{\mathit{\Sigma}_i\}$ as the set of homotopy classes that are identified, the overall problem is then defined as finding a set of local optimal trajectories, $\{\sigma_i^*\}$, such that each of them belongs to distinct homotopy classes, i.e. $\forall i$, $\sigma_i^* \in \mathit{\Sigma}_i$.

## 3 Proposed Method

Proposed method relies on the projection of the state space onto the 2D plane. This space is partitioned into convex cells based on the polygonal obstacles and the topology information is obtained. Following that, a set of promising homotopy classes are selected as sub-problems to the global kinodynamic planning problem and each sub-problem is formalized as an optimal control problem.

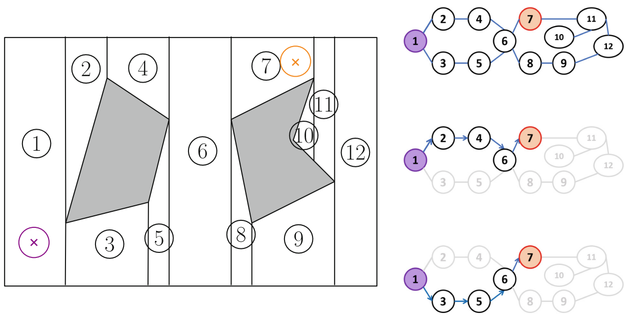


**Fig. 2.** Left: Trapezoidal decomposition and resulting convex cells, purple and orange marks are the initial and the goal positions of the robot. Right: (Top) The adjacency graph that carries the topological information about the environment, root and the end nodes are marked purple and orange respectively corresponding to the cells that contain the initial and the goal positions of the robot. (Bottom) Two example homotopy classes as cell sequences connecting the initial configuration to the goal. (Color figure online)

## 3.1 Identification of Homotopy Classes

The identification of distinct homotopy classes is based on getting the topological information using *convex cell-decomposition* [8] and obtaining a set of cell-sequences (Fig. 2).

The idea of convex decomposition is to partition the $\mathcal{P}_{free}$ into a set of convex regions $\{Q_i\}$ called *cells*. In this work we use a well-known cell decomposition technique called *vertical cell decomposition* [2], which achieves this using the set of vertices $V$ belonging to $\mathcal{P}_{obs}$ and extending rays from each of the vertices $v \in V$ through $\mathcal{P}_{free}$ until they hit $\mathcal{P}_{obs}$.

An undirected graph that captures the adjacency information is then computed using the cells $Q_G = \{Q_i\}$, taking the cells as nodes in the graph, they are connected with an *edge* if two cells $Q_i$ and $Q_j$ share a common boundary $\mathcal{B}_{i,j}$. As a result, a graph carrying this adjacency information is designed as $\mathcal{G} = (Q_G, E_G)$, with $E_G = \{E(Q_i, Q_j) \mid \forall_{i \neq j} \mathcal{B}_{i,j} \neq \emptyset\}$. Assuming that the initial and final positions, $\boldsymbol{\pi}_0$ and $\boldsymbol{\pi}_f$ respectively, lie strictly inside the cells but not on the boundaries, a cell for each one can be assigned as the initial and final nodes in the graph, $Q_0$ and $Q_f$, respectively. Once the initial and final nodes are assigned, the homotopy classes are computed using a graph search on the adjacency graph. Having the topological information in the form of a graph it is necessary to select a set of sequences that correspond to distinct classes, since there are infinite number of sequences.

We use a modified version of *Depth First Search* (DFS) [17] algorithm for graph traversal in order to retrieve the homotopy classes by means of cell sequences that connect a source node, $Q_0$ to an end node $Q_f$. Therefore, instead of stopping once a solution is found that connects initial node to the final one, the algorithm keeps returning possible sequences. In order to restrict the number

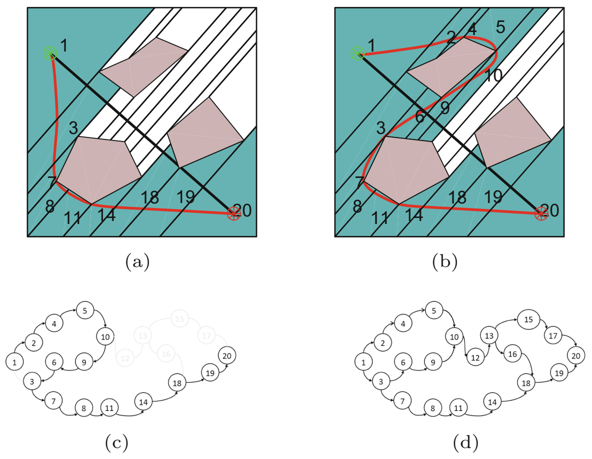


**Fig. 3.** Direction of motion is from green mark to red (a) Optimal trajectory (red line) computed on loopless cell sequence is monotonic along the direction of motion (b) Although the cell sequence do not contain any repetitions the optimal trajectory is not desirable (c) Cell sequence resulting in trajectory on the right when an undirected graph is used (d) Obtaining a directed graph along the direction of motion would avoid selection classes like (b) (Color figure online)

of problems to be solved, we use a no-loop strategy since loops generally indicate trajectories with higher costs and they are not desirable in robotic motion.

Selecting loopless cell-sequences avoid unnecessary turns around the obstacles, however it does not guarantee desirable trajectories. We define the straight line that connects the initial position to the final one as the *direction of motion*. When most of the common cost functions to minimize are concerned, e.g. minimum time, minimum energy expenditure etc., it is reasonable to assume that the trajectories with lower cost are monotonic with respect to this direction of motion. In other words, the distance from the goal decreases monotonically. Therefore, based on this assumption we propose a method to select loopless-cell sequences that would result in loopless and lower cost trajectories. For that reason, the space is decomposed using rays perpendicular to the straight line connecting the initial position to the final one, i.e. direction of motion. Selecting cell-sequences that are monotonic along this line would also result in trajectories that have less change in the course, hence more desirable. Therefore, we propose to obtain a directed graph such that the cells would be passed with a monotonic fashion along the direction of motion by the computed trajectories (Fig. 3). Note that, if the cell containing the final position is not accessible with the directed graph approach, e.g. behind the wall or in a closed environment, the graph builds edges to the adjacent cells until it is connected to the graph.

In order to keep the multiphase trajectory planning problem simple, instead of creating oriented boundaries we rotate the $P_{free}$ such that the straight line connecting the initial and final positions will be horizontal, i.e. correspond to the x-axis. This would provide a phase transition at a specified $x$-intercept.

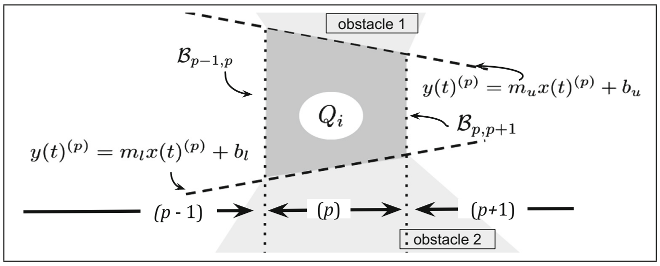


**Fig. 4.** Since each cell $Q_i$ is a convex polygon the collision avoidance can be obtained as a union of half-planes perpendicular to the obstacle edge that is active during the corresponding phase

## 3.2 Trajectory Optimization

After a set of cell sequences are selected, where each sequence corresponds to a different homotopy class, each class is formulated as a multiple phase trajectory optimization problem, such that each convex cell corresponds to a different phase in the problem. Therefore, a homotopy class of $N$ consecutive cells can be formulated as an optimal control problem of $N$ phases.

The independent variable, the time $t$, is defined in the region $t_0^{(p)} \leq t \leq t_f^{(p)}$ for phase $p$, and the phases are sequential with respect to $t$ such that $t_f^{(p)} = t_0^{(p+1)}$. Each phase is subject to differential constraints (1) describing the dynamics of the system

$$\dot{\mathbf{s}}^{(p)} = f\left(\mathbf{s}(t)^{(p)}, \mathbf{u}(t)^{(p)}\right) \tag{3}$$

There are bounds on the state variables

$$\mathbf{s}_{min}^{(p)} \leq \mathbf{s}(t)^{(p)} \leq \mathbf{s}_{max}^{(p)} \tag{4}$$

and on control variables

$$\mathbf{u}_{min}^{(p)} \leq \mathbf{u}(t)^{(p)} \leq \mathbf{u}_{max}^{(p)} \tag{5}$$

The initial state of the first phase is completely fixed and the final state of the final phase can be either partially or fully specified depending on the definition of $S_{goal}$. Therefore,

$$\mathbf{s}(t_0)^{(1)} = \mathbf{s_0},\ \mathbf{s}(t_f)^{(N)} = \mathbf{s_f} \tag{6}$$

where $\mathbf{s}(t_0)^{(1)}$ represents the state variables at the beginning of the first phase and $\mathbf{s}(t_f)^{(N)}$ are at the end of the final phase ($p = N$). Furthermore, in order to obtain a continuous trajectory through phases, the following constraint is enforced at the phase boundaries

$$\mathbf{s}(t_f)^{(p)} = \mathbf{s}(t_0)^{(p+1)} \tag{7}$$

Partial terminal conditions for each phase, apart from the final one, are defined as

$$\boldsymbol{\pi}(t_f)^{(p)} \in \mathcal{B}_{p,p+1} \tag{8}$$

ensuring that the state variables corresponding to the position, $\mathbf{s} = [\boldsymbol{\pi}, \dots]$ and $\boldsymbol{\pi}(t_f)^{(p)} = [x, y]^T$, at the end of each phase lie strictly on the line segment $\mathcal{B}_{p,p+1}$ representing the shared boundary between two consecutive phases. Constraints (7) and (8) together ensure that cell sequence defining the homotopy class is respected.

The trapezoidal decomposition provides trapezoidal cells having two sets of boundaries, one set includes the boundaries $\{B_{(p,p+1)}, B_{(p,p-1)}\}$ for phase transition and the other set the boundaries shared with the obstacles. These boundaries can be expressed as linear inequalities (Fig. 4) for collision avoidance and combined with (4) confining the trajectory inside the cell, as follows

$$y(t)^{(p)} \leq m_u^{(p)} x(t)^{(p)} + b_u \tag{9}$$

$$-y(t)^{(p)} \leq -m_l^{(p)} x(t)^{(p)} - b_l \tag{10}$$

where $m_u$, $m_l$ and $b_u$, $b_l$ represent the slope-intercept parameters of the lines created by the upper and lower boundaries between the cell and the obstacles correspondingly.

## 4 Numerical Example

We developed and tested the algorithm by a MATLAB implementation using the equations of motion of a unicycle like robot

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = w \\ \dot{v} = a \end{cases} \tag{11}$$

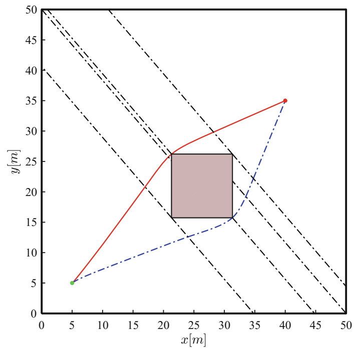
moving on a planar surface, where $(x, y)$ is the position of the vehicle, $(\theta)$ is the orientation, $(v)$ is the linear velocity and the control input is given by $\mathbf{u} = [w, a]^T$ with the angular velocity $w$ and the linear acceleration $a$. Test environment is bounded by $x \in [0, 50]$, $y \in [0, 50]$ and the states other than the position of the vehicle are bounded as $\theta \in [-\pi, \pi]$ and $v \in [0, 3]$.

For this numerical example we used a minimum-time, minimum-energy strategy as a cost metric for optimization, and the optimal trajectory is obtained by minimizing the following cost function over all the phases

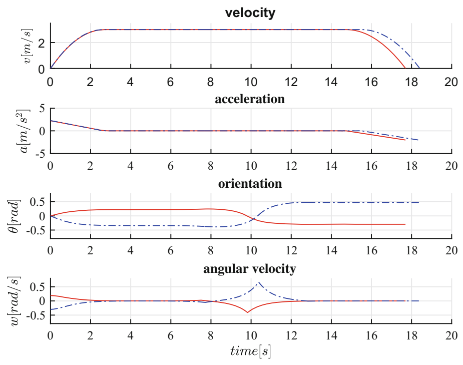
$$J(u) = \int_{T_0}^{T_f} \left[1 + u(t)^T R u(t)\right] dt \tag{12}$$

where $T_0$ and $T_f$ are the initial and the final times for the overall problem. The control variables $a$ and $w$ are equally penalized using the weight $R = 5I_2$ and bounded as $a \in [-5, 5]$, $w \in [-3, 3]$.

A continuous time optimal control problem with multiple phases is transformed into a *nonlinear programming problem* (NLP), which is then solved using the MATLAB toolbox GPOPS [11] which uses a *Legendre-Gauss-Radau orthagonal collocation method.*



(a)



(b)

**Fig. 5.** (a) Distinct local optimal trajectories from the initial position (green) to a final one (red), best cost trajectory is shown in red. (b) Corresponding velocity, orientation and actuation profiles. (Color figure online)

Figure 5(a) shows a simple example for a situation where two different avoidance strategies exist for a single obstacle. Homotopy classes are identified by a decomposition of the space along the direction of motion, i.e. straight line connecting initial and final positions, and two loopless sequences are selected. Figure 5(b) reports the actuation and velocity profiles for both cases, which clearly shows that the limits are never exceeded neither for control variables nor for velocity.

Trajectories generated using loopless cell sequences for different number of obstacles are shown in Fig. 6. Clearly, all of them belong to distinct classes, therefore they represent the optimal solution to distinct sub-problems and they are monotonic along the direction of motion. Trajectories marked with red color correspond to the lowest cost ones for the selected objective function. However, depending on the scope of the problem different navigation decisions can be made, that may or may not correspond to the best cost trajectory among all.

A total number of 40 different environments are created that contain different number of randomly generated rectangular obstacles having various sizes. Simulations are carried out on an IntelCore i7 @2.40 GHz personal computer with 8 GB RAM. The shortest path in the polygonal environment corresponding

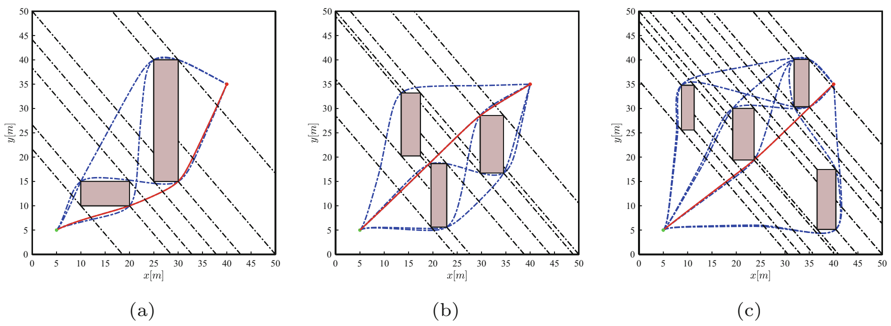


**Fig. 6.** Trajectories that belong to distinct homotopy classes for different number of obstacles.

to the proper homotopy class is used as an initial guess. Figure 7 shows the computation times to solve the trajectory optimization problem for different number of obstacles including the decomposition. The amount of time needed to solve a single problem (generating a trajectory in the designated homotopy class) is also reported, since the trajectory optimization problem for each homotopy class is independent, hence parallelizable. Finally, Table 1 shows the average number of phases for each sub-problem for different number of obstacles. As expected, higher number of obstacles in the environment means higher number of vertices and increases the partitioning of the space, hence increasing the number of phases for each sub-problem. This also increases the amount of time needed to solve each subproblem.

If the problem does not admit a monotonic trajectory from an initial state to the goal, the graph representation of the topology may not include a path to the goal cell if a directed graph is used. For example, when the vehicle has to get out of a U-shaped environment (Fig. 8). In that case, the directed graph that allows only monotonic motion along the direction of motion contains only the cell $Q_3$, hence it does not contain a cell sequence to the goal (Fig. 8(a)). In that case, we augment the directed graph by progressively adding edges to the goal cell from

**Table 1.** For different number of obstacles: Number of phases (length of the corresponding loopless cell sequence on the undirected graph) that correspond to different homotopy classes and number of homotopy classes.

| Number of obstacles | Number of phases | | Number of homotopy classes |
|---|---|---|---|
| 1 | 5 | | 2 |
| 2 | 8 | | 4 |
| | min | max | |
| 3 | 8 | 10 | 5 |
| 4 | 12 | 13 | 8 |

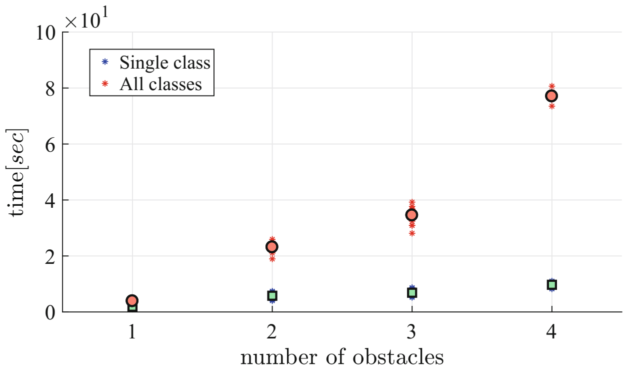


**Fig. 7.** For different number of obstacles: Computation times of generating a trajectory for a sub-problem and the overall problem for different simulation runs and the corresponding average computation times.

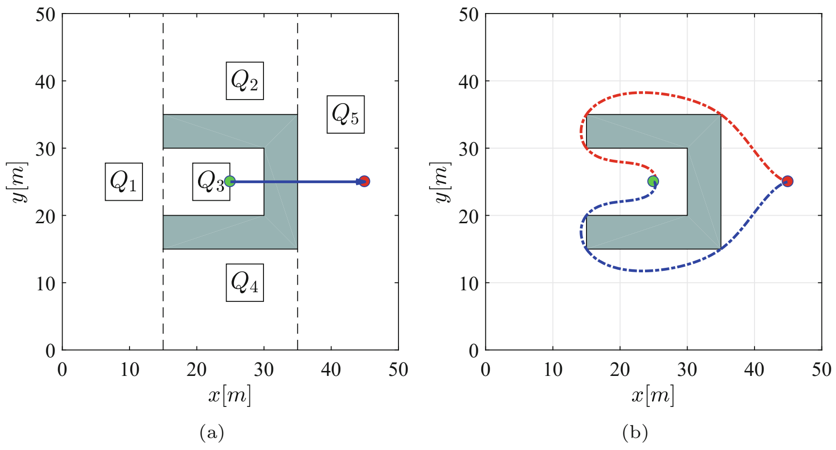


**Fig. 8.** (a) Decomposition of the workspace and the direction of motion from the start to the goal. (b) Solution trajectories that belong to different homotopy classes.

the adjacent ones until the goal cell is connected. Note that, as long as there exists a trajectory connecting the initial and final states, the corresponding cell sequence will be represented in the decomposition.

## 5 Conclusion

In this paper we presented an approach for obtaining optimal trajectories for vehicles with nonlinear dynamics based on homotopy class identification. The method obtains the topology information using trapezoidal decomposition as a graph of convex cells. Distinct homotopy classes are then obtained by traversing this graph. We proposed a heuristic to restrict the number of classes to a limited set of good candidates to reduce the number of problems to be solved and to prevent obtaining an intractable number of cell sequences. We then defined the trajectory generation problem in the homotopy class in a novel scheme using multiple phase optimization that satisfies the collision-free and homotopy class constraints at the same time for nonlinear system dynamics with actuation bounds. The approach was tested in a virtual environment populated with obstacles, more complex environments can also be considered but this is outside the scope of this work and it is still an ongoing research.

## References

1. Bhattacharya, S., Likhachev, M., Kumar, V.: Topological constraints in search-based robot path planning. Auton. Robots **33**(3), 273–290 (2012)
2. Chazelle, B.: Approximation and decomposition of shapes. Algorithmic Geom. Aspects Robot. **1**, 145–185 (1985)
3. Donald, B., Xavier, P., Canny, J., Reif, J.: Kinodynamic motion planning. J. ACM (JACM) **40**(5), 1048–1066 (1993)
4. Gu, T., Dolan, J.M., Lee, J.W.: Automated tactical maneuver discovery, reasoning and trajectory planning for autonomous driving. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5474–5480. IEEE (2016)
5. Hernandez, E., Carreras, M., Ridao, P.: A comparison of homotopic path planning algorithms for robotic applications. Robot. Auton. Syst. **64**, 44–58 (2015)
6. Jeon, J.H., Karaman, S., Frazzoli, E.: Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*. In: Decision and Control and European Control Conference (CDC-ECC) (2011)
7. Kuderer, M., Sprunk, C., Kretzschmar, H., Burgard, W.: Online generation of homotopically distinct navigation paths. In: 2014 IEEE International Conference on Robotics and Automation (ICRA). IEEE (2014)
8. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
9. Likhachev, M., Ferguson, D.: Planning long dynamically feasible maneuvers for autonomous vehicles. Int. J. Robot. Res. **28**(8), 933–945 (2009)
10. Park, J., Karumanchi, S., Iagnemma, K.: Homotopy-based divide-and-conquer strategy for optimal trajectory planning via mixed-integer programming. IEEE Trans. Robot. **31**(5), 1101–1115 (2015)
11. Patterson, M.A., Rao, A.V.: GPOPS-II: a MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming. ACM Trans. Math. Softw. (TOMS) **41**, Article No. 1 (2014)
12. Pearl, J.: Heuristics: intelligent search strategies for computer problem solving (1984)
13. Perez, A., Platt, Jr. R., Konidaris, G., Kaelbling, L., Lozano-Perez, T.: LQR-RRT*: optimal sampling-based motion planning with automatically derived extension heuristics. In: Robotics and Automation (ICRA) (2012)
14. Pivtoraiko, M., Knepper, R.A., Kelly, A.: Differentially constrained mobile robot motion planning in state lattices. J. Field Robot. **26**(3), 308–333 (2009)

15. Ratliff, N., Zucker, M., Bagnell, J.A., Srinivasa, S.: CHOMP: gradient optimization techniques for efficient motion planning. In: IEEE International Conference on Robotics and Automation, ICRA 2009, pp. 489–494. IEEE (2009)
16. Richter, C., Bry, A., Roy, N.: Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In: Inaba, M., Corke, P. (eds.) Robotics Research. STAR, vol. 114, pp. 649–666. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-28872-7_37
17. Tarjan, R.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972)
18. Webb, D.J., van den Berg, J.: Kinodynamic RRT*: optimal motion planning for systems with linear differential constraints. arXiv preprint arXiv:1205.5088 (2012)