Visibility based planners for kinematically constrained vehicles

Mihai Pomarlan
Universitatea Politehnica Timisoara, facultatea ETC

Abstract—We implement a visibility-based motion planner for systems in which, because of kinematic constraints, not all maneuvers may be reversible. We use samples that are non-zero dimensional subspaces of the configuration space in an attempt to make the heuristic more efficient at generating large strongly connected components; the non-zero dimensional samples also reduce the computation load associated with validating trajectories. We integrate the resulting planner method with various local planner strategies for kinematically constrained vehicles, and compare the sizes of the roadmaps needed to solve a problem with bug traps for several vehicle models, one of which is reversible and serves as a base-line. Finally, we suggest directions of improvement of the algorithm based on simulation results.

Index Terms—robot motion planning, sample-based planners, visibility roadmaps, kinematic reducibility, maneuver automata, differential flatness

I. INTRODUCTION

Most of the state of the art motion planners for robots are sampling based. Such a planner generates random point configurations in the configuration space of a robot, such that none of the chosen configurations intersects with forbidden regions (“obstacles”), and then attempts to connect samples that are near to each other into a graph called a roadmap using some local planner method which is tailored to the specifics of the robotic system. The roadmap is intended to be useful for many planning queries. The PRM (Probabilistic Roadmap Method [9]) and RRG (Rapid Random Graphs [7]) are examples of such algorithms. Alternatively, the planner may start from a given configuration and select one random new sample at a time, which it attempts to connect to previously stored samples in the roadmap, like the RRT (Rapid Random Trees) algorithm and its many variants [10], [3], [8], [16]; such algorithms are referred to as single-query, because the structure they produce is only intended to be useful for planning from a given source configuration, to a particular and also given destination.

Sampling based planners are preferred because they are simple to implement, have fast sample-and-connect iteration steps, and they have a chance to quickly find a solution even without constructing a detailed map of all possible configurations. They are probabilistically complete, which means that the more sample-and-connect steps, the better the chance to find a solution if one exists.

In general however, a roadmap contains many samples before it can reliably answer planning queries. There is interest however in reducing the number of samples stored in the roadmap, as this helps reduce query times and may be useful for planners which update the roadmap to account for environment changes [5], [12]. Some heuristics to reduce sample counts in roadmaps have been proposed however. One such heuristic is visibility [13], [14]: a sample is kept only if it cannot be connected to samples previously present in the roadmap (improves the coverage) or if it allows a path between different connected components of the roadmap to exist (it improves the connectivity). Visibility based algorithms initially tested against all previously present nodes in the roadmap, but we propose here an adaptive radius to reduce connection attempts.

In previous work with visibility roadmaps, the systems for which plans were constructed were reversible. Whatever maneuver a system may perform to go from a state $A$ to a state $B$, then it can always perform it in reverse. Therefore, one can use roadmaps that are undirected graphs and formulate visibility heuristics in terms of connected components which can be described by a disjoint set structure [15]. Visibility based planners, if ran for enough sample-and-connect iterations, tend to generate one connected component for each connected component of the configuration space that the robot may occupy.

Real robots however are not necessarily reversible. Dynamics often introduces irreversibility (a car speeding towards a wall may be unable to brake in time because it cannot lose its momentum fast enough), but kinematic constraints can cause irreversibility as well (a boat, if unable to reverse, will not get out of a narrow niche once inside it). Non-reversibility may also be the result of the local planner unable to generate a valid maneuver, one that doesn’t collide with obstacles, to return to the initial configuration.

Because a return path is no longer guaranteed, the roadmap becomes a directed graph and therefore the visibility heuristic must be formulated in terms of strongly connected components for which incremental maintenance algorithms [6] should be used. It is no longer the case that the roadmap tends to contain one strongly connected component for each connected component of the configuration space. In fact, the strongly connected components may be small compared to the size of the roadmap, which negatively affects a visibility heuristic’s ability to filter samples.

In this paper, we study ways in which a variable radius visibility based planner can be applied to non-reversible
vehicles, and integrated with various local planning strategies that can account for real vehicles’ kinematic constraints. We also propose a departure from the typical sampling based planner, which uses point configurations as vertices of the roadmap. Instead, we investigate here the use non-zero dimensional sample subspaces of the configuration space. Such sample subspaces are required to be simple maneuvers that the local planner can produce. They must not intersect obstacles, and they must be strongly connected subsets of the configuration space. Therefore, from any point in a sample subspace one can get to any other point in the same sample subspace.

Using non-zero dimensional subspaces as samples may improve the connection opportunities between them, as well as simplifying the task of checking that the local planner trajectories are safe. For non-holonomic systems like vehicles, one needs to perform a sequence of maneuvers to steer from some point configuration to another even without obstacles being present; parallel parking is a typical example. Then it’s advantageous to store some segments of those maneuvers which have already been checked as obstacle-free, and reuse them for local planning.

Section II describes the visibility based planner and modifications to it. Section III describes the vehicle models and the local planner strategies used for them. For this paper, we study vehicles described by kinematic models (that is, we assume movement is slow enough that dynamics may be ignored). Section IV shows some experimental results with the new algorithms. Finally, conclusions are discussed in section V and future directions for research presented.

II. PLANNER ALGORITHM

A. Notations

Let \( C \) be the configuration space.

The configuration space will be referred to as \( C \), and consists of all states that a system could be in, absent other constraints. \( C_{\text{obs}} \) is the set containing all states which are forbidden because they "intersect" or "collide" with "obstacles". \( C_{\text{obs}} \) is a subset of \( C \), and \( C_{\text{free}} = C - C_{\text{obs}} \) is the subset of allowable, or valid, states. It is assumed that the configuration space and its subsets are vector spaces with metric and measure functions.

A (directed) graph is defined as a pair of sets \( \{V,E\} \) with \( V \) the set of vertices and \( E \) a set of ordered pairs of distinct elements from \( V \). Such a pair is called an edge and the two vertices are its ends. Edges are directed. Intuitively, if an edge \((A, B)\) exists in the graph, then one can go from vertex \( A \) to vertex \( B \). A path is a sequence of vertices with the property that subsequent vertices are linked by appropriately directed edges.

A (strongly) connected component of a directed graph is a subset of its vertices and all the edges between them such that: for any two vertices in the connected component there exist paths in both directions inside the component; and, for any vertex \( A \) not in the strongly connected component, and any vertex \( B \) in the strongly connected component, either it is impossible to go from \( A \) to \( B \), or it is impossible to go from \( B \) to \( A \).

The planner will construct a roadmap of \( C_{\text{free}} \), which is a graph where vertices, referred to as samples, are subspaces in \( C_{\text{free}} \) and edges between two vertices exist if a local planner procedure is capable to steer from one subspace to the other. Note that while typically the samples used by a planner are simply points in \( C_{\text{free}} \), in this paper the samples are subspaces of non-zero dimension.

If the local planner can generate a trajectory from a sample \( A \) to a sample \( B \), then \( B \) is said to be visible from \( A \); this does not necessarily imply that \( A \) is visible from \( B \) however.

B. The visibility based algorithm

The planner algorithm is a variable radius visibility based planner. The basic outline of it is that of a sampling planner: random new samples are generated and connections are attempted with samples previously existing in the roadmap (see alg. 1). Only samples that improve the coverage or the connectivity of the roadmap are retained. The algorithm uses an adaptive radius, \( r_{\text{RRT}}(\mid V \rceil) \), to search for vertices to attempt connections with, and a constant fallback radius \( r_f \) in case all attempts at the adaptive radius fail. The adaptive radius decreases such that the expected number of samples in a visibility domain increases with the logarithm of the number of samples in the roadmap, rather than increase linearly.

A difference is in the ImprovesMap heuristic, because in this paper the roadmap is a directed graph, unlike previous work with visibility roadmaps. It is expected that the roadmap will contain several strongly connected components (indeed, a lone vertex may be a strongly connected component). As such, the notion of "improves connectivity" now means that a new sample is linked by an inbound edge to a strongly connected component, and by an outbound edge to another, so that it allows a path to go from the first component to the second. It will not necessarily be the case that the sample will allow travel in the other direction; but if it did, then it would result in the two strongly connected components merging.

Then the heuristic of deciding when a new sample is "useless" and should be discarded is this: either all the edges of that sample are in the same direction (it is a sink with all edges inbound, or a source with all edges outbound), or it is visible only from several vertices in the roadmap that are all located in the same strongly connected component (see alg. 2).

Another difference from previous versions of visibility based planners is that samples are no longer points in \( C_{\text{free}} \), but they are allowed to be non-zero dimensional subspaces of \( C_{\text{free}} \). Distance functions must be changed accordingly, and the sampling procedure must be able to generate such subspaces and make sure they stay inside \( C_{\text{free}} \). The details of the sampling procedure are given in section III for each of the studied vehicles. The reason for this change is to ease checking maneuver sequences generated by the local planner.
by storing some maneuvers which are known to be obstacle free.

The necessary condition imposed on a sample subspace is that it be a strongly connected subspace of $C_{free}$, meaning, for any two configurations in the subspace, the local planner can generate paths between them both ways, such that the paths do not exit $C_{free}$. If true, the condition implies that if one point in the subspace can be reached from a given configuration, then all are reachable.

III. VEHICLE MODELS

The simulated vehicles for this paper all move in a two dimensional work space and will be maneuvered in such a way that kinematic models (position variables for state and continuous velocity controls) are enough to describe and plan their motion. The following subsections describe, for each vehicle, its model and steering procedure.

Algorithm 1 Variable radius visibility algorithm

1: $V \leftarrow \{x_{rand}\}; E \leftarrow \emptyset; k \leftarrow 0$
2: while $k < N$ do
3: $X_{vis} \leftarrow \emptyset$
4: $X_{rev} \leftarrow \emptyset$
5: $x_{rand} \leftarrow $Sample();
6: $X_{near} \leftarrow $Near$(x_{rand}, V, r_{RRT}, (|V|));$
7: for all $x \in X_{near}$ do
8: if $GenerateTrajectory(x, x_{rand})$ succeeds then
9: $X_{vis} \leftarrow X_{vis} \cup x$
10: end if
11: if $GenerateTrajectory(x_{rand}, x)$ succeeds then
12: $X_{rev} \leftarrow X_{rev} \cup x$
13: end if
14: end for
15: if $\emptyset \equiv X_{vis}$ AND $\emptyset \equiv X_{rev}$ then
16: $X_{near} \leftarrow $Near$(x_{rand}, V, r_{f});$
17: for all $x \in X_{near}$ do
18: if $GenerateTrajectory(x, x_{rand})$ succeeds then
19: $X_{vis} \leftarrow X_{vis} \cup x$
20: end if
21: if $GenerateTrajectory(x_{rand}, x)$ succeeds then
22: $X_{rev} \leftarrow X_{rev} \cup x$
23: end if
24: end for
25: if $ImprovesMap(X_{vis}, X_{rev})$ then
26: $V \leftarrow V \cup \{x_{rand}\};$
27: for all $x \in X_{vis}$ do
28: $E \leftarrow E \cup \{(x, x_{rand})\};$
29: end for
30: end if
31: for all $x \in X_{rev}$ do
32: $E \leftarrow E \cup \{(x_{rand}, x)\};$
33: end for
34: end if
35: end while

Algorithm 2 map improvement condition for the visibility based planner on digraph roadmaps

1: function $ImprovesMap(X_{vis}, X_{rev})$
2: if $0 \equiv |X_{vis}|$ AND $0 \equiv |X_{rev}|$ then
3: return true
4: else if $0 \equiv |X_{vis}|$ OR $0 \equiv |X_{rev}|$ then
5: return false
6: else if $1 < |X_{vis} \cup X_{rev}|$ AND $X_{vis} \cup X_{rev}$ in one component then
7: return true
8: else
9: return true
10: end if
11: end function

Fig. 1. Vehicles and local planner maneuvers: (a) the planar body with variable direction thruster; (b) the planar car; (c) the robot and trailer; (d) steering the planar body using kinematic reduction (“decoupling field”) maneuvers; (e) steering the car; (f) steering the robot and trailer by imposing a trajectory on the trailer

A. Planar body with variable direction thruster

A two dimensional rigid body with a thruster located at some offset away from the center of mass (see fig. 1 (a)). Its model is

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \end{bmatrix} \cdot \mathbf{u}_1 + \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix} \cdot \frac{m \cdot h}{J} \cdot \mathbf{u}_2$$

(1)

where $x$ and $y$ give the position of the object’s center of mass in the plane, while $\theta$ is the heading angle. $J$ is the moment of inertia and $m$ is the mass of the object, while $h$ is the distance between the center of mass and the thruster. The thruster is assumed to not be able to push the object backwards.

Steering is done by noticing that the object model is kinematically reducible [2] and that two decoupling vector fields exist:

$$X_1 = \cos(\theta) \cdot \frac{\partial}{\partial x} + \sin(\theta) \cdot \frac{\partial}{\partial y}$$

(2)

$$X_2 = -\frac{m \cdot h}{J} \cdot \frac{\partial}{\partial \theta} - \sin(\theta) \cdot \frac{\partial}{\partial x} + \cos(\theta) \cdot \frac{\partial}{\partial y}$$

(3)
Each field corresponds to a basic maneuver. \( X_1 \) is simply the object moving in the direction given by the heading angle. \( X_2 \) is the object rotating around a point located at a fixed distance in front of the thruster, in the direction given by the heading. The trajectory traced by an \( X_2 \) maneuver is referred to as an \( X_2 \) circle. The \( X_2 \) maneuver is reversible (the object can spin in either direction in a given circle) but the \( X_1 \) maneuver is not (the object cannot push itself backwards).

The procedure to steer between two given configurations starts by identifying the \( X_2 \) circles associated to the two configurations, and computing the line between their centers. Then, an \( X_2 \) maneuver on the source configuration is done to bring the object on that line such that it faces the destination’s \( X_2 \) circle. An \( X_1 \) maneuver is then done to bring the object to the destination’s \( X_2 \) circle. Finally, another \( X_2 \) maneuver brings the object to the destination (see fig. 1 (d)). Each maneuver must start and end with the planar object at 0 velocity, so that they can be concatenated.

The motion planner uses arcs from \( X_2 \) circles as samples. An arc is a valid sample if it does not intersect obstacles. To generate an arc sample, a random configuration is selected in the workspace. If it is valid, then the planner identifies its \( X_2 \) circle and uses the largest non-colliding arc of that circle that includes the randomly selected configuration.

![Fig. 2. Going from one \( X_2 \) arc to another then back again. The line that unites the arcs’ centers intersects each arc twice.](image)

Notice that it is not necessarily the case that if one can get from a sample A to a sample B, one could also move backwards. Two way travel is possible only if the line that joins the centers of the samples’ \( X_2 \) circles intersects each sample’s arc twice (see fig. 2).

**B. Planar car**

A two dimensional car with steerable front wheels (see fig. 1 (b)). Its model is

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{\theta} \\
\dot{\phi}
\end{bmatrix} = \begin{bmatrix}
cos(\theta) \\
\sin(\theta) \\
\tan(\phi)
\end{bmatrix} \cdot u_1 + \begin{bmatrix}
0 \\
0 \\
0
\end{bmatrix} \cdot u_2
\]

(4)

where \( x \) and \( y \) give the position of the car in the plane, \( \theta \) is the heading angle and \( \phi \) is the steering angle. \( h \) is the distance between the front and back axles. The car is assumed able to also go in reverse.

The car model is fully kinematically reducible [1], but for it a different approach is illustrated, that of the maneuver automaton [4]. A basic “trim” maneuver is for the car to simply go in a straight line with direction given by the heading, with steering angle \( \phi \) equal to 0. This trim maneuver will be referred to as a driving segment, and the states that the planner uses are such driving segments.

Generating a driving segment goes as follows: a random configuration is selected for the car in the workspace. If it is valid (it does not intersect with obstacles) then the longest drive segment that passes through that configuration and does not collide with obstacles is used.

Steering between two drive segments is done by a sequence of three maneuvers (see fig. 1 (e)). First, the car moves toward the destination segment while quickly increasing its steering angle. Second, the car moves in a circular arc with constant steering angle, and the length of the arc is varied such that it will meet the destination drive segment properly after the third maneuver. The third maneuver is the car still moving forward and quickly decreasing its steering angle to 0. The forward velocity of the car is kept constant throughout all maneuvers.

The three maneuvers can be thought of as transformations that move an initial configuration:

\[
x_{fin} = R_{\phi[0]} \cdot R_{\phi=\text{max} (\theta_d)} \cdot R_{\phi=0} \cdot x_{ini}
\]

(5)

The first and last maneuvers (\( R_{\phi=\text{max} (\theta_d)} \) and \( R_{\phi=0} \)) result in fixed displacements in the configuration variables. It is the second (\( R_{\phi=\text{max} (\theta_d)} \)) that can be tuned, by making the arc longer or shorter, so as to change the heading of the car from the heading of the initial drive segment to that of the destination drive segment. Once the necessary \( \theta_d \) angle for the arc is known, the displacement in position can be computed, and therefore one can determine the points on the two segments between which the steering maneuvers occur.

Since in this model the car’s maneuvers are fully reversible, the direction the car is facing on a drive segment is unimportant. It can do the same maneuvers independent of which way it is facing. Drive segments are then undirected.

**C. Planar robot with trailer**

A robot with a trailer attached to it via a rigid rod articulated at the robot (see fig. 1 (e)). The configuration variables are \( x_r, y_r, \theta \) for the robot position and heading, and likewise \( x_t, y_t \) and \( \theta_t \) for the trailer. \( h \) is the distance between the trailer’s and the robot’s centers. Because the system is differentially flat, there are certain relations between the robot and trailer variables [11]:

\[
\begin{bmatrix}
x_r \\
y_r
\end{bmatrix} = \begin{bmatrix}
x_t \\
y_t
\end{bmatrix} + \begin{bmatrix}
cos(\theta_t) \\
\sin(\theta_t)
\end{bmatrix} \cdot h
\]

(6)

\[
tan(\theta_t - \theta) = h \cdot \frac{d\theta_t}{ds}
\]

(7)
where $s$ is the natural parametrization of the curve traced by the trailer, or in other words the time variable if the speed of the trailer is equal to 1. Therefore, if one is given a trajectory for the trailer, one can deduce the trajectory that the robot needs to follow, and this fact is used for the steering procedure.

Just like in the case of the car, the planner uses drive segments as states, where drive segment here means a maneuver in which the robot and trailer are aligned and moving with constant heading. It is assumed that a drive segment is reversible (the robot can either push or pull the trailer in a straight line). However, unlike the case of the car, it is important which way the robot with trailer is facing in a drive segment. Therefore the drive segment is directed, because maneuvers that make the robot leave the drive segment are not reversible.

Steering from one drive segment to another is done by imposing a circular arc trajectory for the trailer that joins points on the two drive segments (see fig. 1 (f)). The circle is generated with a fixed turning radius. From the trailer trajectory the robot’s trajectory can then be deduced using the formulae above.

![Fig. 3. Going from one drive segment to another then back again. The drive segments need to overlap at one point and have long enough tails on either side of that point.](image)

Even if the robot with trailer can be steered from a drive segment A to a drive segment B, it doesn’t necessarily follow that it can be steered from B back to A using the procedure above. For that to be possible, the robot must have enough room to reverse on the drive segment B, and enough room on segment A, to fit a circle arc maneuver between them (see fig. 3).

### IV. SIMULATION RESULTS

The planar object with variable direction thruster is placed in a simple maze and required to go from a given start configuration to a target one (see fig. 4 (a)). Ten runs are given to the algorithm, and it needs on average 4 vertices in the roadmap in order to be able to connect the given start and target configurations.

All vehicles are then placed in a maze containing ”bug traps” (areas with narrow entrances, see fig. 4 (b), (c), (d)), and given one hundred runs each. Average and standard deviation values for the number of samples stored in the roadmap before connecting start to target was possible is given in table I for the classical planner (no visibility heuristic) and in II for the visibility based planner.

Each planner is tested in a version using point samples, which is the typical approach, by performing one hundred runs for each problem. Sample count statistics are similar. Relative average execution times for the planners, non-zero dimensional vs. point samples, are given in III.

<table>
<thead>
<tr>
<th>Problem</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>1770.6</td>
<td>14.71</td>
<td>239.25</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>1697.28</td>
<td>12.86</td>
<td>196.82</td>
</tr>
<tr>
<td>Minimum</td>
<td>93</td>
<td>3</td>
<td>67</td>
</tr>
<tr>
<td>Maximum</td>
<td>7769</td>
<td>73</td>
<td>859</td>
</tr>
</tbody>
</table>

**Table I**

**AVERAGE, STANDARD DEVIATION, MINIMUM AND MAXIMUM OF PLANNER SAMPLE COUNTS OVER A HUNDRED RUNS FOR THE CLASSICAL PLANNER. PROBLEMS ARE: (A) PLANAR OBJECT WITH VARIABLE DIRECTION THRUSTER; (B) PLANAR CAR; (C) ROBOT WITH TRAILER**
The fully reversible vehicle, the planar car, needs the fewest samples in order to solve the planning problem. It’s also the least sensitive to unlucky sampling. The other two vehicles tend to need more samples on average, but it also is the case that the number of samples used varies widely from run to run.

Of the two non-reversible vehicles, the robot and trailer needs fewer samples. This is expected to be because its sample subspaces tend to be larger than those of the planar object: drive segments can be longer than $X$ needs fewer samples. This is expected to be because its use is particularly efficient for this vehicle model; most of the variable radius visibility based planner algorithm was applied to problems in which not all maneuvers of the problem system are reversible and resulted in compact roadmaps that could solve the given planning queries. Besides using local planners adapted to models of real vehicles, the planner was also modified to use non-zero dimensional sample subspaces, a different approach than classical sampling based planners, so as to ease verification of maneuver sequences generated by local planning.

The method presented is applicable to cases where the environment in which the vehicle moves is known at the start or becomes known through some mapping procedure. The method then provides a compact, easy to search library of trajectories through this environment. If changes invalidate some of those trajectories, one can resume growing the sparse roadmap, in the manner described here, to find alternate routes around the changes.

"Luck" while sampling plays a part in determining how many samples are needed in the roadmap. Future work will attempt to define some heuristic criteria for what is a "good" sample and bias the sampler to fulfill them, so as to get less spread in roadmap size and smaller roadmaps in general.

The sample subspace shapes in this paper were designed by intuition and imposing a necessary condition. A more rigorous design procedure might be developed in future work, which will focus on vehicles or articulated bodies moving in three dimensional workspaces.

### REFERENCES


