Guided RRT: A Greedy Search Strategy for Kinodynamic Motion Planning

Jun Zhang, Martijn Wisse, Mukunda Bharatheesha
TU Delft Robotics institute
Delft University of Technology
Delft, the Netherlands
j.zhang-5@tudelft.nl, M.Wisse@tudelft.nl, M.Bharatheesha@tudelft.nl

***Abstract*—Sampling-based methods, such as Probabilistic Roadmap Method(PRM)[1], Rapidly-Exploring Random Tree(RRT)[2], have been proposed as promising solutions for kinodynamic problems. Nevertheless, it's still a challenge for practical application especially for complex systems. In particular, most of the forward propagation is fruitless, which lead to heavy computation and be time-consuming. This paper presents a greedy kinodynamic motion planner: Guided RRT. The main characteristics are that the environments are explored by Geometric trees previously and nodes near the geometric feasible path enjoy more preference. Instead of exploring the environments uniformly, the new approach expands towards the goal greedily along a series of waypoints, with probabilistically completeness. And to guarantee the effective of the greed, a new distance metric based on Euclidean metric are proposed by considering both the current position and the following position with zero-input. We compare our technique with standard RRT and show that it achieves favorable performance when planning under kinodynamic constraints.**

***Keywords*—motion planning, RRT, waypoint, kinodynamic, manipulator, robot**

I. INTRODUCTION

Motion planning compute a path from original state to target state is still an open and challenging problem, especially simultaneous taking kinematic and dynamic constraints into account (kinodynamic motion planning). During the last three decades, there are many notable approaches were proposed in the field of motion planning, such as Bug Algorithms, Cell Decomposition, Roadmap, Potential Field, Sampling-based motion planning and so on. While considering dynamics, most of the aforementioned methods are impractical due to the complexity: explicitly represent the environment and the high dimensionality. Sampling-based approaches are proposed as promising solutions for complex systems as well as kinodynamic problems. Examples include the probabilistic roadmap planners[1] and rapidly-exploring random trees. Different from explicitly constructing the space as the deterministic approaches which is impractical in high-dimensional cases, the sampling-based algorithms sample state in the configuration/state space and try to find a feasible path/trajectory based on the sampling state. Also, those approaches were proved to be probabilistic complete which means if once a solution exist, it will eventually find a solution when given enough time. Some sampling algorithms (RRT, EST[3], PDST[4]) use tree-based structure, which can not only effectively deal with high-dimensional situation but also incorporate dynamics into motion planning during the path planning process. Thus, it's considered to be the most suitable approaches for kinodynamic motion planning (KMP).

Since Rapidly exploring Random Trees originally presented by LaValle in [2], many variants based on tree structure were proposed and significant progress was made. One of the most successful ideas is the bidirectional RRT. Nevertheless, it's encounter with trouble when extend to differential constraints because of BVP problems [5]. [6] proposed a method by perturbation to connect points. Other examples such as ADDRRT[7], which is meritorious for geometric situation also have limitations for dynamic constraints. The tree exploration in most sampling-based motion planners slows down significantly while considering dynamic constraints.

One reason lead to be time-consuming for kinodynamic motion planning is that most of the forward propagations(>90%) are fruitless and cost a lot of time[8]. Thus, a greedy search strategy which prefers exploitation than exploration could be better, through which cut down the useless nodes and propagation. In SyCLoP[9], a lead, which is generated through workspace decomposition, is used to guide the direction of tree exploring, and the information during the tree exploring were fed back for computing leads. In [10], a projection function was used to project high dimensional state space to lower dimensional Euclidean space, and discretized by interior and exterior grid. Biasing the sampling nodes to exterior grid through which biasing the tree toward unexplored region. Another time-consuming reason is that planners are sensitive to distance metric, which is an important factor that limit the tree exploring the environment efficiently. Some valuable propagation which already take some computation time, but it is not the nearest node to the random node, according to the distance metric, are abandoned. In EST[3], KPIECE[8], instead of generating a random state and comparing the distance with the nodes in the tree, those methods just using strategies to select valuable nodes directly for further propagation, through which without using distance metric. Another attempt is to adaptively reduce metric sensitivity[11]. Each node (state) in the tree is associated with a function called constraint violation frequency (CVF) and each time the node or its descendant selected as the candidate nearest node for expansion, the CVF of this node is updated. The probability of selecting the nearest node is according not only the distance metric but also the CVF.

Our paper presents a greedy kinodynamic motion planner with a similar lead, but pre-computed by geometric RRTs instead of cell decomposition. Geometric RRTs could obtain a lead more efficiently than cell decomposition in complex geometric environments. Through the guide of the leads, the states near the geometric trees enjoy more preference for choosing expansion. Also, for searching the nodes near the geometric trees as well as selecting the nearest state for propagation, a new distance metric based on Euclidean metric by considering both the current position and the following position with zero-input was proposed, which is quite effective combined with the greedy planner.

The rest of the paper is organized as follows: Section Ⅱ outlines the kinodynamic motion planning problems. Section Ⅲ present the proposed algorithm in more detail. Section Ⅳ summarizes our set of experiments. And section Ⅴ concludes this paper and discuss future directions of this research.

II. Problem Definition: Kinodynamic Motion Planning

Let X be the state space which consists of all variables that describe the states of the system. The control inputs range is limited to U. And the system is subject to first order kinematic constraints and second order dynamic constraints f.

The input of KMP is:

- Original state: $\mathrm{s} \in \mathrm{X}$, where the robot starts from and target objective state region $\mathrm{g} \in \mathrm{X}$.
- Kinodynamic constraints f and allowed time t during which the planner should find solutions, else terminate.

The output of KMP is:

- A sequence of states parameterized with time and its corresponding sequence of control inputs.

Instead of finding a geometric feasible path, kinodynamic motion planner try to calculate a trajectory submit to both geometric and dynamic constraints (e.g., gravity, friction, limits in force). There are multiple directions of researching this problem. Decoupled approaches divide the problem into two independent parts: path planning and trajectory planning, through which to divide the big problem into modules that are each easier to solve. The main drawbacks of those methods are that separated planning lead to impractical paths in the former step. At any step along the way, completeness may be lost because of difficulty for earlier choices to consider problems that may arise later. This paper continues research in tree-based approaches, which is a direction that has produced promising results for planning under kinodynamic constraints. There three aspects should be considered when using a tree-based search strategy.

A. Which direction to be expanded

Direction towards unexplored region and goal region should be preferential. Tree based methods enjoy the inherent property that grow towards unexplored area because of implicit Voronoi bias. The lower density of nodes, the bigger Voronoi areas are. So there are higher probabilities of sampling nodes from this region. However, uniformly sampling distribution will inevitably spend needless time on exploring the state space. Other explicitly way includes EST and single-query bidirectional PRM planner with lazy collision checking (SBL), which attempt to detect less explored regions and expand from them. Another direction is towards the goal regions. Lack of exploitation generates many fruitless nodes while excessive exploitation leads to trees be trapped in local minimal. In [12], balancing the exploitation and exploration under kinematic constraints are researched. For kinodynamic planners, propagating a new node is much more time consuming than geometric planners. Thus, choosing a direction explicitly and guiding the tree to explore towards the goal regions is an effective way of improving efficiency, through which to reduce the needless nodes propagation.

B. Which node to be selected

Basically, this problem was aroused by the sensitive of distance metric, and there are two factors result in the matter: the differential constraints and obstacles effect [11]. To guarantee the valuable expansion, valuable nodes should be chose. KD-tree is widely used for searching the nearest neighbors, but with limitation in differential situation. Euclidean metric could hardly reflect the real cost between two states. Even if an exact metric does not exist or cannot be computed efficiently, LaValle and Kuffner have argued that approximations will still dramatically improve performance. Quadratic Regulator-based metric functions (QR) are used for reflecting the cost-to-go. Those literatures on how to analytically or numerically solve for the optimal path but the optimal paths can only be computed for a small class of systems. Another research direction is that, since no effective metric for kinodynamic constraints, instead of finding an optimal cost-to-go function, some strategies focuses on alleviating the sensitivity of distance metric, or even without using it (EST, KPIECE).

C. How to expand the selected node

Nodes propagation uses the integration of motion equations or directly invokes Open Dynamics Engine (ODE). There two parameters need to be specified in this step: the control variable u and propagation time Δt. The commonest strategy for selecting control is to pick limited $u \in \mathrm{U}$, which is generally uniformly random selected or from a discretization, within the control input range, then stimulate the model forward in time Δt until a maximum number of steps is exceeded, a collision occurs or a state-constraint is violated. The selection of propagation time depends on the environment (robots and obstacles). In [13], complementarity-based dynamic simulation are proposed, which is relatively insensitive to the choice of Δt and the discretization resolution of the input set.

III. The Planner: Guided RRT

To put it simply, Guided RRT is a combination of geometric trees and kinodynamic tree. In practice, tree-based algorithms perform very well in configuration space planning. The geometric trees, which only obey to geometric constraints, could cover the configuration space with dramatically speed in complex environment. Through continuous updating of a weight function, Guided RRT use geometric trees as waypoints

to guide the propagation direction of kinodynamic tree. Also, since the nodes near the waypoints enjoy more preference, distance metric for nearest neighbors searching are needed not only for choosing the nearest node for expansion, but also for setting the weight function. We put forward a new distance metric considering zero-input prediction, which is simple but quite much more effective than the Euclidean distance metric.

A. Tree Expansion: Greed with Completenesss

GRRT starts by using several geometric trees to explore the environment. Without using forward dynamics, the function of those geometric trees is to find geometric feasible path. First, the "RandomConfiguration" function select a configuration by random. The nearest configuration in the tree is chose by using Euclidean distance metric. "NewConfiguration" function extends the selected configuration towards the random place, which generally makes progress toward the unexplored area, by a constant distance. If there are no collision with any obstacle in the environment would occur by moving to that new configuration, then the new configuration is added into the geometric tree, until an added configuration reach to the goal region, a geometric feasible path are found and stored. In limited time or limited maximum nodes, or after pre-set n paths found, the "GuidedTrees" function terminated with the found paths returned.

Algorithm 1: Geometric Trees

```
for k=1 to K do
    Xrand ←RandomConfiguration();
    Xnear ←NearestNeighbor(Xrand,Tree);
    Xnew  ←NewConfiguration(Xnear,Xrand);
    if ObstacleFree(Xnear,Xnew)
          Tree.addVertex(Xnew);
          Tree.addEdge(Xnew);
    if Xnew ←GoalRegion
          GeometricTree←BackTrace(Xnew,Xstart);
end for
Return GeometricTrees
```

The procedure to find a kinodynamic solution is the almost same as the procedure to find geometric trees. The pseudo code below outlines the algorithm. At each iteration, a state is generated by "SelectState", which have a certain probability p to return the nodes in pre-computed trees as waypoints and the left 1-p probability to return random state. Among the p probability, we choose the waypoints by an interactive way between the nodes in geometric trees and the growing kinodynamic tree: the waypoints in geometric trees guide the expansion of kinodynamic tree, the other way round, the distribution of the kinodynamic tree will influence the selection of the waypoints. According to a weight function, a node in the geometric trees is selected as waypoint and returned. The ZIP-distance (In next chapter) between each new added state and the waypoints are computed. If the ZIP-distance is small than *r*, then this waypoint is set as reached. The waypoints following the last reached waypoint have the highest probability to return as next waypoint, with the probability decreased towards farther waypoints. Also, the more reached times, the lower probability to be selected as returned nodes. If a waypoint is chosen too many times, it means the path along the waypoint is hardly to obey to the dynamic constraints, which will be set a lower probability in the geometric trees. All the influence is reflected in the "*weight*", which be updated once any new state added to the tree. It's worth noting that the final returned node as the "sample node" is not the original waypoints itself, but a node in a circular region with radius *r*. The "ZIPDis" function uses the ZIP-distance to compute the nearest neighbors between the kinodynamic trees and selected state. Here, we store a dummy kinodynamic tree, which is the same as the normal kinodynamic tree except the nodes with too many children are deleted. Through this way, the "ZIPDis" function will only find the nearest neighbors with limited children, to avoid too dense of certain region.

Algorithm 2: Guided RRT

```
for k=1 to K do,
   for i=1 to last unreached node in geometric trees do,
   dis ← ZIPDis(KTree.endnode, GeometricTrees(i))
   if dis < r
         reached ← GeometricTrees(i);
         update weight;
    end for
    Xrand ←SelectState(weight);
    Xnear ←ZIPDis(Xrand,Tree);
    Xnew  ←Steer(Xnear,Xnew);
    if ObstacleFree(Xnear,Xnew)
          Tree.addVertex(Xnew);
          Tree.addEdge(Xnew);
    if Xew ←GoalRegion
          Return Reached;
end for
Return Failure;
```

B. Proximity Queries: Zero-Input Prediction(ZIP-Distance)

The guided RRT rely on the distance metric not only in choosing the nearest neighbors for further propagation, but also in selecting the random state. The normal Euclidean distance metric could hardly represent the real cost between two states in differential situation. To handle this issue, a more strait forward way is the Reachability-Guided Sampling (RG-RRT) [14]. It builds a reachable set attached to each node. The distance to the random point between the nearest node in the tree and any point in reachable set are compared. If any point in reachable set is closer than the nearest node, the point in reachable set and its corresponding parent node are returned, else threw away. The approach still uses the basic Euclidean metric but notably eliminate the node in unreachable regions. An improvement of this idea (EG-RRT), using the feedback information from exploration process for passing narrow spaces. One limitation of those methods is that to describe the reachable set is time-consuming and intractable in high dimensional environments. Find a way to describe the reachable set is important for this method.

$$ZIPDis = \phi * dis(current) + \delta * dis(predition) \quad (1)$$

Here, instead of computing the whole reachable set of each node. We simplify this problem by just considering the reachable state in a certain time Δt with zero input. Using forward dynamics, we can predict the next state in a fixed time, which are drove by the inertial. Thus, the next predictive state is determined by the current state: position, velocity, orientation and other inherent property. If the Euclidean distance increased dramatically, it means the current state cost heavily towards the goal state. Conversely, the cost reduced when the distance between predictive state and goal state decreased. Whole distance is influenced both by the current position and the zero-input predictive position as in (1). In Figure 1, Point c is regarded as the nearest neighbor of Point x, rather than Point a and Point b.

Algorithm 3: ZIPDis(Xrand,Tree)

```
min_distance=inf;
for i=1 to size(Tree.nodes) do,
    Predition_position(i)=steer(Tree.nodes(i), zero input);
   dis(i)=φ*dis(Predition_position(i))+δ*dis(Tree.nodes(i));
    if dis (i) < min_distance
        min_distance=dis (i);
        nearest_node_index=i ;
end for
return nearest_node_index
```

The information included in each node in state space is increased: current position and velocity → current position, velocity, predictive position. Also, in order to alleviate the computation work, rather than comparing the ZIP-distance with each state in the kinodynamic tree, we can first find the k-nearest neighbors by using the KD-Tree, which is an effective approach for finding nearest neighbors in Euclidean space. Then comparing the ZIP-distance with each k-nearest neighbors and the minimum one returned. This is left to further research.

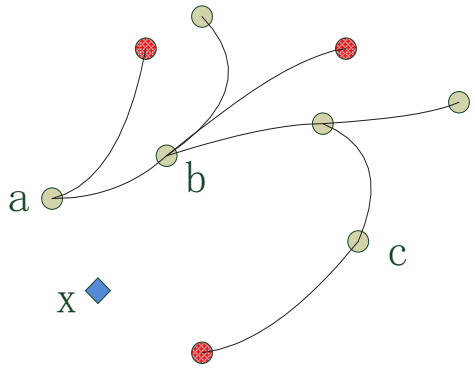

Figure 1. One example of nearest neighbors by ZIP metric. The red point is the propagation with zero input.

IV. EXPERIMENTS

We implemented the Guided-RRT to perform kinodynamic planning for two different systems in simulation. The first application is to solve for collision-free trajectories for a simple Nonholonomic car driving among obstacles in different environments. The second test is to find solutions for a 2-degree manipulator. For comparison, we also applied the standard RRT kinodynamic planner within both simulations. The standard RRT planers use a uniform distribution over the state space to generate samples and the Euclidean metric to identify nodes for expansion. All the simulation was run on a 2.9 GHz Intel i5-3380M CPU with 8 GB RAM.

A. Noholonomic mobile car

A differential drive robot is assumed and its equations are defined as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{V}_L \\ \dot{V}_R \end{bmatrix} = \begin{bmatrix} \cos\theta * \frac{R}{2} * (V_L + V_R) \\ \sin\theta * \frac{R}{2} * (V_L + V_R) \\ \frac{R}{2*L} * (V_R - V_L) \\ \alpha_L \\ \alpha_R \end{bmatrix} \quad (2)$$

Where x and y is horizontal and vertical displacement of the robot, θ is the heading of the robot. Parameters R and L are the radius of the wheel and the distance between a wheel and the robot's center. In a differential drive robot, its motion is directly controlled by the left α_L and right α_R wheel accelerations, with the limitations of maximum allowable wheel velocities, turning velocity, wheel angle acceleration and turning acceleration:

$$|V| \le 3\frac{m}{s}, \left|\dot{\theta}\right| \le 20\frac{\deg}{s}, |\alpha| \le 0.6\frac{m^2}{s}, \left|\ddot{\theta}\right| \le 3\frac{\deg^2}{s} \quad (3)$$

Table I. Computational times in seconds for differential drive

	Basic-RRT	Guided RRT(Euclidean metric)	Guided RRT(ZIP metric)
Easy	22.03	3.59	2.36
Medium	43.09	6.40	3.89
Difficult	709.18	24.12	14.12

Three levels of difficulty are considered (see Figure 2). Both algorithms (standard RRT and Guided RRT) shared much of the same code base. Figure 2 compares examples of trees resulting from both algorithms. Each was run a total of 50 times, with the same start and goal pose and obstacle configuration as shown in figure. Table Ⅰ gives computational time for Guided RRT with zip metric, Euclidean metric separately, and the standard methods it is compared to. On average of 50 times, in the easy scenario, the Guided-RRT

using ZIP distance metric find a solution in 2.36 seconds, a mean of 218 nodes in the kinodynamic tree and a mean of 241 integrations. The large number of integrations resulted from a substantial proportion of propagation attempts that failed due to collision. Six integrations were performed, with half of them were zero-input integrations for each propagation. On the other hand, the standard Kinodynamic RRT required 22.03 seconds on average, with a mean of 2953 nodes in the kinodynamic tree and 3624 integrations required. The superiority is much more evident under difficult scenario, which is about 50 times faster than the standard RRT. Another remark is that ZIP metric outperforms the Euclidean metric in all environments combined with GRRT.

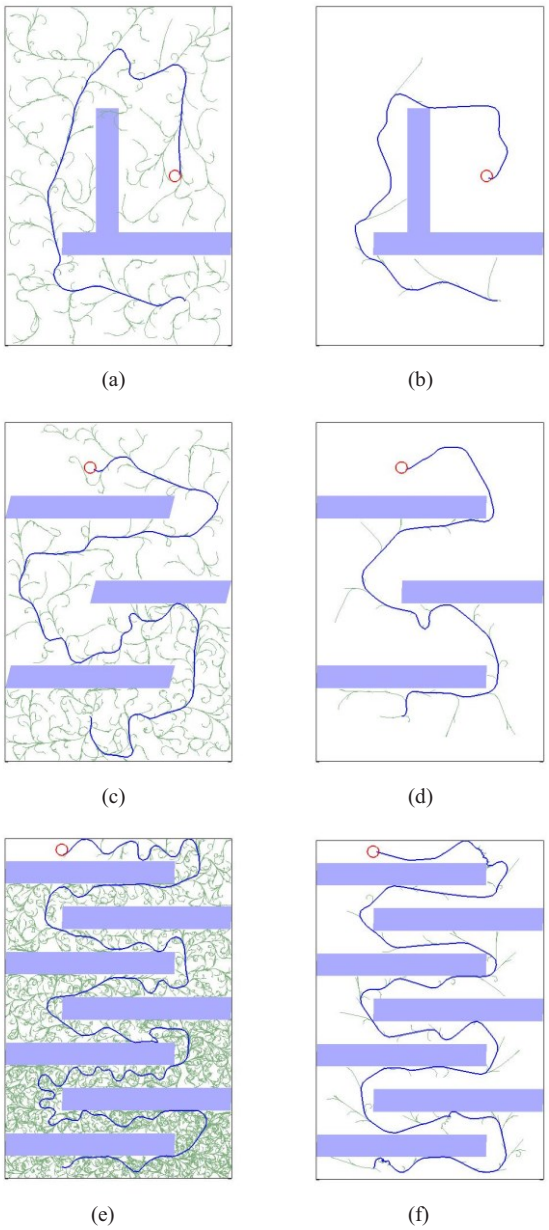

Figure 2. Green lines represent the branches of the tree and Blue lines represent solutions. (a) (c) (e) returned from standard RRT and (b) (d) (f) from GRRT.

B. Two Degree Manipulator

The second example, a two degree manipulator dynamic model, is derived from Lagrange's equation defined as:

$$\begin{bmatrix} \alpha+2\beta c_2 & \delta+\beta c_2 \\ \delta+\beta c_2 & \delta \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} -\beta s_2 \dot{\theta}_2 & -\beta s_2(\dot{\theta}_1+\dot{\theta}_2) \\ \beta s_2 \dot{\theta}_1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \quad (4)$$

Where

$$\alpha = I_{z1} + I_{z2} + m_1 r_1^2 + m_2(l_1^2 + r_2^2)$$
$$\beta = m_2 l_1 r_2$$
$$\delta = I_{z2} + m_2 r_2^2$$

The first term in this equation represent the inertial forces due to acceleration of the joints, the second represents the Coriolis and centrifugal forces, and the right-hand side is the applied torques. The task is to move the manipulator from an original configuration to a goal configuration in a gravity free environment with two circular obstacles (Figure 2). The system parameters were set to a mass of m1=1.25Kg, m2=0.8Kg, I_{z1} =0.066kg·m², I_{z2} =0.043kg·m², a maximum torque of $|\tau| \le 1.5$, the distance of center of gravity from joint frame $r_1 = r_2 = \frac{l_1}{2} = \frac{l_2}{2}$ =0.2, and the step size for each propagation are set to 0.04.

Figure 2 shows one example of the trees generated by the Guided RRT with different goals. The kinodynamic tree is attracted by the waypoints, which constitute geometric feasible paths. Therefore, rather than exploring the whole environment uniformly, the Guided RRT are much more greedy than the standard RRT in growing toward the goal. The best combination of parameters that we were able to find: $P(waypoint) = 0.7$ (for the sampling probability), $\phi = 1$, $\delta = 0.8$ (for the ZIP-Distance). Table Ⅱ compares the performance of different trees. Even though Guided RRT spent more time at each node propagation, it reduced the nodes number dramatically and speeds up of 5 to 20 times in such environment with different goals. It's worth noting that the trajectory execution time and the physical performance are not considered. And the effect of the ZIP metric here is much more remarkable than the mobile car scenario.

Table II. Computational times in seconds for Manipulator

	Standard-RRT	Guided RRT(Euclidean metric)	Guided RRT(ZIP metric)
GOAL Ⅰ	16.5	22.17	2.92
GOAL Ⅱ	174.65	21.02	8.70
GOAL Ⅲ	78.52	27.89	13.84

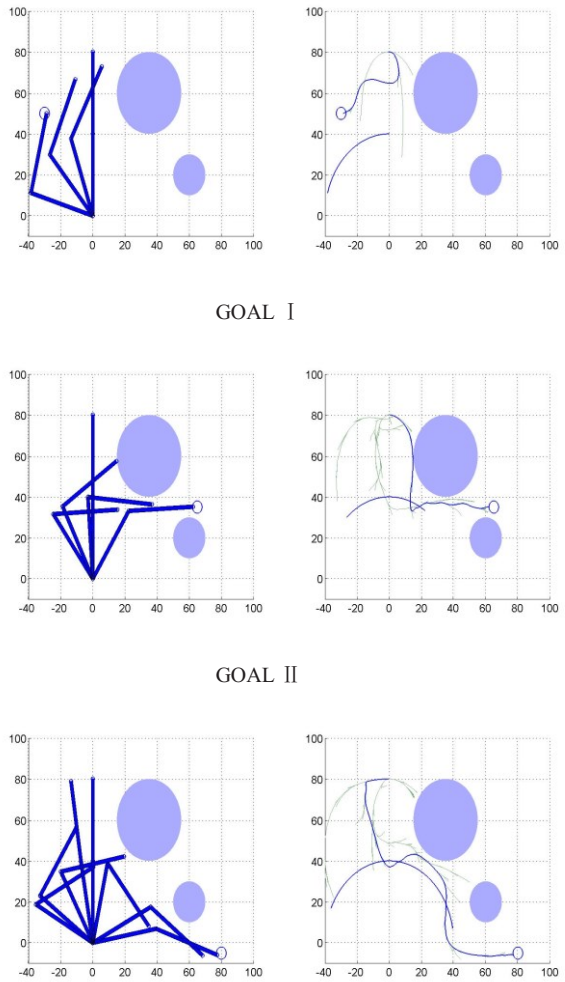

Figure 3. Left parts represent several middle configuration from start to goal configuration. Right parts show the trees. Green lines represent the branches of the tree and Blue lines represent solutions.

V. Discussion and Future Work

This paper presented a Guided RRT path planner that significantly reduces the solution times under kinodynamic constraints. One limitation for sampling-base algorithms is that most of the nodes expansion is fruitless. Nevertheless, it's prerequisite due to the unknown environment. The Guided RRT explores the environment using several geometric trees firstly, whose computation time is negligible compared to the expansion of kinodynamic tree. And with the guidance of those geometric feasible paths, the Guided RRT expands towards the goal greedier than the standard RRT. Also, the new algorithm relies on distance metric not only in choosing the nearest node for propagation, but also in selecting the sampling nodes. The normal Euclidean metric could hardly represent the cost in differential environments. Thus, an improved distance metric was put forwarded for this new approach. By considering the next state with zero-input of each nodes, the distance function are influenced not only by the current position, but also by the velocity and orientation, which could better reflect the cost between different states.

There are various directions for extending the current work. Firstly, balancing the exploration and exploitation are always a challenge for greedy planners, which need elaborated parameters. Strategies for alleviating the sensitive to those parameters should be studied. Also, since the geometric trees already cover the environment, reusing it in replanning tasks is possible. And based on the information from the geometric trees, it's also possible to reduce the repetitive collision checking. Moreover, extending the new approach in high-dimensional space deserve further researching.

Acknowledgment

We would like to thank Wouter Wolfslag and Michiel Plooij for their helps related to this work.

References

[1] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on,* vol. 12, pp. 566-580, 1996.

[2] S. M. LaValle, "Rapidly-Exploring Random Trees A Щew Tool for Path Planning," 1998.

[3] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, 1997, pp. 2719-2726.

[4] A. M. Ladd and L. E. Kavraki, "Fast tree-based exploration of state space for robots with dynamics," in *Algorithmic Foundations of Robotics VI*, ed: Springer, 2005, pp. 297-312.

[5] S. M. LaValle, *Planning algorithms*: Cambridge university press, 2006.

[6] F. Lamiraux, E. Ferré, and E. Vallée, "Kinodynamic motion planning: connecting exploration trees using trajectory optimization methods," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, 2004, pp. 3987-3992.

[7] L. Jaillet, A. Yershova, S. M. La Valle, and T. Siméon, "Adaptive tuning of the sampling domain for dynamic-domain RRTs," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, 2005, pp. 2851-2856.

[8] I. Şucan and L. E. Kavraki, "A sampling-based tree planner for systems with complex dynamics," *Robotics, IEEE Transactions on,* vol. 28, pp. 116-131, 2012.

[9] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Discrete Search Leading Continuous Exploration for Kinodynamic Motion Planning," in *Robotics: Science and Systems*, 2007, pp. 326-333.

[10] I. A. Sucan and L. E. Kavraki, "On the performance of random linear projections for sampling-based motion planning," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 2009, pp. 2434-2439.

[11] P. Cheng and S. M. LaValle, "Reducing metric sensitivity in randomized trajectory design," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, 2001.

[12] M. Rickert, O. Brock, and A. Knoll, "Balancing exploration and exploitation in motion planning," *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on,* pp. 2812-2817, 19-23 May 2008 2008.

[13] N. Chakraborty, S. Akella, and J. Trinkle, "Complementarity-based dynamic simulation for kinodynamic motion planning," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 2009, pp. 787-794.

[14] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 2009, pp. 2859-2865.