Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 11(4): 274-281, Dec. 2013 Regular paper

Kinodynamic Motion Planning with Artificial Wavefront Propagation

Dmitriy Ogay[1] and Eun-Gyung Kim[2*], *Members*, *KIICE*

[1]Department of Computer Science& Engineering, Graduate School, Korea University of Technology and Education, Cheonan 330-708, Korea
[2]School of Computer Science& Engineering, Korea University of Technology and Education, Cheonan 330-708, Korea

Abstract

In this study, we consider the challenges in motion planning for automated driving systems. Most of the existing online motion-planning algorithms, which take dynamics into account, find it difficult to operate in an environment with narrow passages. Some of the existing algorithms overcome this by offline preprocessing if environment is known. In this work an online algorithm for motion planning with dynamics in an unknown cluttered environment with narrow passages is presented. It utilizes an idea of hybrid planning with sampling- and discretization-based motion planners, which run simultaneously in a full configuration space and a derived reduced space. The proposed algorithm has been implemented and tested with a real autonomous vehicle. It provides significant improvements in computational time performance over basic planning algorithms and allows the generation of smoother paths than those generated by the recently developed hybrid motion planners.

Index Terms: Autonomous vehicle, Differential constraints, Kinodynamic planning, Motion planning

I. INTRODUCTION

Recently, there has been considerable research in the area of motion planning, which is an essential part of almost all robotic systems. Its area of application goes beyond robotics into the areas of computational biology, computer animation, and economics.

Further, in the past few years, the development of automated driving systems has been a source of many challenges for motion planning. Many approaches to solving the problem have been developed and successfully implemented.

The initial motivation for this work was to develop a motion planning algorithm that could drive an autonomous vehicle in an unknown cluttered environment under differential constraints. The development was a part of preparations for Hyundai-Kia Autonomous Vehicle Competition 2012, held in Korea.

Among the recently developed motion planning methods for autonomous vehicles, several types of planning approaches may be distinguished. The most widely used ones are space discretization-based and sampling-based methods. The former methods work well in cluttered environments but have trouble in generating smooth trajectories and taking kinodynamic constraints into account, and vice versa.

The proposed method is based on the idea of simultaneously running two different types of motion planning algorithms, which interact with each other and benefit from each other's advantages. This approach was studied in [1, 2] and applied to offline motion planning with dynamics in a

Received 06 September 2013, **Revised** 20 October 2013, **Accepted** 08 November 2013
***Corresponding Author** Eun-Gyung Kim (E-mail:egkim@koreatech.ac.kr, Tel: +82-41-560-1350)
School of Computer Science & Engineering, Korea University of Technology and Education, 1600 Chungjeol-ro, Byeongcheon-myeon, Dongnam-gu, Cheonan 330-708, Korea.

Open Access http://dx.doi.org/10.6109/jicce.2013.11.4.274 print ISSN: 2234-8255 online ISSN: 2234-8883

This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/li-censes/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Copyright ⓒ The Korea Institute of Information and Communication Engineering

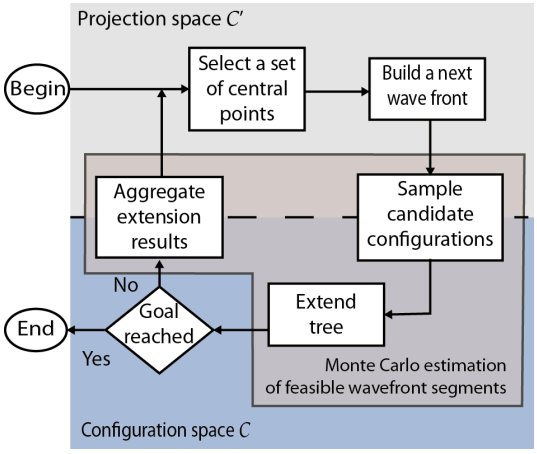

Fig. 1. Block diagram of the proposed planning algorithm.

cluttered environment. In these works, a graph search algorithm runs in a discretized workspace and guides the extension of a tree in a continuous configuration space. The progress of tree extension is then estimated and is utilized to update the extension heuristics of the higher-level path planner.

In our work, we have applied a similar strategy. However, instead of running a discretization-based algorithm in a projection space, we introduced a simple method that propagates an artificial wavefront, which helped us to resolve some issues with the resolution of a projection space discretization. It allowed us to run the planning algorithm in real time. Simultaneously, we noticed that the generated wavefront may be utilized to construct a potential field with no local minima, which converges in the initial point. This field may later be utilized to produce a navigation function [3]. However, navigation function for this particular problem is not studied in this paper.

Navigation functions open more opportunities for the feedback motion planning, where a separate pathfollowing controller is not required. Of greater significance is the fact that in conjunction with reactive planning algorithms, these functions allow the creation of planners that can cope with unpredicted dynamic obstacles in unknown environments.

Works on Euclidean shortest paths include [4, 5]. Unlike methods based on interpolation, the proposed wavefront propagation method is based on random sampling, performs a feasibility check not only on the obstacle presence but also on the kinodynamic constraints of the robot, and easily integrates with a sampling-based motion planner.

II. RELATED WORKS

One of the common approaches in the area of motion planning is to discretize the configuration space and then run a graph-search algorithm like A* or Dijkstra's algorithm. This approach was applied to solve the problem of autonomous vehicle driving in [6, 7].

Some other works (rapidly exploring random tree [RRT] and probabilistic roadmap [PRM]) [8, 9] introduce approaches for motion planning in continuous space with kinodynamic constraints. Unlike methods with space discretization, these methods suffer less from the so-called curse of dimensionality, when the algorithm complexity increases exponentially with the number of dimensions. Further, these methods can generate smooth trajectories, which do not require additional smoothing or optimization [6]. However, depending on the environment (presence of narrow paths), these methods may run for an indefinitely long time. To overcome these issues, many strategies for sampling biasing have been introduced [10, 11].

III. METHOD DESCRIPTION

The proposed method combines a sampling-based motion planning method and an artificial wavefront propagation method (Fig. 1). The sampling-based algorithm is run in a full configuration space C, and an artificial wavefront propagation method is run in the projection space $C' = \phi(C)$, where $\phi()$ denotes a transform, which may be identity. We will refer to our method as overlapping disks expansion (ODEx), which reflects how the wavefront is propagated.

A sampling-based planner is a type of rapidly exploring dense tree (RDT) algorithm. It manages the kinodynamic constraints check and the obstacle collision checks by means of a simulation. The sampling is biased by an artificial wavefront, which propagates in a projection space. After the tree is extended to the configurations sampled at the current wavefront, the progress of the propagation is analyzed, and the center points for the construction of the next wavefront are determined.

A. Wavefront

We consider the wavefront as a curve or a set of curves in a two-dimensional space or a surface or set of surfaces in a higher-dimensional space.

To describe the wavefront construction, let us introduce two variables λ, which serves as an analog of wavelength, and $k \in \mathrm{N}$, which denotes the ordinal number of a wavefront. Let the wavefront originate from a center point $o_{0,0}$ and propagate in all directions at equal speed. It will form a circle c_0 with radius λ, enclosing an open set d' (Fig. 1).

We then utilize an idea from Huygens principle that every point where the wave has reached becomes a source of secondary waves. However, following the original Huygens principle would imply an infinite number of points. There-

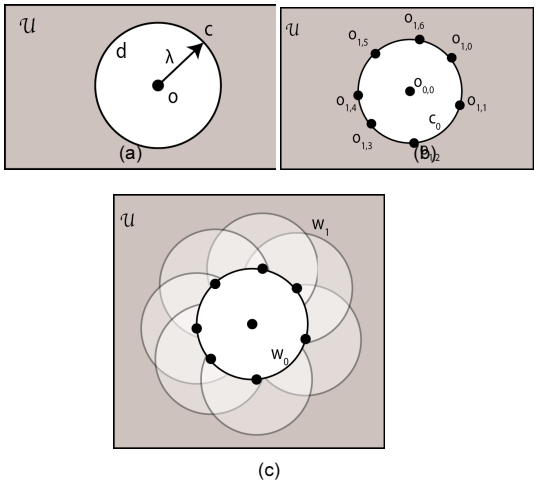

Fig. 2. Wavefront is propagated with overlapping disk expansion. (a) The first propagation. (b) Center points for the next propagation are selected. (c) Overlapping disks are constructed.

fore, in our case, we limit the number of points. Let us now sample several points $o_{i,k} \in 0_k \in c_0$. See Fig. 2(b).

Then, we construct overlapping circles $c_{i,k}$ with the centers in $o_{i,k}$. The circle segments, which are not overlapped by any open disks $d_{i,k}$, form a new wave front w_k. See Fig. 2(c).

At the next step, new center points o_{k+1} are sampled from w_k, and the process iterates.

$$w_k = \bigcup(p \in c_{i,j}) \setminus \bigcup(d_{i,j}) \forall i,j < k\,. \qquad (1)$$

Such a definition of a wavefront gives us several advantages:

- Even if just one center point is sampled for the construction of the next wavefront, the wavefront may still be constructed.
- Normals to the wavefront point to the corresponding center points and thus to the previous wavefront, converging to the point of the origin.
- Points on the wavefront may be sampled with a simple algorithm, described below, even for high-dimensional spaces.

B. Wavefront Sampling Algorithm

One of the issues that arise when working with curves or surfaces is how to represent them. A helpful feature of the proposed wavefront propagation method is that a wavefront is defined only by a set of center points O. Since in the proposed method, we utilize wavefronts to sample candidates for the sampling-based motion planner. Further, a sufficient representation implies an ability to sample candidate states on the wavefront.

Algorithm 1 Wavefront sampling

```
1: function SAMPLEWAVEFRONTPOINT(O, λ)
2:     repeat
3:         Randomly select a center point o from O;
4:         Sample random state p at distance λ from o;
5:         o_nearest ← nearest(p, O);
6:     until o_nearest ≠ o
7:     return p
8: end function
```

The sampling function is shown in Algorithm 1. First, it samples random points on all circles and then, filters those, which do not overlap any of the open disks. The function $nearest(p, O)$ is a nearest neighbor search, where O denotes an input set to search in, and p represents the point to search the nearest neighbor for. It may be efficiently implemented with k-d trees.

C. Dense Sampling

For the completeness of the sampling-based motion planner, it is important for the sampled sequence to be dense. Let us consider an approach that makes the proposed artificial wavefront a dense sequence.

There are two parameters that affect wavefront construction: λ, which denotes the radius of the elementary circles and k, which represents the ordinal number of the wavefront. Let us consider a sequence $z = 0,1,2.. \in (N)$ and then, construct a sequence of wavefronts such that $\lambda_7 = 2^{-z}\lambda,\ z \to \infty$ and $k \to \infty$.

Theorem 1. *A sequence of wavefronts $w_0, w_1, \ldots w_k$ is dense in the area, covered by the set of all disks $d \in D$.*

Proof. The distance from any point covered by a disk d to the corresponding center point $o$is less than λ. We may choose a sufficiently large z so that $2^{-z}\lambda < \varepsilon$, where ε denotes an arbitrarily small value.

Theorem 2. *As $k \to \infty$, D overlaps any connected subset $A \in C'$ of a configuration space.*

Proof. As long as w_k exists, there is a non-zero probability that any point $p \in w_k$ is selected as a center point o. The new disk d is then may be constructed around o and overlap an additional subset of $B \in A$. Thus, the process of expansion does not stop unless w_k exists.

As the wavefront is the boundary of D, the growth will continue unless A is overlapped by D.

Theorem 3. *As $z \to \infty$ and $k \to \infty$, the sequence of wavefronts becomes dense in configuration space C.*

Proof. As it is seen from the previous theorem, D may cover an arbitrarily large $A \in C$ and the sequence of wavefronts is dense in D.

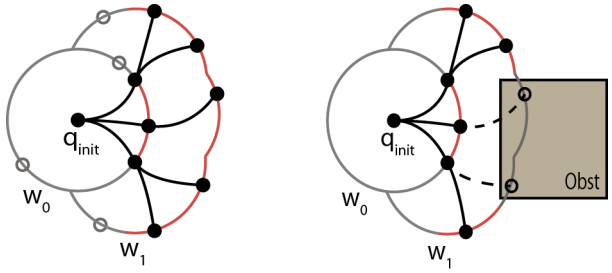

Fig. 3. (a) New candidate configurations are sampled at the wave-front; the planning tree is then extended to the sampled candidates (denoted with filled dots). (b) Configurations that are not feasible because of the presence of obstacles or differential constraints are ignored (denoted with unfilled dots).

D. Dense Sampling

In the proposed method, a sampling-based planner runs synchronously in a full configuration space C together with the wavefront propagating planner, running in a projection space C'. Further, it serves several goals:

- generates a plan for execution;
- ensures the feasibility of paths by means of constraints and obstacle collisions checks; and
- provides the means to simulate the propagation of the system to the next wavefront.

The sampling motion planner is based on a RDT (also known as RRT) [2]. An important feature of the RRT planner is its probabilistic completeness in the case of a dense sampling sequence. It uses a tree to represent a set of generated paths, where nodes correspond to configurations, and edges correspond to the feasible paths between configurations. The tree is iteratively extended towards sampled configurations unless the target or a certain limiting condition is met. In the proposed method, candidate configurations are sampled on the wavefronts, as shown in Fig. 3.

Algorithm 2 Extension of tree to the next wavefront

```
1:  function EXTENDTREE(G, O_k, K, λ_k)
2:      W ← ∅
3:      for i = 0 → K do
4:          p_rand ← sampleWaveFrontPoint(O_k, λ_k);
5:          q_rand ← sampleRandom(φ^{-1}(p_rand));
6:          q_nearest ← nearest(q_rand);
7:          q_new ← steer(q_nearest, q_rand);
8:          if q_new ≠ ∅ then
9:              G.addVertex(q_new);
10:             G.addEdge(q_nearest, q_new);
11:             W.add(q_new);
12:         end if
13:     end for
14:     return W
15: end function
```

Let the tree be denoted as $G(V, E)$. We initialize it with a starting configuration q_{init}. Then, extend the tree to the nodes, sampled at wavefronts, as shown as Fig. 3(a). To be more precise, candidate configurations q_{rand} are sampled in the configuration space C so that $\phi(q_{rand}) \in w$, where ϕ denotes the transform from the configuration space to the projection space and w represents the current wavefront.

The tree extension procedure is shown in Algorithm 2. It is similar to that of RRTs, except the random sampling part. First, a point p_{rand} is sampled from a wavefront $w \in C'$, and then, a candidate configuration is randomly sampled from the preimage of p_{rand}. Moreover, *sampleRandom()* generates random samples. It should be dense in $\phi^{-1}()$, in order for sampled configurations to be dense in C. Dense sampling in C is necessary to achieve the probabilistic completeness.

After a candidate is sampled, the nearest configuration in the tree is selected. Then, the algorithm attempts to connect the closest configuration to the candidate configuration with a valid collision-free trajectory by means of simulation. If kinodynamic constraints are maintained (Fig. 3(a)) and the paths are obstacle free (Fig. 3(b)), then the candidate state and the linking path are added to the tree.

Propagation is performed in stages, wavefront by wavefront, and at every stage, some limited number of iterations is performed. For the proposed system, we determined this parameter experimentally by choosing between the smoothness of the path and the available computational resources.

After each stage, configurations that were successfully added may be considered as the representatives of the feasible segments of the current wavefront. This set of configurations is then analyzed to determine the shape of the next wavefront.

E. Stochastic Analysis of Expansion Progress

When new candidate points are sampled at a new wavefront, which parts of the new wave front are feasible and which ones are not is not known. The complete solution to determine, which wavefront segments are feasible, may

Algorithm 3 Aggregation

```
1:  function ESTIMATECENTERPOINTS(W)
2:      CP ← ∅;
3:      CL ← densityClustering(W);
4:      for all cl ∈ CL do
5:          SBCL ← SBCL ∪ split(cl);
6:      end for
7:      for all sbcl ∈ SBCL do
8:          CP ← CP ∪ φ(mean(sbcl));
9:      end for
10:     return CP
11: end function
```

Algorithm 4 Planning with wavefront propagation

1: $G(V,E) \leftarrow (q_{init}, \emptyset)$;
2: **for** $k = 0 \rightarrow K$ **do**
3: $\quad \lambda_k \leftarrow 2^{-k}\lambda_0$;
4: $\quad O_k \leftarrow \phi(q_{init})$;
5: $\quad$ **for** $j = 0 \rightarrow J$ **do**
6: $\quad\quad W \leftarrow extendTree(G, O_k, N)$;
7: $\quad\quad O_k \leftarrow O_k \cup estimateCenterPoints(W)$;
8: $\quad$ **end for**
9: **end for**

not exist because obstacles in the considered space may have various shapes. From the other side, the tree G, of the sampling-based planner, extends only to feasible configureations. Therefore, a tree extension may be considered a tool for the stochastic estimation of the feasible segments of a wavefront.

From there, we can pick the center points $o_{k,i}$ to construct the next wavefront more efficiently.

Let us propose some variable $\rho \in \mathrm{R}$ and a mapping $w \in \varepsilon(\rho)$, which maps ρ onto a wavefront. Then, we may introduce a probability distribution function $p(\rho)$, which would determine the probability that a given wavefront point is feasible, as shown in Fig. 4. If we could estimate $p(\rho)$, then we would know which parts of a wavefront are better for a further extension.

When we apply a tree extension, we may notice that $p(\rho)$ may have several peaks and valleys between them. This happens because obstacles make a wavefront discontinuous. Let us consider a set of functions $g_i(\rho)$, with single high peaks; therefore, $p(\rho) = \sum a_i g_i(\rho)$. Then, we may use $E(g_i)$ as center points o_i for the next wavefront expansion.

In order to determine $g(\rho)$, we split a set of propagated configurations into clusters. We utilize density clustering to determine continuous segments, as shown in Fig. 4(a) and (b). Then, we split large clusters into smaller ones, and hence, $g(\rho)$ has a narrow peak. Then, we calculate the mean of each cluster and use it as the expectation of $g(\rho)$. Finally, we use it as a center point $o_{i,k}$ to construct the next wavefront.

Algorithm 3 shows how propagated configurations are aggregated. The function *densityClustering()* splits the input set into the clusters of the nearest neighbors. The function *split()* creates subclusters in such a way that the distance between the furthest points in each cluster is less than 2λ. Any suitable clustering algorithms may be utilized for these purposes.

Algorithm 4 shows the main routine of the proposed method. The algorithm works by consecutively propagating a wavefront. It gradually increases the resolution to achieve the sampling density and thus the planner completeness. It should be noted that the tree $G(V,E)$ remains the same as the resolution increases. The parameters K, which denotes

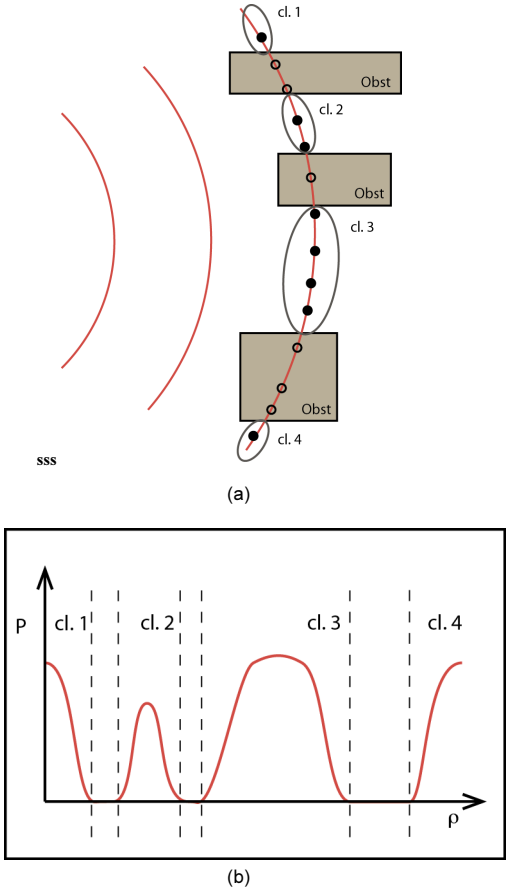

Fig. 4. Successfully propagated configurations form clusters cl. 1, cl. 2, cl. 3, and cl. 4. (a) Successfully extended configurations are gathered in clusters. (b) Probability to reach a point at a wavefront.

the resolution limit, J, which represents the number of wavefronts, and N, which refers to the number of candidate samples per propagation of a wavefront, are determined depending on a particular task and the computational resources.

IV. EXPERIMENTS

The initial motivation for the development of a motion planner was participation in a national autonomous vehicles competition held in Korea. We implemented a multi-threaded version of our algorithm and integrated it into an autonomous vehicle. We then performed several tests in a simulated environment in order to compare our method to other related motion planning algorithms, and to investigate the wavefront behavior in a variety of environments. We also tested our motion planner on an autonomous vehicle, driving it through obstacle patterns difficult for a sampling-based planner.

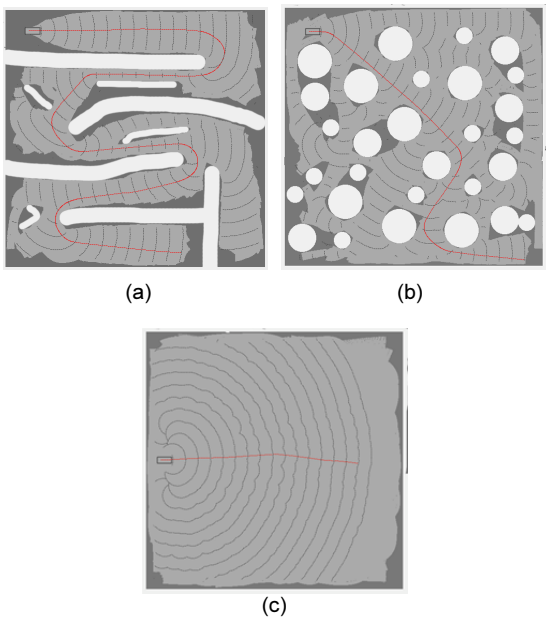

Fig. 5. Simulation was performed in three different environments: (a) labyrinth, (b) randomly scattered round obstacles, and (c) free space.

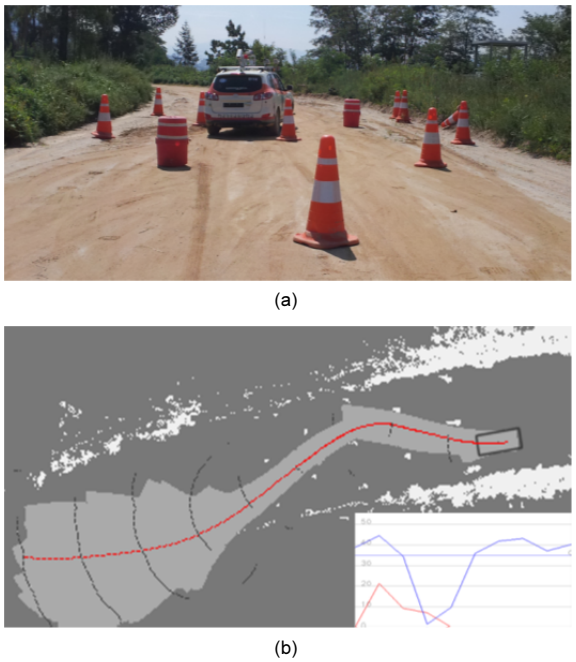

Fig. 6. Experiments with a real autonomous vehicle. (a) Closely placed road cones. (b) Snapshot of running motion planner. Velocity profile (red) and curvature of the path (blue) is depicted in the right bottom corner.

A. Simulation

To estimate the performance of the developed method, we ran a simulation in different types of environments: in a narrow labyrinth, with scattered obstacles, and with no obstacles (Fig. 5). We ran each algorithm ten times on a laptop with quad-core CPU and measured the computational time. Algorithms were made to run in parallel, with shared memory.

We compared our approach to a modified RRT algorithm, because it was already tested on a real autonomous vehicle, and to an algorithm based on Kinodynamic Planning by Interior-Exterior Cell Exploration (KPIECE) algorithm, because our approach is similar to the idea of combining two different planning algorithms.

To generate a sufficiently smooth path, we applied an approach, which was utilized in RRT* [12] and included cost-to-go into the distance for the nearest-neighbor search. Optimization was performed on time and lateral and longitudinal accelerations. Hence, we did not compare our results to the original KPIECE [2] or SyCloP [1] planners, as these planners utilize a forward dynamics simulation, lack the nearest-neighbor search, and have limited capability of increasing the optimality. Nearoptimal paths are important for driving at relatively high speeds. Although we do not claim optimality convergence in the proposed method, the main goal was to create trajectories that were sufficiently smooth for high-speed driving.

We implemented a nearest-neighbor search-based algorithm and applied a sampling strategy similar to KPIECE. We ran a discretization-based planner in the projection space and biased the sampling to the boundary region between the explored and the unexplored zones. For the experiments, we labeled this method as nn-KPIECE.

The results are presented in Table 1: average, minimum, and maximum algorithm run times were recorded. From these results, it can be concluded that ODEx performs considerably better on narrow paths, approximately similar to RRT in an environment with scattered obstacles, and RRT performs better in an open environment. The other advantage of ODEx is that the computational time is considerably less random in comparison to RRT. In comparison to nn-KPIECE, our algorithm performed slightly

Table 1. Comparison of run time in different environments

Map	RRT (ms)	nn-KPIECE (ms)	ODEx (ms)
Labyrinth			
Avg.	22825	1355	771
Min–max	7000–37000	978–1834	427–1118
Rounds			
Avg.	1519	1118	508
Min–max	710–2751	724–3581	316–1159
No obstacles			
Avg.	119	263	227
Min–max	38–185	213–295	197–257

RRT: rapidly exploring random tree, nn-KPIECE: nearest-neighbor Kinodynamic Planning by Interior-Exterior Cell Exploration, ODEx: overlapping disks expansion.

faster but in the same order. The performances of KPIECE and SyCLOP are influenced by the cell size in discretization-based high-level planners. This is one of the disadvantages of these methods, and probably, the cell size was not optimal during the experiments.

B. Real Environment Test

We performed various tests in a real environment. One of the most challenging tests for an autonomous vehicle is navigation in narrow passages, which are difficult for sampling-based motion planners. In the test, we placed road cones in a grid pattern; the distance between the cones was approximately 4–6 m in Fig. 6(a). Our planner generated a trajectory in real time from the map updates, which were generated with the readings from laser scanners in Fig. 6(b). The graphs of curvature and speed profiles are also shown. The speed was not high, as we applied the speed-dependent width of a car for the collision checker in order to compensate for the inaccuracy of the motion controller at high speeds. The initial propagation step λ_n was chosen to be 6 m.

On relatively wide paths, we tested the vehicle by driving at speeds of more than 60 km/h. Planning was carried out considering constraints on lateral and longitudinal accelerations, minimum turn radius, and maximum allowed speed. Dynamics with a side slip was not considered. Depending on the difficulty of the obstacles, the planning time varied from 90 to 600 ms.

V. CONCLUSIONS

During preparations for a national autonomous vehicles competition, we developed a new motion planning approach and implemented it in a real system. We tested the system in a real environment at speeds more than 60 km/h.

The proposed planner solves a combination of planning challenges, including online planning, multiple dimensions, presence of narrow paths, kinodynamic constraints, and unknown environment.

The efficiency of the proposed algorithm was shown experimentally by comparing the performance of the proposed algorithm to that of other planning algorithms in a simulated environment.

REFERENCES

[1] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 469-482, 2010.

[2] I. A. Sucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *Algorithmic Foundation of Robotics VIII*. Heidelberg, Germany: Springer, pp. 449-464, 2009.

[3] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501-518, 1992.

[4] D. S. Yershov and S. M. LaValle, "Simplicial Dijkstra and A* algorithms for optimal feedback planning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco: CA, pp. 3862-3867, 2011.

[5] J. A. Sethian, *Level Set Methods and Fast Marching Methods*, 2nd ed. New York, NY: Cambridge University Press, 1999.

[6] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," in *Proceedings of the 1st International Symposium on Search Techniques in Artificial Intelligence and Robotic*s, Chicago: IL, 2008.

[7] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments," *Journal of Field Robotics*, vol. 25, no. 11-12, pp. 939-960, 2008.

[8] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378-400, 2001.

[9] L. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566-580, 1996.

[10] S. Dalibard and J. P. Laumond, "Control of probabilistic diffusion in motion planning," in *Algorithmic Foundation of Robotics VIII*. Heidelberg, Germany: Springer, pp. 467-481, 2009.

[11] S. Rodriguez, X. Tang, J. M. Lien, and N. M. Amato, "An obstacle-based rapidly-exploring random tree," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Orlando: FL, pp. 895-900, 2006.

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

Dmitriy Ogay
received his bachelor's degree in July 2004 and his master's degree in February 2008 from the Computer Science Department of National Aviation University, Kiev, Ukraine. In September 2009, he enrolled with the Department of Computer Science & Engineering, at the Graduate School of KOREATECH, where he is currently pursuing his Ph.D. degree. His research interests include motion planning, machine learning, and parallel and distributed programming.

Eun-Gyung Kim
received her bachelor's degree in February 1983 from the Physics Department of Sookmyung Women's University, her master's degree in February 1987 from the Computer Science Department of Graduate School of Chung-Ang University, and her Ph.D. in February 1991 from the Computer Engineering Department of Graduate School of Chung-Ang University. Since March 1992, she has been working with the School of Computer Science & Engineering at KOREATECH, and is at present, a professor. Her research interests include intelligent agents, smart learning, and TRIZ.