International Journal of Electrical Electronics & Computer Science Engineering
Volume 4, Issue 1 (February, 2017) | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

Finding Diverse Paths for Robot Navigation Using a Fast Random Walk Approach

R. Sarathkumar Kithiyon[1], S .Immanuel Prabaharan[2], A. Samsu Nighar[3], C. David[4], J. Gold Beulah Patturose

[1] B.E. Student, Department of ECE, Chandy College of Engineering, Tuticorin, Tamil Nadu, India
[2, 3, 4]Assistant Professor, Department of ECE, Chandy College of Engineering, Tuticorin, Tamil Nadu, India
[5]Assistant Professor, Department of ECE, JACSI Engineering College, Tuticorin, Tamil Nadu, India
[1]sarathkk611997@gmail.com, [2]imman.nce@gmail.com, [3]nigarjahir@gmail.com, [4]davidengg4@gmail.com, [5]beulahpattu@gmail.com

Abstract: ***Finding a set of diverse paths among dynamic obstacles is an appealing navigation strategy for mobile robots to qualitatively reason about multiple path hypotheses to the goal. We introduce an efficient randomized approach, based on weighted random walks, that finds K diverse paths on the Voronoi diagram of the environment, where each path represents a distinct homotopy class. We show experimentally that our approach is significantly faster at finding paths of higher diversity in distinct homotopy classes than two state-of-the-art methods. Moreover, we prove that our method is probabilistically complete.***

Keywords: Nonholonomic motion planning, motion and path planning, reactive and sensor-based planning.

I. INTRODUCTION

SINGLE-QUERY robot motion planning generates a single solution from start to goal position under a cost function such as shortest path or maximal transversability. In the presence of unmodeled dynamic obstacles a solution may quickly become obsolete, and the path has to be replanned for each change in the environment. Alternatively, one can consider an algorithm that reactively computes and maintains multiple diverse solutions. This approach has benefits in a variety of scenarios: a set of diverse paths that is continuously checked for validity in the presence of unexpected obstacles reduces the number of replanning queries in highly dynamic environments [1], allows for novel human-robot-interfaces in shared autonomy applications [2], [3] and highly efficient, qualitative planning paradigms for social navigation. To ensure robustness of the path set against changes in the environment, paths in the set should be spatially well separated, since similar or nearby paths are more likely to be invalidated together. Also in shared autonomy applications having a set of more diverse paths is obviously a beneficial strategy, since switching between very similar paths yields limited if not questionable usability. Furthermore, when generating a set of paths, a reasonable approach would be to have *K* paths lying in different homotopy classes. A homotopy class is defined by the set of paths with the same start and goal points, in which any two paths can be continuously deformed into one another without intersecting obstacles. Trajectory optimization methods, which are limited to finding local minima, may benefit from a set of homotopically distinct paths ([2], also discussed in [4]).

With the goal of efficiently finding a set of diverse, high quality paths from different homotopy classes for robot navigation, we make the following contributions:

a) We present a fast and easy to implement random walk approach to generate a set of *K* diverse paths belonging to different homotopy classes. Our method was first introduced in [5] as an alternative to solve the *K* shortest paths problem with deterministic graph search algorithms [2], [6], [7].We build the navigation graph from the Voronoi diagram of the environment (see Fig. 1), where each path represents a distinct homotopy class, and perform a randomized graph search based on random walks. To improve planning performance of the approach introduced in [5], we bias the random walk towards not frequently visited subsets of the state space, therefore increasing the probability to generate paths not yet found. In practice this modification of the algorithm leads to considerable performance improvement. Additionally, we prove that our approach is probabilistically complete, i.e., it finds all the paths, and therefore all the homotopy classes, in the navigation graph.

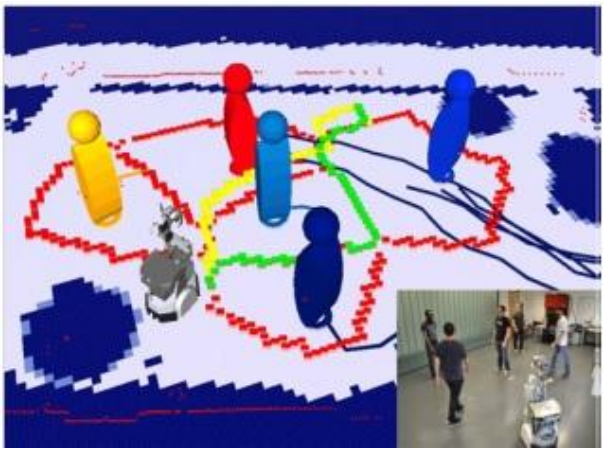

Fig. 1. We test our approach in real-world experiments by applying it to social robot navigation in dynamic environments. The red Voronoi diagram, which implicitly encodes homotopy classes, describes the possible ways to go through a crowd in a room. Our approach rapidly selected two possible paths (in yellow and green).

b) We conduct an extensive evaluation in terms of planning performance and path quality of our approach, comparing it to two baseline methods that generate

International Journal of Electrical Electronics & Computer Science Engineering
Volume 4, Issue 1 (February, 2017) | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

diverse and homotopically-distinct paths [1], [2], and show that our approach is faster and finds more diverse paths (the robot has a more diversified set from where to choose the path to follow), whilst obtaining a negligible loss in path quality. We use the notion of *robust diversity* of a path set to measure mutual special separation of paths. To evaluate the quality of paths in the set, we adopt the *normalized cumulative gain* measure which is used to evaluate web search engine's returned results and ranking quality.The paper is structured as follows: a brief discussion of related works in Section II is followed by the description of our approach and its analysis in Section III. Section IV describes the experiments' settings. We discuss the results in Section V. Section VI concludes the paper.

II. Related Work

Prior research has introduced different methods to generate a set of paths from different homotopy classes. Demeyen and Buro in [8] introduce a method that searches on a graph built using constrained Delaunay triangulations. The obstacles are described via polygonal representation. The paths found in the graph belong to different homotopy classes. In contrast to polygonal representation of the environment, our method works on arbitrary occupancy grid input, which is simpler to handle and nowadays a standard *de facto* to incorporate data from sensors for real-world operation [1] introduce an algorithm that seeks to find a set of diverse, short paths through a roadmap graph. The algorithm searches the graph for shortest paths avoiding a collection of balls - simulated obstacles in the environment. The obtained paths often belong to different homotopy classes.

The authors compare their approach to a *K* shortest paths algorithm and show that, with tolerable loss in shortness, they produce equally diverse path sets more quickly. Compared to Voss, our approach is much faster, it always returns the requested *K* paths if they exist in the navigation graph, the returned paths always belong to different homotopy classes and generally are more diverse. Furthermore, our approach has only one parameter and its runtime does not depend on the density of the roadmap graph. propose a method to find different homotopy classes based on A* search over an augmented graph. The graph encodes topological information via the H-signature, a complex analysis value that characterizes a homotopy class. Kuderer *et al.* in [2] select *K* best homotopy classes by generating *K* shortest paths in a Voronoi-based navigation graph.

During the navigation, the paths feed an optimization algorithm used to generate homotopically distinct trajectories. Among those the best one is selected for the navigation. They show that the method is one order of magnitude faster than Bhattacharya's approach [6]. Moreover, the authors show that the paths in the Voronoi diagram are safer and better suited for social navigation, as they lie as far as possible from the obstacles among all paths in the same homotopy class. Kuderer's approach employs Katoh's algorithm [9] to find the *K* best paths in the Voronoi diagram. However, it was shown by Brander and Sinclair [10] that for small size graphs and paths of small number of vertices, like in the case of Voronoi-based navigation graph we consider, Yen's algorithm [11] is faster than Katoh's. Therefore, we compare our approach to Yen's algorithm and show that our method is faster and returns more diverse paths. Furthermore, in very complex environments with many homotopy classes, deterministic*K* best paths are very similar to each other, therefore losing the advantage of having a set of distinct paths. On the contrary, our approach helps to deliver much higher diversity in such scenarios, providing high quality paths that explore different regions of the map (see Fig. 2).

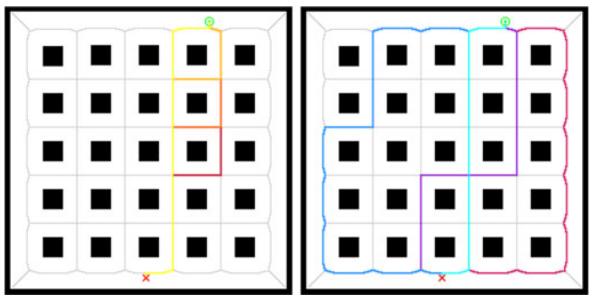

Fig. 2. Comparison of the paths found by our approach and by Kuderer's approach in the "Cubicles" scenario. The red cross represents the robot position, the goal is displayed by the green circle. Left: Kuderer's 4 best paths in the Voronoi diagram. They all traverse the same region of the space. Right: 4 diverse paths found by our approach.

III. A Random Walk Approach to Find Diverse Paths

In this section we detail our approach to find a set of diverse paths lying in distinct homotopy classes.A brief definition of the navigation graph is followed by the description of the random walk procedure. Next, we prove that our technique is probabilistically complete.

A. Navigation Graph:

To frame the path planning task as a graph search problem, we build the navigation graph of the workspace environment from a Voronoi diagram *VD* generated from the sensor data (see Section IV-B for details on the VD generation). The graph *G(V,E)* consists of a set of nodes (or vertices) *V* and a set of edges *E*. Let *N* be the number of nodes and *M* the number of edges. *E*(*vj*) denotes the set of incoming and outgoing edges of the vertex *vj* .We associate to each edge *eij* = (*vj*, *vi*) a weight or cost *cij* (e.g., length of the edge). The adjacency matrix *A* expresses the topology of the graph *G* and is defined as

International Journal of Electrical Electronics & Computer Science Engineering
Volume 4, Issue 1 (February, 2017) | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

$$[A]_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E, i \neq j \\ 0 & \text{otherwise} \end{cases}$$

We compute the set of diverse paths by running our random walk based algorithm on *G*. A walk *w* of length $k - 1$ in *G* is a sequence of nodes *v*1, *v*2, . . . , *vk* , where each pair of nodes is connected by an edge, ($vi-1$, *vi*) $\in$ *E* for $2 < i \leq k$. Henceforward, *walks* are referred to as *paths*.

B. Randomized Homotopy Classes Finder (RHCF):

To find paths belonging to different homotopy classes we introduce the Randomized Homotopy Classes Finder (RHCF), detailed in Algorithm 1. Hereinafter all the steps of the while loop in Algorithm 1 are referred to as *iteration* of RHCF. We iteratively run the Random Walk procedure (see Algorithm 2) on the weighted undirected graph *G* until *K* distinct paths are found and stored in the result set *P*. The walk starts at the initial node *vs* $\in G$, where the robot position is mapped to, and aims to find a random path to the goal node *vg* . At each step of the random walk we choose a random neighbor of the current node *vj* (see RandomNeighbor(*vj*) in Algorithm 2) with probability *pij* inversely proportional to the cost *cij* associated to the edge *eij*

$$P_{ij} = \frac{w_{ij}}{\sum_k w_{kj}} \quad (1)$$

with $w_{ij} = \frac{1}{c_{ij}}$ *Aij* and where *Aij* is an element of the adjacency matrix *A*. The weights *wij* are nonnegative over the entire workspace. The $N \times N$ transition matrix *P* of the graph *G* is composed of the elements defined in Eq. 1. The node *vj* is marked as visited by removing its adjacent edges from the local copy of *Gp* and the algorithm is not allowed to walk through it again in the current random walk. To bias the search towards a not frequently visited subset of the state space, therefore increasing the probability to generate new paths in the next random walks, we adopt a Discounting Strategy(*G*, *α, vj*, *vi*) procedure. Each time we leave a node *vj* , the probability *pij* associated to the edge *eij* is multiplied by a *discounting factor* $\alpha \in (0, 1)$ (for parameter *α*), therefore, the probability to follow the edge *eij* in the next run of the walk decreases

$$pij := \alpha pij \, . \quad (2)$$

The transition matrix *P* is then properly normalized. It is worth mentioning that the case of discounting factor $\alpha = 1$ corresponds to our previous implementation of RHCF [5] with no biasing towards unexplored regions. Smaller values of *α* correspond to heavier bias. The walk stops when the goal node is found (which means, we have generated a *valid* path) or when we reach a node with all neighbours marked as visited. Each time a valid path **P** is generated, we compare it to the ones already found and save it in the result set *P* if **P** is new, i.e., not generated before. All the visited nodes are then marked unvisited.

C. Probabilistic Completeness of RHCF

In Lemma 1 we describe a homotopy-encoding property of the Voronoi diagram. Theorem 1 proves that RHCF finds any arbitrary path in an undirected weighted graph. Finally we prove that RHCF is probabilistically complete in Theorem 2. Here we denote $|P|i$ as the size (or cardinality) of the result set *P* at iteration *i*, *Lemma 1.* In the navigation graph *G*(*V*,*E*), built from a Voronoi diagram generated from a 2D environment, two different paths with the same *vs* and *vg* belong to different homotopy classes.

Proof: The Voronoi diagram is defined as the set of points in the free space which have equal distance to two or more closest obstacles. The Lemma 1 is derived from the defining property of the Voronoi diagram for 2D environments: only one path between any two obstacles exists. If two paths between two obstacles existed, then they would have different distance to each of those obstacles, which contradicts the definition of the Voronoi diagram. Hence two different paths have at least one obstacle between them, therefore cannot be continuously deformed into one another and belong to different homotopy classes.

Theorem 1: Given any arbitrary path in an undirected weighted graph, the *Randomized Homotopy Classes Finder* will find it with probability greater than zero.

Algorithm 1: Randomized Homotopy Classes Finder.

```
function RHCF(v_s, v_g, G, K, α)
k ← 0
P ← ∅
while k < K do
  P ← RandomWalk(v_s, v_g, G, α)
  P_new ← (P ∉ P) ∧ isValid(P)
  if P_new then
    P ← P ∪ P
    k ← k + 1
  end if
end while
return P
```

Algorithm 2: Random Walk.

```
function RandomWalk(v_s, v_g, G, α)
v_j ← v_s
G_p ← G
P ← v_j
while v_j ≠ v_g and E(v_j) ≠ ∅ do
  v_i ← RandomNeighbor(v_j)
  P ← P ∪ v_i
  G_p ← G_p \ E(v_j)
  G ← DiscountingStrategy(G, α, v_j, v_i)
  v_j ← v_i
end while
return P
```

Proof: A random walk is a sequence of transitions (or edges) from a vertex *vj* to another *vi* where at each step

International Journal of Electrical Electronics & Computer Science Engineering
Volume 4, Issue 1 (February, 2017) | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

an edge *eij* is chosen with a probability higher than zero, *pij* > 0.We assume that the weights *wij* are nonnegative over the entire workspace. Every possible path **P***k* in the graph is a concatenation of *Z* number of edges' transitions,

$$\mathbf{P}_k = \cup_{z=1}^{Z} e_{z,z-1}. \quad (3)$$

Given that all the transitions in the graph have a probability greater than zero (*pij* > 0), every possible concatenation of transitions has a probability greater than zero. Therefore, any arbitrary path **P***k* in the graph has non-zero probability to be found during the random walk.

$$Pr(\mathbf{P}_k) = \prod_{z=1}^{Z} p_{z,z-1} = p_{1,s}\, p_{2,1} \dots p_{Z,Z-1} > 0, k = 1...K. \quad (4)$$

Theorem 2: Consider an undirected graph *G*(*V*,*E*), built from a 2D Voronoi diagram, with nonnegative weights and with encoded *K* possible paths connecting the start vertex *vs* and the goal vertex *vg* . Then the probability that RHCF finds all the *K* paths lying in different homotopy classes, in the graph *G*(*V*,*E*) connecting *vs* to *vg* , that is the size of the result set *P* is equal to *K*, converges to 1 as the number of iterations *n* approaches infinity:

$$\lim_{n\to\infty} Pr\left(|\mathcal{P}|_n = K\right) = 1$$

path **P***k* (or walk) from *vs* to *vg* with a non-zero probability. Each time a valid and new path **P***k* is generated, it is added to *P* and thus increasing of one the size of *P*. Given that the graph *G*(*V*,*E*) is built from a 2D Voronoi diagram, by means of Lemma 1 every new valid path added to *P* belongs to a new homotopy class. Given enough time, from Theorem 1, any arbitrary new valid path **P***k* can be generated by the random walk and added to *P* therefore satisfying $\lim n\to\infty \Pr(|P|n = K) = 1$.

IV. EXPERIMENTAL SETUP

We present a set of experiments to evaluate the performance of our method and show that it outperforms current state-of-the art algorithms to compute a set of diverse paths presented by Voss *et al.* [1] and Kuderer *et al.* [2], detailed respectively in Section IV-E and Section IV-F. For our experiments we choose proper environments of varying complexity and describe appropriate metrics to demonstrate the efficiency of RHCF in terms of planning performance and solutions' quality. All the simulated experiments are carried out on a PC with 2.3 GHz Intel Core i5 and 4 GB of RAM. Each reported value is an average of 200 runs. In all the experiments we use path length as the edge costs.

A. Simulated Environments:

We design four simulated environments (shown in Fig. 3) to stress different properties of the planner and to study how the algorithm behaves in scenarios of varying complexity. In the *wall of people* scenario, the robot needs to find different ways to the goal through a queue of standing people. This scenario has 36 possible homotopy classes. In the *crowd* scenario,which has 380 possible homotopy classes, the people are placed in a sparser way forming different groups. In the *surrounded* scenario (710 homotopy classes), the robot is placed in the crowd, surrounded by several people. The *corridor* scenario (over 60000 homotopy classes) represents a challenging situation with a crowded corridor and dozens of people, walking alone and in groups.

B. Voronoi Diagram:

To generate a Voronoi diagram of the environment, we utilize the open-source C++ package developed by Lau *et al.* [12]: a computationally efficient approach that is based on incremental updates, applicable in dynamic environments, which was shown to dramatically reduce the computation time needed to build (and update) the Voronoi diagram. Lau's algorithm is in the same complexity class as a simple image passing algorithm, i.e., *O*(*n*2) for an *n* × *n* input map. The authors in [12] also show that in very narrow and cluttered environments Voronoi-based planning (first build a Voronoi diagram and then plan over it) outperforms single-query sampling-based motion planners (like RRT and KPIECE) in terms of planning efficiency.

C. Performance Metrics:

To quantify the planning performance and quality of our approach, we compute the averages and the standard deviations of the following metrics: *Tk* time to get *K* paths, *nCGk* normalized cumulative gain, *RDk* robust diversity of the result set *PK* of *K* paths returned by the algorithms. For runtime evaluation we are only interested in measuring the time to generate K paths in the navigation graph, excluding the time to compute the Voronoi diagram of the environment or generate the probabilistic roadmap graph (PRM) that is used by Voss' algorithm. We report the time to build the navigation graph separately. The *robust diversity* measures how large are the intra-set distances between pairs of paths in *PK* . Let us consider the distance between two paths *pa* and *pb* to be the discrete Fr´echet distance *df* (*pa*, *pb*), as in [1]. We define *RDk* as

$$RD_k = \frac{1}{|\mathcal{P}_K|} \sum_{p_a \in \mathcal{P}_K} \min_{p_b \in \mathcal{P}_K, p_a \neq p_b} d_f(p_a, p_b)$$

Higher diversity value indicates better spatial separation of paths in the set. The *normalized cumulative gain nCGk* is used to evaluate ranking performance of web search engine algorithms. It computes how far is the candidate ranking set (e.g., a set of *K* random paths) from the ideal ranking set (a set of*K* best paths). *nCGk*

International Journal of Electrical Electronics & Computer Science Engineering
Volume 4, Issue 1 (February, 2017) | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

is based on the definition of relevance (*rel*) of a single path. We define the relevance as the inverse of the path cost:

$$rel(p) = \frac{1}{cost(p)}$$

To paths with smaller costs correspond higher *rel* values. The cumulative gain *CGk* of a set of paths *PK* is the sum of *rel* values of all paths in the set:

$$CG_k = \sum_{p_i \in \mathcal{P}_K} rel(p_i)$$

The cumulative gain is normalized by the maximum cumulative gain of *K* best paths in the graph between the start and goal points:

$$nCG_k = \frac{CG_k}{\max(CG_k)}$$

Therefore the *nCGk* of the *K* best paths, e.g., found by Kuderer's algorithm, equals 1 for any*K*. In general, *nCGk* $\in$ [0, 1], with higher values corresponding to sets of paths with lower costs. It is important to note the trade-off between the quality of paths in the set (e.g., their lengths) and diversity: the best *K* paths are often very similar to each other, contributing to higher cumulative quality of the path set at the cost of very lowdiversity (high *nCGk*, low *RDk*). Adding diverse paths to the set may decrease the *nCGk* value. To provide a baseline, in our evaluation we compare the *nCGk* value of RHCF to the *nCGk* value of a *Uniform Random Path*: a na¨ıve, uninformed algorithm, that samples random *K* paths uniformly from the set of all possible paths between two nodes.

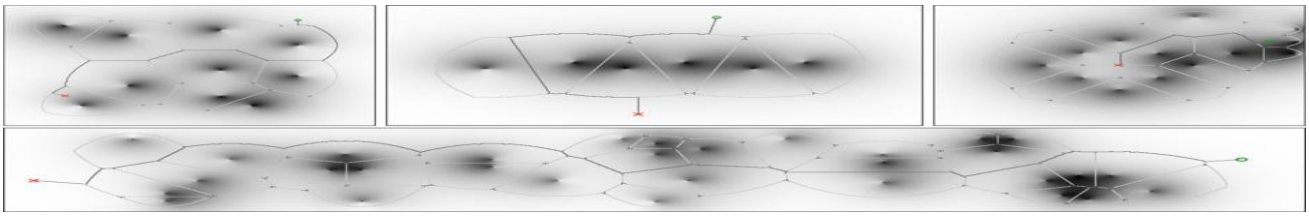

Fig. 3. Top left: crowd environment, top middle: wall of people scenario, top right: surrounded environment. At the bottom we have the corridor environment. In all the environments, the occupancy grids' cells are marked as obstacles according to the humans poses and their personal space. We assume that humans poses are provided by a people tracker. Personal space of human agents is displayed in grey scale, with darker regions corresponding to higher social cost: its peaks represent the agent positions. The red cross represents the robot position, the goal is displayed by the green circle. The edges of the Voronoi diagram are in dark grey and example paths generated by our approach are displayed with **black** edges.

D. RHCF Parameters:

Prior to the main experiments, we analyze the impact of the single parameter α on the performance of RHCF. After an informal validation, we see that the number of random walks *N RWk* needed to generate *K* distinct paths, and consequently RHCF runtime, decreases monotonically as α goes from 1 to 0.5 (see Fig. 4, where we show the results for the *crowd* scenario, but qualitatively similar trends are visible in the other scenarios too).

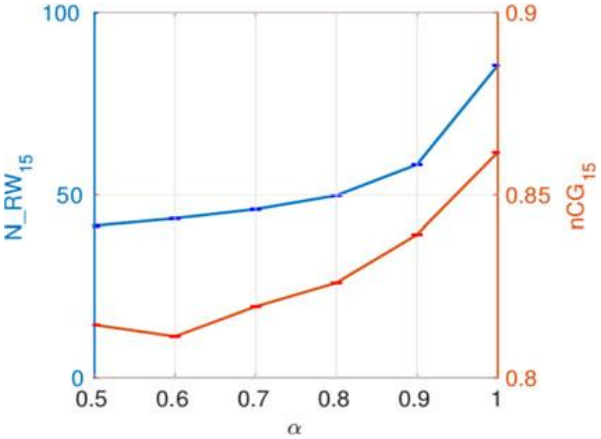

Fig. 4. Value of the normalized cumulative gain *nCG*15 (in red) and number of random walk iterations *N_RW*15 (in blue) needed to generate 15 homotopy classes (same trends visible with other values of *K*) for different values of α in the *crowd* scenario. *N_RWk* (consequently also *Tk*) monotonically decreases with a smaller value of α. The *nCGk* has only a slight decrease over the same α range.

For our experiments we choose α = 0.8, where we achieve a good trade-off between planning time and path quality: a smaller α value yields no considerable improvement to the planning time but causes further decrease of the normalized cumulative gain, therefore compromising the path quality.

E. Voss's Algorithm:

Voss' algorithm seeks to generate a set of diverse paths in the roadmap navigation graph (PRM) of the environment. We evaluate Voss' algorithm on the following metrics: *Tk* time to generate *K* paths, and *RDk* robust diversity of the result path set. We use a C++ implementation, provided by the authors of the paper. Parameters of Voss' algorithm are set empirically, following the suggestions from their paper: we choose the recommended branching factor $b = 2$, we set the ball radius ρ in each scenario individually to the highest value that still returned an average of 80% of *K* paths requested, typically $\rho \approx 0.1 - 0.2$. The number of PRM samples is set to 100–500, depending on the complexity of the scenario. We use the C-space distance for

International Journal of Electrical Electronics & Computer Science Engineering
Volume 4, Issue 1 (February, 2017) | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

simulated obstacles placement and the filtration step of the algorithm accepts all paths.

F. Kuderer's Algorithm:

Both Kuderer's algorithm and RHCF seek to find paths on a navigation graph based on the Voronoi diagram of the environment. Kuderer's approach employs Katoh's algorithm to find the*K* best paths in the graph. For a fair runtime comparison, we consider three popular algorithms for finding the *K* best paths (Yen's, Katoh's and Eppstein's) and choose the Yen's algorithm [11] for comparison. Yen's algorithm finds *K* shortest loopless paths for a given pair of start and goal poses. The algorithm's computational upper bound increases linearly with the value of *K*: with modern data structures it can be implemented in $O(KN(M + Nlog(N)))$ worst-case time.We use the C++ implementation by Martins and Pascoal [13], which is reported to have better performance than the straightforward implementation. Yen's loopless *K* best paths have higher diversity than the paths with loops found by Eppstein's algorithm. Additionally, as shown by Brander and Sinclair [10], for small size graphs and paths with a small number of vertices (like the graphs generated from a Voronoi diagram), Yen's algorithm is faster than Katoh's. We evaluate Kuderer's algorithm on the following metrics: *Tk* , *RDk* . We also measure the *nCGk* value of our *K* random paths with respect to the *K* best paths found in the Voronoi diagram by Kuderer's algorithm.

TABLE I
PLANNING TIME RESULTS

T_k [ms], $K = 10$			
Scenarios	RHCF	Kuderer	Voss
Crowd	0.36 ± 1.87	1.9 ± 3.93	183 ± 135.72
Corridor	3.86 ± 4.90	8.85 ± 4.50	6474 ± 1749
Wall of People	0.36 ± 1.88	0.85 ± 2.78	242 ± 540
Surrounded	0.52 ± 2.21	1.8 ± 3.85	69 ± 66
T_k [ms], $K = 50$			
Scenarios	RHCF	Kuderer	Voss
Crowd	1.8 ± 3.8	7.4 ± 4.7	6541 ± 13732
Corridor	9 ± 4.4	38.6 ± 6.2	98570 ± 89990
Wall of People	-	-	-
Surrounded	2.3 ± 4.2	6.7 ± 4.9	2406 ± 3234

V. RESULTS AND DISCUSSION

Tables I–V collect the empirical results generated for all the scenarios. The best values are highlighted in boldface. RHCF significantly outperforms the baselines with respect to all the performance metrics. Moreover, we test the approach in realworld experiments by applying it to socially-aware navigation in dynamic environments.

A. Empirical Results:

Table I shows the planning time results for $K = \{10, 50\}$: our approach is at least two times faster than Kuderer's to find a subset of homotopy classes among those present in the navigation graph. In very complex scenarios the difference in runtime becomes significant, e.g., 18.8 ms vs. 143.9 ms in the *corridor* scenario for $K = 200$. Moreover, RHCF is faster than Voss's algorithm by several orders of magnitude, also if the graph building times were considered. Table II reports the building times for the Voronoi-based navigation graphs and PRM graphs (the latter built using the OMPL library [14]) for each scenario.

Table III details the planning time obtained by state-of-the art sampling-based motion planners (these experiments were carried out using the OMPL library [14]) to find a pure geometric solution for the *surrounded* scenario. As pointed out in Section IV-B in very complex environments, like the case of the *surrounded* scenario where start and goal poses are surrounded by obstacles, a Voronoi-based path planner outperforms sampling-based motion planners, whose trees (or graphs) fatigue to escape from narrow corridors and complex

TABLE II
NAVIGATION GRAPH BUILDING TIME

Building Time [ms]		
Scenarios	Voronoi Graph	PRM Graph
Crowd	8.2	130.9
Corridor	30.7	530
Wall of People	5.5	56
Surrounded	6	54

TABLE III
COMPARISON TO SAMPLING-BASED MOTION PLANNERS

Surrounded Scenario	
Algorithms	Planning Time [ms]
RRT	1007.2 ± 1.5
KPIECE	1007.3 ± 1.3
STRIDE	1007.1 ± 1.4
EST	1007.1 ± 1.7
Informed RRT*	1006.9 ± 1.4

TABLE IV
CUMULATIVE GAIN RESULTS

nCG_{10}		
Scenarios	RHCF	Uniform RP
Crowd	0.7839 ± 0.0628	0.4216 ± 0.0454
Corridor	0.5567 ± 0.0435	0.4781 ± 0.0363
Wall of People	0.8884 ± 0.0451	0.5789 ± 0.0929
Surrounded	0.8009 ± 0.0699	0.3538 ± 0.0346

International Journal of Electrical Electronics & Computer Science Engineering
Volume 4, Issue 1 (February, 2017) | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

areas of the environments. In our evaluation for the case of the *surrounded* scenario, sampling-based motion planners are several orders of magnitude slower to find a single solution to the path planning problem while our approach needs a few milliseconds (in average 8.3 ms) to build the navigation graph and to find 50 different paths. In the other scenarios RHCF and the evaluated sampling-based motion planners are equally fast.

Table IV details the results related to the normalized cumulative gain for $K = 10$. RHCF, whilst being faster, also finds solutions with a gain close to the maximum value of 1, which means the solution quality is very high. As a reference, we provide *nCGk* results of the *Uniform Random Paths* (Uniform RP) algorithm that draws samples from the set of all paths between two nodes with uniform distribution. RHCF revealsmuch higher *nCGk* results, indicating its bias towards high quality solutions. Only in one scenario, *corridor*, we have a lower *nCGk* : this is due to the higher number of total homotopy classes of the scenario. In this case the above mentioned quality-diversity trade-off is prominent: the best K paths of over 60000 present in the *corridor* scenario are very similar, so introducing a certain degree of diversity into the path set inevitably leads to lower normalized cumulative gain.

TABLE V
ROBUST DIVERSITY RESULTS

	RD_{10}		
Scenarios	RHCF	Kuderer	Voss
Crowd	3.01 ± 0.17	2.76	1.34 ± 0.22
Corridor	3.83 ± 0.26	2.13	2.97 ± 0.30
Wall of People	2.51 ± 0.10	2.49	1.44 ± 0.13
Surrounded	2.05 ± 0.19	1.60	1.04 ± 0.18

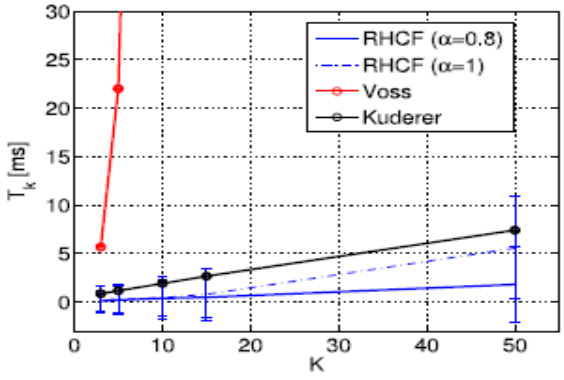

Fig. 5. Planning time **T***k* for different values of *K* in the *crowd* scenario. In blue our approach considering two values of the discounting factor α, in black Kuderer's algorithm and in red Voss's method. Our approach is faster than all the baselines. The introduction of the discounting strategy ($\alpha = 0.8$) further improved the planning performance of our previous RHCF implementation ($\alpha = 1$).

Table V details the diversity of the paths generated by the approaches for $K = 10$: RHCF outperforms Kuderer's and Voss's algorithms in all the environments, delivering more diverse path sets. Figs. 5–7 show the metrics trends for different values of *K* in the *crowd* scenario (same trends are visible in other scenarios). In all figures the standard deviation is depicted with vertical lines. RHCF is significantly faster than the baselines, as Fig. 5 indicates. The introduction of the discounting strategy further improved the planning performance of our previous RHCF implementation [5]. Our approach has noticeably higher robust diversity *RDk* , see Fig. 6: the paths produced are more diverse than the ones generated by the baselines for small values of *K*. RHCF quickly converges to the optimal value of the normalized cumulative gain as *K* increases, consequently generating high quality paths (see Fig. 7, where the *nCGk* trends are reported for all the scenarios).

B. Application to Social Navigation:

We conduct further experiments in real-world settings. RHCF allows us to address the task of social robot navigation as a qualitative planning problem that enables a robot to evaluate several diverse paths (see Figs. 1 and 8) with respect to social costs, more rapidly than the other baselines. More specifically,we integrate the approach in a hierarchical motion planning framework that re-plans at a given frequency (6 Hz)1: firstly a set of diverse paths lying in different homotopy

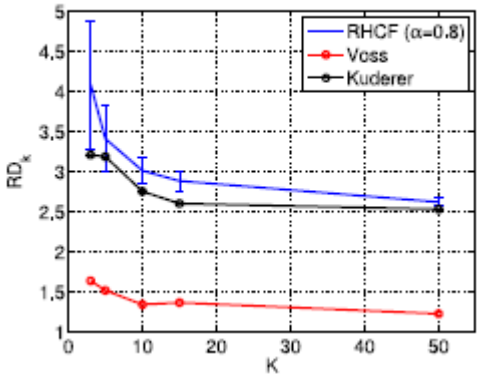

Fig. 6. Robust diversity **RD***k* obtained by varying *K* in the *crowd* scenario. The paths produced by our approach are more diverse than the ones generated by the baselines for small values of *K*.

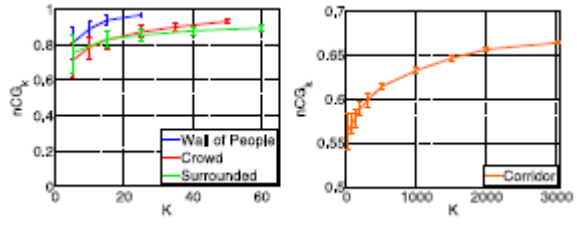

Fig. 7. Normalized Cumulative Gain **nCG***k* obtained by varying *K* in the *wall of people*, *crowd*, *surrounded* and *corridor* scenarios. As *K* increases, our approach converges to the optimal value of the normalized cumulative gain **nCG***k*.

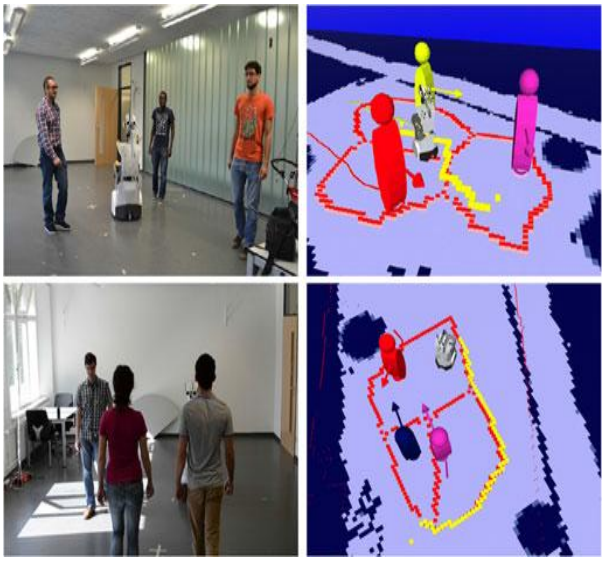

Fig. 8. Application in social settings. Upper Left: dynamic scene with three people walking, the robot has to reach a point in front of them, respecting a social cost. Upper Right: The robot reaches its goal by following the yellow path selected by RHCF from the Voronoi diagram (in red). People, tracked by a multimodal tracker [17], are represented by colored cylinder-shaped objects. Bottom row: three people that interact with each other and robot following the rapidly generated socially acceptable path.

classes is generated from a Voronoi diagram using RHCF, subsequently among those we choose the best path according to a social cost based on the social force model by Helbing [15]. Finally, a smooth trajectory with a velocity profile that respects the dynamic constraints of the robot is generated in the chosen homotopy class by using an non-holonomic RRT* based algorithm [16]. Throughout several experiments where the movements of the people were uncertain, our method promptly reacts to the environment changes while generating high quality solutions and respecting the social constraints of the scene. As Section V-A points out, for this task (where the robot is trapped by a group of people moving around it) using sampling-based motion planners, to plan a path considering the entire environment, would result in higher planning times.

VI. CONCLUSIONS

In this paper we introduce theRandomized Homotopy Classes Finder that finds a set of *K* diverse paths belonging to distinct homotopy classes in an undirected weighted graph built from a Voronoi diagram.We prove that our approach is probabilistically complete, i.e., it finds all the paths, and therefore all the homotopy classes. Our extensive experimental evaluation shows that RHCF finds diverse paths significantly faster than two state-of the-art methods. Moreover, as the cumulative gain results show, the paths produced by our approach are of similar quality to Kuderer's true *K* best paths. A key property of our method is that it computes a set of more diverse paths with respect to the baselines: usually in dynamic environments spatially separated paths are more robust to invalidation due to unexpected obstacles, than similar or adjacent paths. Furthermore we test our approach in real-world experiments by applying it to social navigation settings in dynamic environments. In future work, we intend to formally study the space and time complexity of the algorithm. Additionally, we are interested in extending the algorithm to generate a more robust paths set by considering sensing and motion uncertainty.

VII. REFERENCES

[1] C. Voss, M. Moll, and L. E. Kavraki, "A heuristic approach to finding diverse short paths," in Proc. Int. Conf. Robot. Autom., Seattle,WA, USA, 2015, pp. 4173–4179.

[2] M. Kuderer, C. Sprunk, H. Kretzschmar, andW. Burgard, "Online generation of homotopically distinct navigation paths," in Proc. Int. Conf. Robot. Autom., Hong Kong, China, 2014, pp. 6462–6467.

[3] P. Liu, G. Xiong, H. Zhang, Y. Jiang, J. Gong, and H. Chen, "A multi-path selecting navigation framework with human supervision," in Proc. 4th Int. Conf. Social Robot., Chengdu, China, 2012, pp. 506–515. Available at: http://lissi.fr/iros-ar2015/doku.php/wiki/program

[4] F. T. Pokorny,M. Hawasly, and S. Ramamoorthy, "Multiscale topological trajectory classification with persistent homology," in Proc. Robot.: Sci. Syst., Berkeley, CA, USA, Jul., 2014.

[5] L. Palmieri, A. Rudenko, and K. O. Arras, "A fast randomized method to find homotopy classes for socially-aware navigation," in Proc. Intell. Robots Syst. Workshop Assistance Service Robot. Human Environ., Hamburg, Germany, 2015.

[6] S. Bhattacharya, V. Kumar, and M. Likhachev, "Search-based path planning with homotopy class constraints," in Proc. 3rd Annu. Symp. Combinatorial Search, AAAI, 2010, pp. 2097–2099.

[7] P.Vela, A.Vela, andG.Ogunmakin, "Topologically based decision support tools for aircraft routing," in Proc. Digit. Avionics Syst. Conf., Salt Lake City, UT, USA, 2010, pp. 1.A.5-1–1.A.5-11.

[8] D. Demyen and M. Buro, "Efficient triangulation-based pathfinding," in Proc. AAAI Conf. Artif. Intell., 2006, pp. 942–947.

[9] N. Katoh, T. Ibaraki, and H. Mine, "An efficient algorithm for k shortest simple paths," Networks, vol. 12, no. 4, pp. 411–427, 1982.

[10] A.W. Brander and M. C. Sinclair, "A comparative study of k-shortest path algorithms," Proc. 11th UK Perform. Eng. Workshop, 1996, pp. 370–379.

[11] J. Y. Yen, "Finding the k shortest loopless paths in a network," Manage. Sci., vol. 17, no. 11, pp. 712–716, 1971.

[12] B. Lau, C. Sprunk, and W. Burgard, "Efficient grid-based spatial representations for robot navigation in dynamic environments," Robot. Auton. Syst., vol. 61, no. 10, pp. 1116–1130, 2013.

[13] E. Q. Martins and M. M. Pascoal, "A new implementation of Yen's ranking loopless paths algorithm," Quart. J. Belgian, French Italian Oper. Res. Societies, vol. 1, no. 2, 121–133, 2003.

[14] I. A. S¸ucan, M. Moll, and L. E. Kavraki, "The open motion planning library," IEEE Robot. Autom. Mag., vol. 19, no. 4, pp. 72–82, Dec. 2012. [Online]. Available: http://ompl.kavrakilab.org

[15] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," Phys. Rev. E, vol. 51, no. 5, 4282–4286, 1995.