Feedback Motion Planning via Non-holonomic RRT* for Mobile Robots

Jong Jin Park[1] and Benjamin Kuipers[2]

***Abstract*— Here we present a non-holonomic distance function for unicycle-type vehicles, and use this distance function to extend the optimal path planner RRT* to handle non-holonomic constraints. The critical feature of our proposed distance function is that it is also a control-Lyapunov function. We show that this allows us to construct feedback policies that stabilizes the system to a target pose, and to generate the optimal path that respects the non-holonomic constraints of the system via the non-holonomic RRT*. The composition of the Lyapunov function that is obtained as a result of this planning process provides stabilizing feedback and the cost-to-go to the final destination in the neighborhood of the planned path, adding much flexibility and robustness to the plan.**

I. INTRODUCTION

For efficient and intelligent navigation, a mobile robot planning a path must have a good measure of how far a given pose (position and orientation) in navigable space is to another pose. For mobile robots with non-holonomic constraints (e.g. robots that cannot move sideways), the widely-used Euclidean distance in Cartesian coordinates is clearly not a sufficient metric for this, since it fails to capture the constraints and does not reflect the true cost-to-go of the system. (Fig.1.)

This incompatibility of the Euclidean distance for non-holonomic systems is a well-known problem. In fact, it has been shown that for sampling-based planners such as RRT [1] the space is efficiently explored only when this distance function reflects the true cost-to-go [2]. But this has been getting more attention recently (e.g., see [3] [4] [5] [6]) since the introduction of RRT* [7], the sampling-based motion planner that guarantees asymptotic optimality, which critically requires a proper distance function. In the original formulation [7] Euclidean metric was used to measure distance, which is a proper metric only for holohomic systems.

Several attempts have been made to extend RRT* to non-holonomic systems, where many recent works focus on some specific form of steering function that connects two states and the cost-to-go under the control. In [4], the distance between a pair of states is measured based on a fixed-final-state/free-final-time controller and a cost function on time and control effort. In [5] and [8], LQR-based cost functions are used

This work has taken place in the Intelligent Robotics Lab in the Computer Science and Engineering Division of the University of Michigan. Research of the Intelligent Robotics lab is supported in part by grants from the National Science Foundation (CPS-0931474, IIS-1111494, IIS-1252987, and IIS-1421168).

[1]J. Park is with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109, USA `jongjinp@umich.edu`

[2]B. Kuipers is with Faculty of Computer Science and Engineering, University of Michigan, Ann Arbor, MI 48109, USA `kuipers@umich.edu`

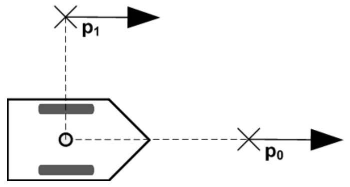

Fig. 1. Consider two poses p_0 and p_1. Although p_1 is nearer the robot in Euclidean distance, it is harder to get to due to differential constraints. In this paper, we propose a directed distance function applicable to unicycle-type vehicles, that properly reflects the true cost-to-go of the system under the non-holonomic constraint.

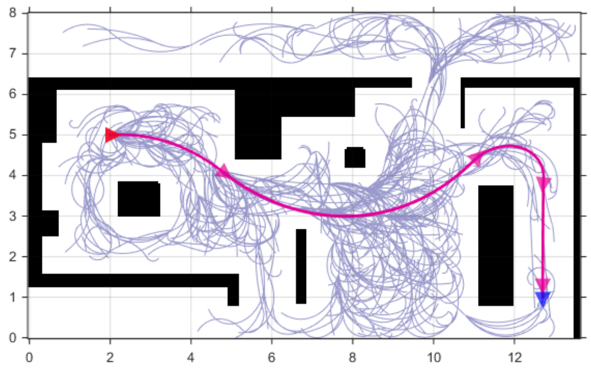

Fig. 2. (Best viewed in color) An example minimum-distance path (bold line) found by our non-holonomic RRT* after 1000 vertices, using the proposed distance function (10). The path exactly connects the starting pose at top left facing right (red triangle) and destination pose at bottom right facing downward (blue triangle) in a cluttered 14m$\times$8m office environment. The vertices explored (grey) are also shown. Note that our algorithm rejects candidate paths with shorter Euclidean distance, such as paths that cross below the rightmost obstacle and reach the destination facing in the wrong direction.

to measure distance. Both [4] and LQR-based methods [5] [8] rely on local linearization of dynamics for non-linear systems. In [6], an approximation of the path-integrated cost is used as the distance metric, where the approximation is learned from simulated trajectories of a differential-drive vehicle under a stabilizing control law. A different approach [3] was proposed by the authors of [7], which does not concern the steering or the cost-to-go. The key idea of [3] is to use a weighted Euclidean box which is narrower in the direction the motion is constrained, rather than an Euclidean ball, when identifying nearby neighbors, thus resulting in less number of bad candidate connections and improved efficiency of the RRT*.

In this paper, we present a parameterized closed-form distance function from a pose (position and orientation) to a target pose for unicycle-type vehicles (e.g. differentially driven mobile robots), and an optimal sampling-based planner using the distance function (the non-holonomic RRT*), where the free parameters in the distance function controls the shape of the resulting path. The critical feature of our proposed distance function is that it is also a control-Lyapunov function (CLF) for the system, which makes it a natural measure of cost-to-go. We show that it is straightforward to develop feedback policies given the distance function, and develop a non-holonomic RRT* based on this distance and the derived policy. With our approach, the non-holonomic RRT* not only generates an optimal path but also a composition of Lyapunov functions (so-called funnels [9] [10] [11] [12]) that provides stabilizing feedback and the cost-to-go to the final destination in the neighborhood of the planned path, adding much flexibility and robustness to the plan.

In Section II, we introduce the distance function and the coordinate system the distance function is defined in, and present a Lyapunov stability analysis based on two-time scale decomposition. The non-holonomic RRT* algorithm using this distance function is presented in Section III, followed by results and discussion in Section IV.

II. Egocentric Polar Coordinates and Non-holonomic Distance

We define our distance function on a smooth manifold defined by the egocentric polar coordinates [13]. This egocentric polar coordinate system (Fig.3.) describes the relative location of the target pose observed from a vehicle with radial distance r, orientation of the target ϕ, and orientation of the vehicle δ, where angles are measured from the line of sight vector from the vehicle to the target. Formally, we write

$$\mathscr{T}_{\mathrm{p}_0} : (x, y, \theta)^T \mapsto (r, \phi, \delta)^T \tag{1}$$

to denote the mapping from the usual Cartesian coordinates to the egocentric polar coordinates, where p_0 is the pose of the target and $\mathrm{p} = (x, y, \theta)^T$ is the pose of the vehicle.

The kinematics of the vehicle in the egocentric polar coordinates can be written as

$$\begin{pmatrix} \dot{r} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} -v\cos\delta \\ \frac{v}{r}\sin\delta \end{pmatrix} \tag{2}$$

$$\dot{\delta} = \frac{v}{r}\sin\delta + \omega \tag{3}$$

where (2) describes the dynamics of the vehicle in the position subspace, and (3) describes the dynamics of the vehicle in the heading (steering) subspace. This essentially decomposes the system to a slow (position) subsystem and a fast (steering) subsystem. Note that in the slow subsystem (2), the heading δ is a (virtual) control.

In the position subspace, we use the following weighted norm to measure how far a point $(r, \phi)^T$ is to the origin (target pose):

$$l^-(r, \phi) = \sqrt{r^2 + k_\phi^2\,\phi^2} \tag{4}$$

where k_ϕ is a positive constant that represent the weight of ϕ with respect to r. Note that this is a metric in polar coordinates, and the orientation of the target pose is incorporated in ϕ.

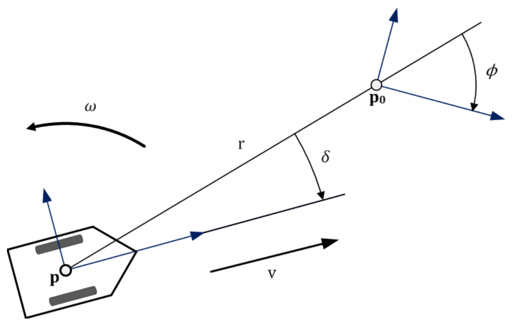

Fig. 3. Egocentric polar coordinate system with respect to the vehicle. From an observer situated on a vehicle fixating at target pose p_0, r is the radial distance to the target, θ is the orientation of p_0 with respect to the line of sight from the observer to the target, and δ is the orientation of the vehicle heading with respect to the line of sight. Here, both θ and δ have negative values. At $r = 0$, we take the line of sight to be aligned with the target orientation and set $\phi = 0$, and δ is the the vehicle orientation measured from the target orientation.

It is trivial to show that (4) is a CLF for the virtual control δ, as there always exists some heading that reduces this positive definite function. For example, simply setting the desired heading at $\delta^* = 0$ (which points directly to target pose) results in the classic turn-travel-turn approach. Here we show the origin is Lyapunov-stable under the following feedback examples, assuming non-zero positive velocity v:

$$\delta_s^*(\phi) = \mathrm{atan}(-k_\phi \phi) \tag{5}$$

$$\delta_g^*(r, \phi) = \mathrm{atan}(-k_\phi^2 \phi / r^2) \tag{6}$$

where (5) is the heading that reduces r and ϕ very smoothly at ratio $\dot{\phi}/\phi = k_\phi \dot{r}/r$ (see [13]), and (6) is the gradient of (4) along (2). As will be shown in the results, these are very distinct and useful steering strategies, with (5) generating very smooth curves and (6) generating curves that quickly approaches the target pose and then aligns to the target orientation. Each of these feedback control laws for the heading (5)-(6), which specifies a heading vector at every point in the position space by construction, describes a stabilizing *vector field*. This vector field encodes a collection of paths converging to the target over the entire space, with tangents of the paths aligned with the vector field (Fig.4). We use this to connect two nodes in the RRT tree, in Extend() and Steer() processes as described in Section III.

To show the origin is Lyapunov stable, we need to check the sign of the derivative of the positive definite function (4) along the position dynamics (2) under the control (5) - (6). We have

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}(l^-(\cdot)^2) &= 2r\dot{r} + 2\phi\dot{\phi} \\ &= -2rv\cos(\delta_{[\cdot]}^*(\cdot)) + \frac{2v}{r}\,\phi\,\sin(\delta_{[\cdot]}^*(\cdot)) \end{aligned}$$

strictly less than zero everywhere other than $r = 0$, since

$$\begin{array}{rclrcl} \cos(\delta_{[\cdot]}^*(\cdot)) & > & 0, & \because \delta_{[\cdot]}^*(\cdot) & \in & (-\pi/2, \pi/2) \\ sgn(\delta_{[\cdot]}^*(\cdot)) & = & -sgn(\phi), & \forall\phi & \in & (-\infty, \infty) \end{array} \tag{7}$$

due to property of $\mathrm{atan}(\cdot)$ and $r \geq 0$ and $v > 0$ by definition. Thus $\frac{\mathrm{d}}{\mathrm{d}t} l^-(\cdot)^2$ is also negative definite, and the origin (target pose) is Lyapunov stable under $\delta_{[\cdot]}^*(\cdot)$.

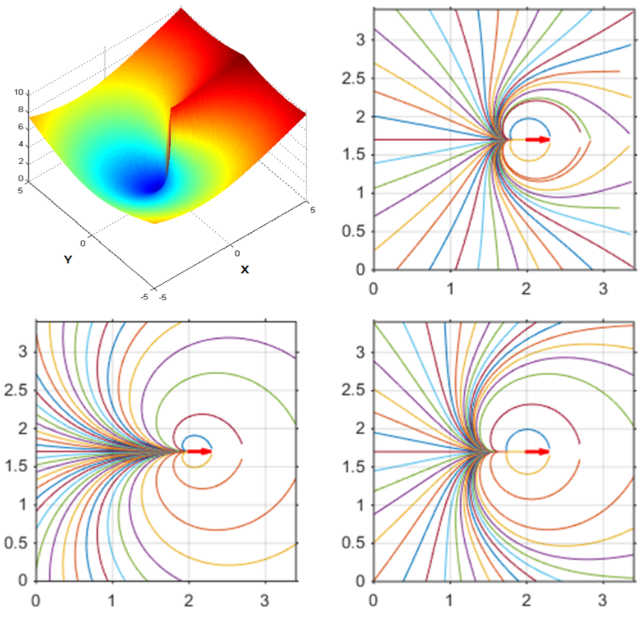

Fig. 4. Example paths following the user-specified vector fields, and visualization of (reduced) distance function. These stabilizing vector fields and the shape of the resulting paths can be tuned to user's preference with a positive weight k_ϕ, where higher k_ϕ results in quicker reduction in angular coordinate ϕ. Note that with (5) (bottom left) the paths always curves in smoothly to the target pose regardless of initial position, but with (6) (right column) large portion of the paths shown are almost straight lines until it is near the target and then elbows in to the target orientation. **Top Left**: Illustration of distance from various positions to a target pose at $(0, 0, 0)^T$ (facing to the positive x-axis), assuming the heading is aligned with the specified vector field, so the distance function (10) reduces to (9). Note the singularity at origin and the high cost at positions along positive x-axis, where the the vehicle have to make a large arc to move to the target pose ($k_\phi = 1.0$). **Bottom Left**: Smooth curves following (5), with $\dot{\phi}/\phi = k_\phi \dot{r}/r$ ($k_\phi = 1.5$). Target pose is depicted with a red arrow at $(2, 1.7, 0)^T$. **Top Right**: Sample paths following (6), the gradient of (9) along the dynamics (2) ($k_\phi = 0.8$). **Bottom Right**: Sample paths following (6) ($k_\phi = 1.5$). Note that the angular coordinate ϕ is reduced much quicker compared to figure on top right.

These vector fields are examples of so-called *funnels* in feedback motion planning [9]. A funnel [10] [11] can be described as a (locally) valid feedback policy or a Lyapunov function, which takes a broad set of initial conditions to a local subgoal. *Feedback motion planning* is a process of finding a sequential composition of funnels that takes the system to the final destination, which can be more robust and flexible than simple path planning. See [14] [15] for the use of vector fields as funnels; [12] is an excellent example of feedback motion planning based on local linearizations and LQR policies. Existing approaches tend to focus on finding local controllers due to usual difficulty in finding a general Lyapunov function. One of the contribution of this paper is the presentation of (9), which is a control-Lyapunov function for unicycle-type vehicles, which is our funnel.

The domain of attraction for this Lyapunov function (i.e. the mouth of the funnel) is the entire configuration space for a unicycle in completely free space, but in cluttered environments (or for curvature-constrained vehicles) it needs to be estimated numerically. We can make a conservative estimate by tracing the vector field, as shown in Fig.5.

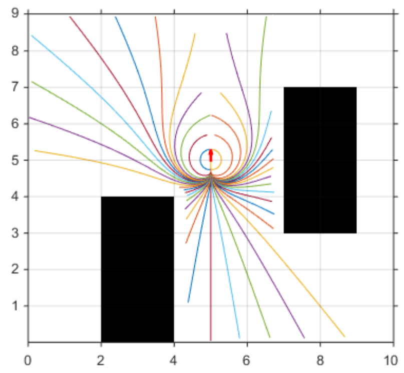

Fig. 5. Estimating the domain of attraction for a Lyapunov function for a unicycle by tracing the vector field in environment with obstacles. Here a target pose(red arrow) is at $(5, 5, \pi/2)^T$, and the paths follow (6) with $k_\phi = 1.2$.

Now, given the stabilizing vector field in position space characterized by the desired heading $\delta^*_{[\cdot]}(\cdot)$, we can define the heading error with respect to the vector field as

$$\delta_e(r, \phi, \delta) = |\delta - \delta^*_{[\cdot]}(\cdot)| \tag{8}$$

which is the distance between the robot orientation and the desired heading $\delta^*_{[\cdot]}(\cdot)$. With this we can define a CLF in the full configuration space, where linear and angular velocities are treated as control inputs,

$$\begin{aligned} l(r, \phi, \delta) &= l^-(r, \phi) + k_\delta\, \delta_e(r, \phi, \delta) \\ &= \sqrt{r^2 + k_\phi^2\, \phi^2} + k_\delta |\delta - \delta^*_{[\cdot]}(\cdot)| \end{aligned} \tag{9}$$

where k_ϕ and k_δ are positive constants. Note that the choice of k_ϕ controls the shape of the local path segments via $\delta^*_{[\cdot]}(\cdot)$, and k_ϕ and k_δ influences the distance definition, i.e. how far the origin is from a pose.

This is a CLF for kinematic inputs since there always exists some linear and angular velocity (v, ω) that reduces this positive definite function.[1] [2] Note that this is a weighted L1 norm of the distances in position (4) and heading (8). The choice of L1 norm is motivated by the fact that minimizing L1 norm tends to reduce one of the terms first, while minimizing L2 norm tends to reduce all terms simultaneously. As will be seen in the Section IV, this choice of L1 norm tends to create very smooth path, as it strongly motivates the planner to compose funnels that align well, i.e. have minimal heading error at joints.

Now, we define the non-holonomic directed distance from a pose $\mathrm{p} = (x, y, \theta)^T$ to a target pose p_0 via (1) and (9)

$$\mathrm{Dist}(p, p_0) = l(\mathscr{T}_{\mathrm{p}_0}((x, y, \theta)^T)) \tag{10}$$

[1]A unicycle can always rotate in place to align to the vector field, and move forward if the heading is well aligned.

[2]Although out of scope of this paper, we note that it is straightforward to develop kinematic control laws from the presented vector fields (the virtual control for the position subsystem). The key idea is to guarantee that the heading error (8) is reduced sufficiently faster than the position error (4) (via two-time scale decomposition [16]) so that the vehicle always converges to the virtual control, and ultimately to a small neighborhood of target pose. See [13] for derivation of a smooth kinematic control law with the vector field (5) as the virtual control for the position subsystem.

Recall that this measures the distance assuming vehicle is moving forward. Flipping both p and p_0 by 180 degrees yields the distance for backward motion. This function is positive definite but is not symmetric, i.e. Dist(p, p_0) $\neq$ Dist(p_0, p). Also, this distance function is similar to the radial distance r when $r \gg 1$, while promoting alignment to target orientation as $r \to 0$.

In the next section, we use the heading control $\delta^*_{[\cdot]}(\cdot)$ as our steering function and (9) as our distance and cost function to construct non-holonomic RRT* for unicycle-type vehicles.

III. Non-holonomic RRT* for Unicycles

RRT* [7] is an incremental, sampling based planner with guaranteed asymptotic optimality, originally developed for holonomic systems. The algorithm (Alg. 1-3) is an extension of the RRT* for unicycle-type vehicles, using the steering and the distance function developed in the previous section. It solves the optimal path planning problem by growing a tree $\mathcal{T} = (V, E)$ with vertex set V of poses connected by edges E of feasible path segments to find a path that connects exactly to the destination pose with minimum cost, using the set of basic procedures described below. X is the configuration space of the vehicle, and $x : [0, 1] \to X$ is a path in the configuration space. The notations generally follows the RRT* algorithm presented in [17].

Sampling: The Sample() process samples a pose $z_{\text{rand}} \in X_{\text{free}}$ from obstacle-free region of the configuration space. The sampling is random with a goal bias, with which the destination pose is sampled to ensure the path exactly connects to the destination pose.

Distance: Dist(z, z_0) as defined in (10) returns the directed distance from a *pose* z to a *target pose* z_0.

Cost: The cost function $c(z, z_0)$ measures the cost of the path from a pose z to a pose z_0. In this paper, we are finding the minimum-distance path, so $c(z, z_0) = \text{Dist}(z, z_0)$. Cost($v$) returns the accumulated cost of a node $v \in V$ in the tree from the root.

Nearest Neighbor: Given a pose $z \in X$ and the tree $\mathcal{T} = (V, E)$, $v = \text{Nearest}(\mathcal{T}, z)$ returns the node in the tree where the distance from the node to the pose Dist(v, z) is minimum.

Near-by nodes: Given a pose z, tree $\mathcal{T} = (V, E)$, and a number n, we have

$$\text{NearTo}(\mathcal{T}, z, n) \equiv \{v \in V \,|\, \text{Dist}(v, z) \leq L(n)\} \qquad (11)$$

where $L(n) = \gamma(\frac{log(n)}{n})^{(1/d)}$ with constant γ and dimension of the space d [7], where we have $d = 3$ for our configuration space. This returns set of nodes from which the distance *to* the pose z is small. Similarly,

$$\text{NearFrom}(\mathcal{T}, z, n) \equiv \{v \in V \,|\, \text{Dist}(z, v) \leq L(n)\} \qquad (12)$$

returns the set of nodes that is near *from* the pose z. This distinction between NearTo() and NearFrom() is necessary due to the use of directed distance.

Algorithm 1: $\mathcal{T} = (V, E) \leftarrow$ Nonholonomic RRT*(z_{init})

```
1  $\mathcal{T} \leftarrow$ InitializeTree();
2  $\mathcal{T} \leftarrow$ InsertNode($\emptyset, z_{\text{init}}, \mathcal{T}$);
3  for $i = 1$ to $N$ do
4      $z_{\text{rand}} \leftarrow$ Sample($i$);
5      $z_{\text{nearest}} \leftarrow$ Nearest($\mathcal{T}, z_{\text{rand}}$);
6      $(z_{\text{new}}, x_{\text{new}}) \leftarrow$ Extend($z_{\text{nearest}}, z_{\text{rand}}, \epsilon$);
7      if ObstacleFree($x_{new}$) then
8          $Z_{\text{nearTo}} \leftarrow$ NearTo($\mathcal{T}, z_{\text{new}}, |V|$);
9          $z_{\text{min}} \leftarrow$ ChooseParent($Z_{\text{nearTo}}, z_{\text{nearest}}, z_{\text{new}}$);
10         $\mathcal{T} \leftarrow$ InsertNode($z_{\text{min}}, z_{\text{new}}, \mathcal{T}$);
11         $Z_{\text{nearFrom}} \leftarrow$ NearFrom($\mathcal{T}, z_{\text{new}}, |V|$);
12         $\mathcal{T} \leftarrow$ ReWire($\mathcal{T}, Z_{\text{nearFrom}}, z_{\text{min}}, z_{\text{new}}$);
13 return $\mathcal{T}$
```

Algorithm 2: $z_{\text{min}} \leftarrow$ ChooseParent$(Z_{\text{nearTo}}, z_{\text{nearest}}, z_{\text{new}})$

```
1  $z_{\text{min}} \leftarrow z_{\text{nearest}}$;
2  $c_{\text{min}} \leftarrow$ Cost($z_{\text{nearest}}$) + $c(z_{\text{nearest}}, z_{\text{new}})$;
3  for $z_{near} \in Z_{nearTo}$ do
4      $x' \leftarrow$ Steer($z_{\text{near}}, z_{\text{new}}$);
5      if ObstacleFree($x'$) then
6          $c' =$ Cost($z_{\text{near}}$) + $c(z_{\text{near}}, z_{\text{new}})$;
7          if $c' <$ Cost($z_{new}$) $and$ $c' < c_{min}$ then
8              $z_{\text{min}} \leftarrow z_{\text{near}}$;
9              $c_{\text{min}} \leftarrow c'$;
10 return $z_{min}$
```

Algorithm 3: $\mathcal{T} \leftarrow$ ReWire$(\mathcal{T}, Z_{\text{nearFrom}}, z_{\text{min}}, z_{\text{new}})$

```
1  for $z_{near} \in Z_{nearFrom} \backslash \{z_{min}\}$ do
2      $x' \leftarrow$ Steer($z_{\text{new}}, z_{\text{near}}$);
3      if ObstacleFree($x'$) $and$
       Cost($z_{new}$) + $c(z_{new}, z_{near})$ < Cost($z_{near}$) then
4          $\mathcal{T} \leftarrow$ ReConnect($z_{\text{new}}, z_{\text{near}}, \mathcal{T}$);
5  return $\mathcal{T}$
```

Steering: We assume the vehicle follows the vector field, e.g. the feedback control on the heading $\delta^*_{[\cdot]}(\cdot)$ described in (5) and (6).[3] We have $x = \text{Steer}(z, z_0)$ which generate a path segment x that starts from z and end exactly at z_0, and $(z_{\text{new}}, x_{\text{new}}) = \text{Extend}(z, z_0, \epsilon)$ starts from z and extend the path toward z_0 until z_0 is reached or the distance covered is ϵ, and returns the new sample z_{new} at the end of the extension. Extend() is a standard RRT procedure for exploration. In our experiments, we obtained best results when we used Extend() for exploration (not forcing exact connection to a target sample) and Steer() for choosing the best parent node and to rewire graph, as described in Alg. 1-3.

[3]Assuming initial condition $\delta_e = 0$, the heading control $\delta^*_{[\cdot]}(\cdot)$ can be exactly followed with $\omega = \dot{\delta}^* - \frac{v}{r}\sin\delta^*$ via (3), where the positive linear velocity v can be chosen so that the accelerations and velocities are bounded. See footnote 2 in the previous section for discussion on more general controllers.

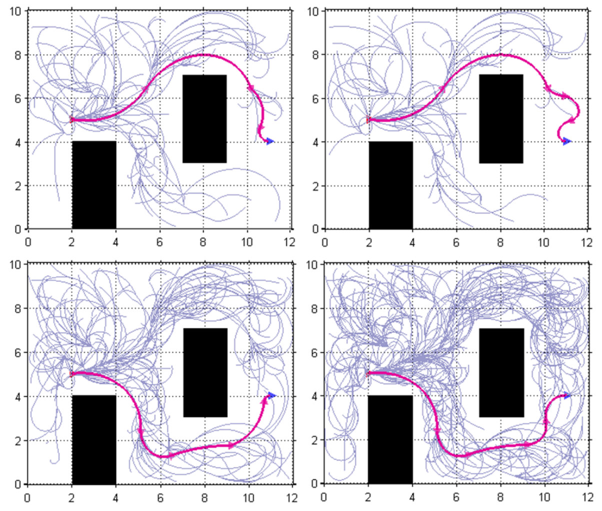

Fig. 6. Incremental sampling process performed by the proposed algorithm, using the distance function (10) via (5), with weights $k_\phi = 1.2$, $k_\delta = 3.0$. **From top right, counterclockwise**: The tree constructed by the non-holonomic RRT^* with 112, 163, 355, and 731 vertices. The algorithm first found a path across the larger open space in top area, but eventually finds shorter distance path through bottom, meanwhile improving the overall smoothness of the path.

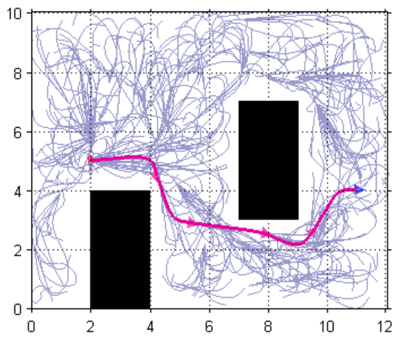

Fig. 7. A minimum-distance path in the tree of 907 vertices, with the proposed algorithm using the distance function (10) via (6), with weights $k_\phi = 1.2$ and $k_\delta = 3.0$. Note that with (6), the path is composed of L-shaped segments, which quickly moves toward a target pose (the next node) to reduce the radial distance and then elbows in to the target orientation.

Collision Checking: The ObstacelFree(x) function checks whether a path x lies within the obstacle-free region of the configuration space, considering the size and the shape of the robot. Note that any mission-critical constraints (e.g. path curvature, minimum clearance, etc.) that needs to be checked can be added here.

Node Insertion: Given the current tree $\mathcal{T} = (V, E)$ and a node $v \in V$, the InsertNode(v, z, $\mathcal{T}$) adds z to V and create an edge to v from z.

With theses basic processes, the non-holonomic RRT* explores the configuration space by sampling and extending as the classic RRT [1] (Alg.1), and considers all nearby nodes of a sample to choose the best parent node and rewires the graph to streamline the tree as the RRT* (Alg.2-3) where ChooseParent() and ReWire() processes guarantee asymptotic optimality. We provide proper distance functions and exact steering for unicycle-type vehicles.

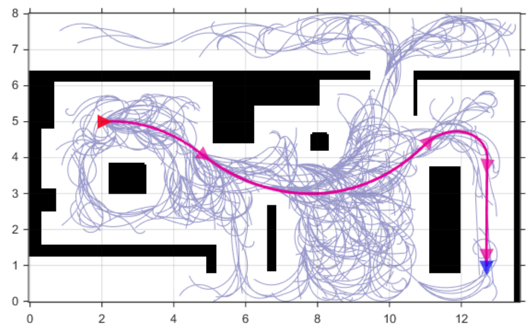

Fig. 8. (Best viewed in color) The minimum-distance path found after 1000 vertices explored in cluttered 14m×8m office environment via (5) and (10), with $k_\phi = 1.2$ and $k_\delta = 3.0$. The proposed algorithm is able to generate highly smooth and intuitive paths without requiring any extra processing.

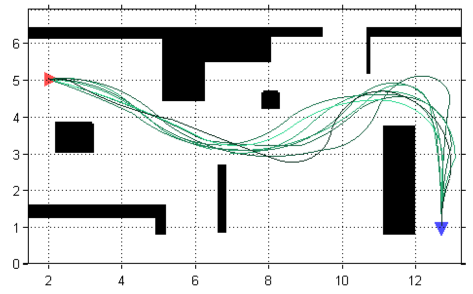

Fig. 9. (Best viewed in color) Multiple (suboptimal) solutions obtained over several runs of RRT*. Green to black color gradient indicates low to high cost of the path.

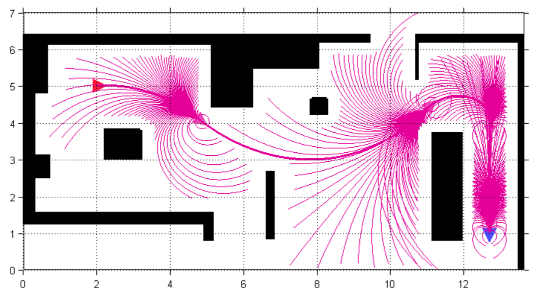

Fig. 10. Visualization of implicit paths over the domain of attraction of each Lyapunov function (so-called funnels) attached to each node in the graph. These represents the nominal paths the vehicle will take when deviated from the original path.

IV. Results and Discussion

In this section we demonstrate the non-holonomic RRT* with varying steering functions (Fig. 6-10). Recall that the distance function also depends on the steering function, so that the planning is integrated with the control. Overall, the algorithm is able to accommodate the two example controls (5)-(6) very well, with (5) leading to very smooth and elegant paths, while the use of (6) generates more aggressive composition of L-shaped curves.

Fig. 6 illustrates the incremental sampling process, using (5) with weights $k_\phi = 1.2$ and $k_\delta = 3.0$. Notice that the quality of the path, especially the smoothness of the curve, gradually improves as the number of vertices increase. The high value of k_δ heavily penalizes heading error between the sampled pose and the subsequent path segment, which enforces the overall smoothness. k_ϕ determines the shape of local path segments via steering. For comparison, Fig. 7 shows an example path obtained using (6) with the same parameters $k_\phi = 1.2$, $k_\delta = 3.0$. Fig. 6-7 shows both (5) and (6) produce satisfactory, yet qualitatively distinct, paths.

Fig. 8 shows an example minimum-distance path found in cluttered office environment, using (5) for the steering and for the distance definition. Shorter, smoother paths are selected over longer, less smooth ones, as desired (Fig. 9).

Fig. 10 shows implicit paths over the domain of attraction of each Lyapunov function along the nodes in the path, constructed by tracing the stabilizing vector field attached to each node. Note that at some pose z within the domain of attraction of a node v, the cost-to-go to the final destination is simply $Cost(g) - Cost(v) + Dist(z, v)$ where g is the goal node at the destination pose.

Compared to the original RRT* equipped with Euclidean distance, the proposed algorithm is much more efficient due to proper identification of the nearest and nearby neighbors. We observed up to three times speedup, in agreement with the results reported in [3].

Having a good measure for the distance between two configurations is very important for planning an optimal path, via sampling or over grid. To be a good measure, this (distance) function must reflect the true cost-to-go between configurations, and respect the constraints of the system. It needs to be positive definite; it should always be possible to be decreased with some control input, so that it does not create local minima and can be used to generate exact steering. That is, it needs to be qualified as a control-Lyapunov function. It should be possible to extend RRT* to any system where a control-Lyapunov distance function is available. We have provided such a distance measure for unicycle-type vehicles.

V. Conclusion

Here we present a non-holonomic distance function for unicycle-type vehicles, and use this distance function to extend the optimal path planner RRT* to handle non-holonomic constraints for unicycle-type vehicles. The critical feature of our proposed distance function is that it is also a control-Lyapunov function, so it better represents the true cost-to-go between configurations and properly reflects the constraints of the system. It allows us to readily generate smooth, intuitive, and feasible paths. By using provably stable control laws and the closed-form distance function that properly reflects the constraints, our algorithm finds smooth and precise paths that exactly reaches the goal for unicycle-type vehicles, and provides stabilizing vector field and the cost-to-go to the final destination around the planned path by composition of local control-Lyapunov functions.

For future work, we are particularly interested in utilizing the global cost-to-go obtained by the composition of the control-Lyapunov function. Availability of the control-Lyapunov cost-to-go can be highly beneficial, since it guarantees that the robot will reach the goal as long as it reduces this cost-to-go, while allowing much freedom in how to do so. We are also interested in improving the current algorithm, such as finding control-Lyapunov distance function for other systems, using more efficient sampling process for faster convergence, and extending the cost definition to account for local curvature and proximity to obstacles.

References

[1] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” Computer Science Dept., Iowa State University, Tech. Rep. 98-11, Oct 1998.

[2] P. Cheng and S. M. LaValle, “Reducing metric sensitivity in randomized trajectory design,” in *2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1. IEEE, 2001, pp. 43–48.

[3] S. Karaman and E. Frazzoli, “Sampling-based optimal motion planning for non-holonomic dynamical systems,” in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[4] D. J. Webb and J. v. d. Berg, “Kinodynamic RRT*: Optimal motion planning for systems with linear differential constraints,” *arXiv preprint arXiv:1205.5088*, 2012.

[5] G. Goretkin, A. Perez, R. Platt, and G. Konidaris, “Optimal sampling-based planning for linear-quadratic kinodynamic systems,” in *2013 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 2429–2436.

[6] L. Palmieri and K. O. Arras, “Distance metric learning for rrt-based motion planning for wheeled mobile robots,” in *2014 IROS Machine Learning in Planning and Control of Robot Motion Workshop*, Sep 2014.

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

[8] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, “LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics,” in *2012 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 2537–2542.

[9] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[10] M. T. Mason and J. K. Salisbury Jr, “Robot hands and the mechanics of manipulation,” 1985.

[11] R. Burridge, A. Rizzi, and D. Koditschek, “Sequential composition of dynamically dexterous robot behaviors,” *International Journal of Robotics Resaech*, vol. 18, pp. 534–555, 1999.

[12] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, “LQR-trees: Feedback motion planning via sums-of-squares verification,” *Int. J. Rob. Res.*, vol. 29, no. 8, pp. 1038–1052, Jul 2010.

[13] J. Park and B. Kuipers, “A smooth control law for graceful motion of differential wheeled mobile robots in 2D environment,” in *2011 IEEE International Conferernce on Robotics and Automation (ICRA)*, May 2011, pp. 4896–4902.

[14] S. R. Lindemann and S. M. LaValle, “Simple and efficient algorithms for computing smooth, collision-free feedback laws over given cell decompositions,” *The International Journal of Robotics Research*, vol. 28, no. 5, pp. 600–621, 2009.

[15] L. Zhang, S. M. LaValle, and D. Manocha, “Global vector field computation for feedback motion planning,” in *2009 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2009, pp. 477–482.

[16] H. K. Khalil, *Nonlinear Systems (3rd Edition)*, 3rd ed. New York, NY, U.S.: Prentice Hall, 2002.

[17] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, “Anytime motion planning using the rrt*,” in *2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 1478–1483.

[18] J. Park, C. Johnson, and B. Kuipers, “Robot navigation with model predictive equilibrium point control,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2012.

[19] E. Glassman and R. Tedrake, “LQR-based heuristics for rapidly exploring state space,” in *Submitted to IEEE International Conference on Robotics and Automation. ICRA*, 2010.

[20] K. Konolige, “A gradient method for realtime robot control,” in *2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1. IEEE, 2000, pp. 639–646.