# A Fast Randomized Method to Find Homotopy Classes for Socially-Aware Navigation

Luigi Palmieri Andrey Rudenko Kai O. Arras .

*Abstract*— **We introduce and show preliminary results of a fast randomized method that finds a set of $K$ paths lying in distinct homotopy classes. We frame the path planning task as a graph search problem, where the navigation graph is based on a Voronoi diagram. The search is biased by a cost function derived from the social force model that is used to generate and select the paths. We compare our method to Yen's algorithm, and empirically show that our approach is faster to find a subset of homotopy classes. Furthermore our approach computes a set of more diverse paths with respect to the baseline while obtaining a negligible loss in path quality.**

## I. Introduction

In socially-aware navigation, like in the case where a robot assists, informs and guides passengers in large and busy airports [1], the motion planner deals with non-static scenarios where crowds of pedestrians and dynamic obstacles regularly invalidate paths generated by using standard approaches (e.g. A* [2], RRT [3], PRM [4]). Each time a path is invalidated, a new motion planning problem has to be solved or in the case of replanning algorithm like D* [5] a repairing phase should recompute a new path by first updating costs over a grid map. Having a set of $K$ precomputed distinct paths, that may be checked for validity in case of the appearance of unexpected obstacles, is a more reasonable approach than solving from scratch the motion planning problem or to replan for each environment's change. Moreover a more rational approach would be to generate $K$ distinct paths from $K$ different homotopy classes. A homotopy class is defined by the set of paths with the same start and goal points which can be continuously deformed into one another without intersecting obstacles.

Different approaches have been already introduced to generate a set of paths belonging to different homotopy classes. Demeyen and Buro [6] introduce a method for efficient path planning that searches on a graph built using constrained Delaunay triangulations. The obstacles are described via polygonal representation. The paths, found in the graph, represent different homotopy classes. Eriksson *et al* [7] find $K$ homotopy classes solving the $K$ shortest paths problem in a two-dimensional environment with polygonal obstacles. They introduce the $Kth$ shortest path map: a map of the entire free workspace, partitioned into equivalence class regions such that the $k_{th}$ shortest path from a start vertex $s$ to any point in a single region has the same structure. Moreover they introduce a simple visibility-based algorithm, based on Yen's algorithm [8], for computing the $K$ shortest paths between two fixed points. Bhattacharya *et al.* [9] propose a method to find different homotopy classes based on A* search over an augmented graph. The graph represents the topological information via the H-signature, a complex analysis value that characterizes a homotopy class, the graph may contain multiple paths to the goal within the same homotopy class. Kuderer *et al* in [10] select *K* best homotopy classes by generating *K* shortest paths using Katoh's algorithm [11]. During navigation the paths feed an optimization algorithm used to generate homotopically distinct kinodynamic trajectories. Among those the best one is selected for the navigation. They show the method is one order of magnitude faster than [9]. Vela *et al* [12] detail a decision support tool to aid air traffic controllers and managers in re-routing traffic: they generate a set of homotopy classes for a given pair of start and goal poses, by computing the $K$ shortest paths via the Dijkstra algorithm on a Voronoi graph. The latter is generated from a map that encodes

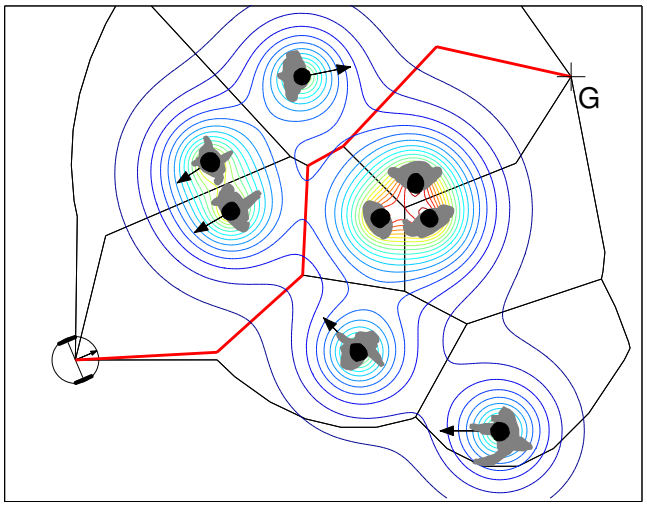


Fig. 1. An example path selected from the $K$ best homotopy classes in the Voronoi diagram. The robot is enclosed in the **black circle**, in **red** the path selected to reach the goal position **G**. The **black** Voronoi diagram describes the possible ways to go through a crowd by implicitly encoding different homotopy classes.

L. Palmieri, A. Rudenko, K.O. Arras are with the Social Robotics Lab, Dept. of Computer Science, University of Freiburg, Germany. {palmieri,arras}@cs.uni-freiburg.de, andrey.rudenko@saturn.uni-freiburg.de

weather conditions. A final step optimize the set of $K$-shortest paths with respect to path length and weather avoidance. Voss *et al* [13] introduce an algorithm that seeks to find a set of diverse, short paths through a roadmap graph. The algorithm finds diverse homotopy classes by finding diverse shortest paths avoiding a collection of balls imposed on the graph as simulated obstacles. The authors compare their approach to the Eppstein algorithm [14] that finds the optimal set of $K$ shortest paths with loops and show that, with tolerable loss in shortness, they produce equally diverse path sets orders of magnitude more quickly. Hernandez *et al* [15] propose and compare three different path planning algorithms that exploit the set of homotopy classes generated for any 2D workspace: Homotopic A*, Homotopic RRT and Homotopic Bug. Their method to generate homotopy classes modifies the one introduced by Jenkins [16]: it first builds a reference frame in the workspace which is used to identify the homotopy classes and afterwards it builds a topological graph which allows an easy and systematic computation of homotopy classes. The homotopy classes are sorted according to a lower bound heuristic estimator, then they are used to guide and to constrain topologically the path search.

Instead of solving the $K$ shortest paths problem with deterministic graph search algorithms like in [9, 10, 12], we introduce and show preliminary results of a fast randomized method based on random walks that finds a set of $K$ homotopically distinct paths, according to a social cost. As in [10] we build a navigation graph from the Voronoi diagram. Each path found in the Voronoi diagram represents a distinct homotopy class. Differently from [13, 10] we compare our method to Yen's algorithm, a fast algorithm that finds loopless paths. In [13] the authors compare their approach to Eppstein's algorithm which finds paths with loops therefore having a lower chance to find a more diverse set of paths. In [10] the authors use Katoh's algorithm, however, it was shown by Brander and Sinclair [17] that for small size graphs and paths of small number of vertices, like in the case we consider, Yen's is faster than Katoh's. Furthermore in our approach, instead of using a polygonal representation of the obstacles as in [6, 7], we use occupancy grids where obstacles are represented by blocked cells, therefore permitting an easier integration with existing mapping frameworks. We assume in our work that the pedestrians' poses are given by a people (or group) tracker [18, 19].

The contribution of our paper is as follows:

- we introduce a fast, and easy to implement, randomized approach to find a set of $K$ paths belonging to $K$ different homotopy classes.
- we perform an extensive evaluation and compare our method to Yen's algorithm. Our approach is faster than Yen's to find a subset of homotopy classes in a Voronoi diagram;
- moreover our approach generates more diverse paths than Yen's (the robot has a more diverse set from where to choose the path to follow), while obtaining a negligible loss in path quality.

The paper is structured as follows: we detail our new approach and its analysis in Section II. We present the experiments in Section III and discuss the results in Section IV. Section V concludes the paper.

## II. Our Approach

We introduce a probabilistic approach to find paths belonging to different homotopy classes for socially-aware robot motion planning. Our approach is complete, it can find all the possible paths, namely all the homotopy classes implicitly encoded in the navigation graph built from a Voronoi diagram that describes the scenario. Having a set of possible paths, we choose the best one according to a cost that considers social interactions between humans.

### A. Navigation Graph

Our method could be used with discrete information received from a people tracker, however, it is equally well suited for use on general occupancy grids. In our case the input scenario is given as a collection of discrete people poses, $(x, y, \theta)$, in the 2D workspace, see Fig.2 (left). To frame the motion planning task as a graph search problem, we build the navigation graph *G* of the scenario from the Voronoi diagram *VD*, generated considering as obstacles also the people poses, see Fig.2 (middle). We create two additional vertices for the initial robot position and goal position, and connect them to the closest point of the Voronoi diagram, see Fig.2 (right). In the navigation graph, built in this way, different paths from the initial robot position to the goal position belong to different homotopy classes.

The graph *G(V,E)* consists of a set of nodes (or vertices) *V* and a set of edges *E*. In this work, *N* is the number of nodes in the graph and *M* the number of edges. We associate to each edge $e_{ij}$, connecting the node $v_j$ to its neighbor $v_i$, an attribute or cost $c_{ij}$. $E(v_j)$ denotes the set of incoming and outgoing edges of $v_j$.

We compute the set of homotopy classes by running our random walk based algorithm on *G*. A walk $w$ of length $k-1$ in a graph is a sequence of nodes $v_1, v_2, \ldots, v_k$, where each pair of nodes is connected by an edge, $(v_{i-1}, v_i) \in E$ for $1 < i \leq k$ . The adjacency matrix $A$ of $G$ expresses the topology of the graph and is defined as

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

*Walks* are usually referred to as *paths*.

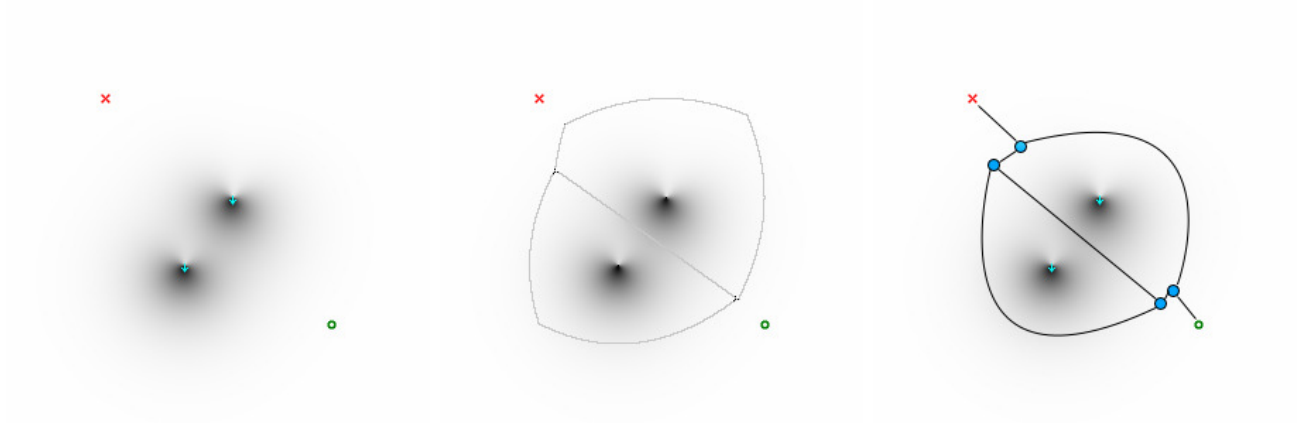

Fig. 2. In all the figures the **red** cross is the robot position, the **green** circle is the goal. **Left:** pedestrians' poses, marked with **blue** arrows, are provided by the people tracker, a social force field is generated over the two pedestrians. **Middle:** a Voronoi diagram is built considering the pedestrian positions as obstacles. The Voronoi diagram points are displayed in **darker grey**. **Right:** navigation graph built from the Voronoi diagram. The robot and goal nodes are mapped to their closest Voronoi points. The navigation graph **vertices** are depicted in **blue**, its edges in **black** .

### B. Randomized Homotopy Classes Finder

To find different homotopy classes we introduce the Randomized Homotopy Classes Finder (**RHCF**), detailed in Alg.1. We iteratively run the random walk algorithm, see Alg.2, on the weighted undirected graph until $K$ distinct paths are found.

Given the weighted graph $G$, we start at the initial node $v_s$, the one where the robot position is mapped to.

At each step of the random walk we choose a random neighbor of the current node $v_j$ (see RandomNeighbor($v_j$) in Alg.1) with probability $p_{ij}$ inversely proportional to the cost $c_{ij}$ associated to the edge $e_{ij}$

$$p_{ij} = \frac{\frac{1}{c_{ij}} A_{ij}}{\sum_k \frac{1}{c_{kj}} A_{kj}} \tag{1}$$

where $A_{ij}$ is an element of the adjacency matrix $A$.

The transition matrix $P$ of the graph $G$ is defined as the $N \times N$ matrix where each of its elements is defined as in Eq.1.

Each time we leave a node, we mark it as visited (by removing it from the local copy of the graph $G_p$) and do not allow the algorithm to walk through it again in the current run of the random walk. The walk stops when the goal node is found or when we reach a node with all neighbors marked as visited. Each time a path **P** is generated, we compare it to the ones already found. If the same path was not generated before and it is a valid path, we save it in our homotopy classes set $H$. A path is valid if its last node is $v_g$. All the visited nodes are then marked unvisited.

After the $K$ paths have been found, the robot chooses the best one to follow in terms of the social cost function. In the case that the followed path is invalidated by an unexpected obstacle, the planner selects the best path from the set of available paths, given the current status of the robot and of the environment.

**Algorithm 1** Randomized Homotopy Classes Finder

**function RHCF(**$v_s$, $v_g$, $G$, $K$**)**
$k = 0$
**while** $k < K$ **do**
  **P** $\leftarrow RandomWalk(v_s, v_g, G)$
  **if P** $\notin H$ **and** $isvalid$(**P**) **then**
    $H \leftarrow H \cup$ **P**
    $k = k + 1$
  **end if**
**end while**

**Algorithm 2** Random Walk

**function RandomWalk(**$v_s$, $v_g$, $G$**)**
$v_j \leftarrow v_s$
$G_p \leftarrow G$
**P** $\leftarrow v_j$
**while** $v_j \neq v_g$ **do**
  $v_i \leftarrow RandomNeighbor(v_j)$
  **P** $\leftarrow$ **P** $\cup\, v_i$
  $G_p \leftarrow G_p \setminus E(v_j)$
  $v_j \leftarrow v_i$
**end while**
**return P**

### C. Cost definition

Several state of the art approaches focus on finding the $K$-shortest paths [9, 10, 12, 13]. Here we are interested in finding the $K$ best homotopy classes for socially-aware motion planning, therefore we compute the edges weights $c_{ij}$ by using a cost function derived from the social force model introduced by Helbing [20].

The cost $c_{ij}$ is defined by the line integral of the pedestrians' interaction forces on the planar curve $s_{ij}$ described by the edge $e_{ij}$ and the edge length $l_s$.

$$c_{ij} = \int_{s_{ij}} F_s \; ds + l_s \tag{2}$$

The force $F_s$ represents the force generated from the interactions of all the pedestrians $p_i$ with the robot, defined as pedestrian $p_j$,

$$F_s = \sum_{i \in \mathcal{P}} \mathbf{f}_{i,j} \tag{3}$$

with $\mathcal{P} = \{p_i\}_{i=1}^{N_p}$ being the set of $N_p$ pedestrians. The forces $\mathbf{f}_{i,j}$ decrease proportional to the distance of their sources to the robot, and are modelled as

$$\mathbf{f}_{i,j} = a_j e^{\left(\frac{r_{i,j} - d_{i,j}}{b_j}\right)} \boldsymbol{n}_{i,j} \tag{4}$$

where $a_j$ specifies the magnitude and $b_j$ the range of the force. The distance $d_{i,j}$ is given by the Euclidean distance between the pedestrian $p_i$ and robot, $r_{i,j}$ is the sum of their radii (we approximate each pedestrian with a circle). The term $\boldsymbol{n}_{i,j}$ is the normalized vector pointing from $p_i$ to the robot which describes the direction of the force.

To better describe the limited field of view of the pedestrian, the forces are scaled with an anisotropic factor (see Fig.3)

$$\mathbf{f}_{i,j} = a_j e^{\left(\frac{r_{i,j} - d_{i,j}}{b_j}\right)} \boldsymbol{n}_{i,j} \left(\lambda + (1-\lambda)\frac{1 + \cos\left(\varphi_{i,j}\right)}{2}\right) \tag{5}$$

where $\lambda$ defines the strength of the anisotropic factor and

$$\cos\left(\varphi_{i,j}\right) = -\boldsymbol{n}_{i,j} \cdot \widehat{\boldsymbol{e}}_i \,. \tag{6}$$

with $\widehat{\boldsymbol{e}}_i$ representing the direction of the pedestrian $p_i$.

Notice that our approach is not limited to a single definition of cost, different definitions from the social navigation literature could be used [10, 21, 22].

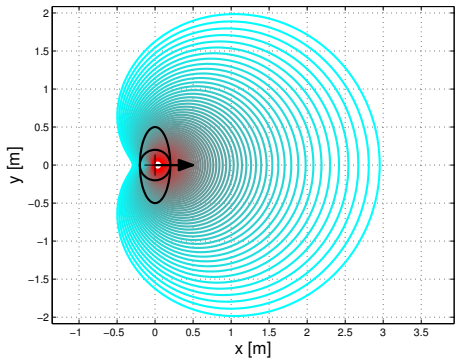


Fig. 3. Anisotropic influence model for $\lambda = 0.1$, $a_j = 2$, $b_j = 1$, $r_{ij} = 0.4$ generated by the social force $F_s$ for a single pedestrian. **Red** regions denote higher cost regions.

## III. Experiments

To evaluate our approach in terms of planning performance and quality of the solutions, we design a set of experiments by choosing proper environments and metrics. We compare our approach to Yen's algorithm. In both algorithms we adopt the cost defined in Sec.II-C. The algorithms are implemented in C++. All experiments were running on an ordinary PC with 2.3 GHz Intel Core i5 and 8 GB of RAM.

### A. Yen's Algorithm

In [8] Yen introduces an algorithm to find $K$-shortest loopless paths for a given pair of start and goal poses. The algorithm computational upper bound increases linearly with the value of $K$: with modern data structure it can be implemented in $O(KN(M + Nlog(N)))$ worst-case time. We use the C++ implementation introduced by Martins and Pascoal in [23], which is reported to have better performance than the straightforward implementation. We compare our approach to the Yen's algorithm, because it finds a set of $k$ best paths with an higher diversity than the ones found by Eppstein's and it was shown by Brander and Sinclair [17] that for small size graphs and paths of small length (like the graphs generated from a Voronoi diagram), it is faster than Katoh's.

### B. Environments

We design four different environments (shown in Fig 4), to stress different properties of the planner and to study how the algorithm behaves in environments of varying complexity. We choose scenarios resembling potential situations that could occur while a robot is navigating into an airport. 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 33 possible homotopy classes. In the *crowd A* (670 homotopy classes) and in the *crowd B* (576 homotopy classes) scenarios, the people are placed in a sparser way forming different groups. In the scenario *surrounded*, that has 1826 possible homotopy classes, the robot is placed in the crowd, surrounded by several people.

### C. Metrics

To quantify planning performance and difference in quality with respect to Yen's search method we compute the averages of the following metrics: $T_k$ time to get $K$ different homotopy classes, $nCG_k$ normalized cumulative gain, $RD_k$ robust diversity of a set $\mathbf{P}_K$ of K paths returned by the algorithms. The *robust diversity* measures how large are the intra-set distances between pairs of paths in the set $\mathbf{P}_K$. Let us consider $d_f(p_a, p_b)$ to be the Fréchet distance between two paths $p_a$ and $p_b$ evaluated at the vertices, as in [13]. We define $RD_k$ as

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

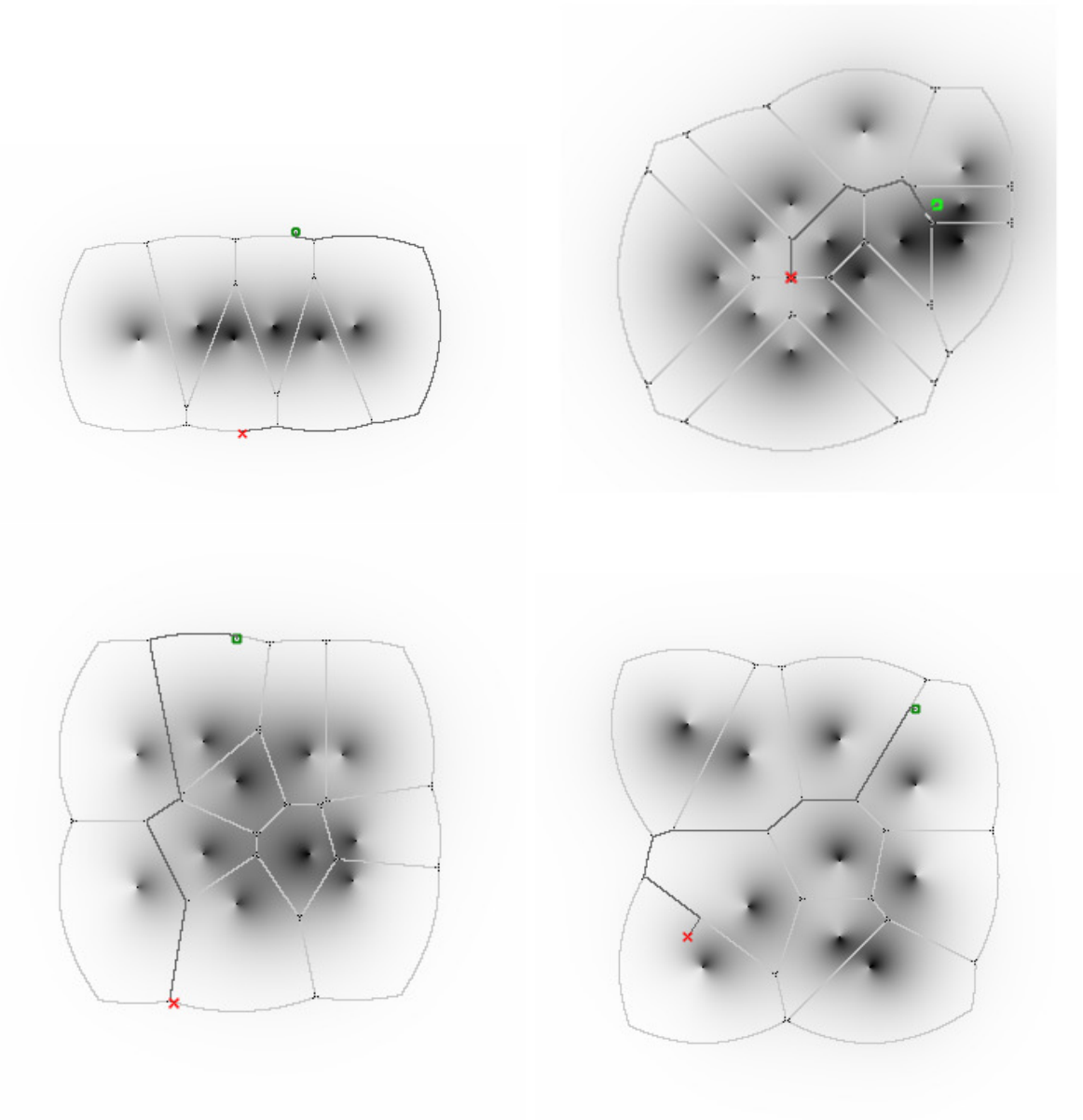

Fig. 4. In all the environments, the social force is displayed in grey scale: darker regions have a higher social cost. The peaks of the social force field represent the agent positions. The scenario at **top left** is the *wall of people* environment. The one at **top right** part of the figure is the *surrounded* environment. At the **bottom left** and at the **bottom right** are respectively the *crowd A* and the *crowd B* scenarios. **Red** crosses represent the robot position, the goal is displayed by **green** circles. In all the figures, the edges of the Voronoi diagram are in **dark grey** and example paths generated by our approach are displayed with **black** edges.

The normalized cumulative gain $nCG_k$ is often used to measure the goodness of the ranking results returned by a web search engine algorithm. It computes how far is the candidate ranking set from the ideal ranking set. It is based on the definition of relevance ($rel$) of a single path. In our case the relevance is defined as the inverse of the path cost.

$$CG_k = \sum_{k}^{K} rel_k$$

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

To paths with smaller costs correspond higher values of cumulative gain. $nCG_k$ is normalized by the maximum cumulative gain of $k$ best paths generated by Yen's.

### D. Parameters

In the algorithm only one parameter need to be set: the number of homotopy classes to find. For Yen and RHCF algorithms we find the first 5 paths or homotopy classes (K=5). The parameters of the cost function after several informal validations have been set to (see Fig.3):

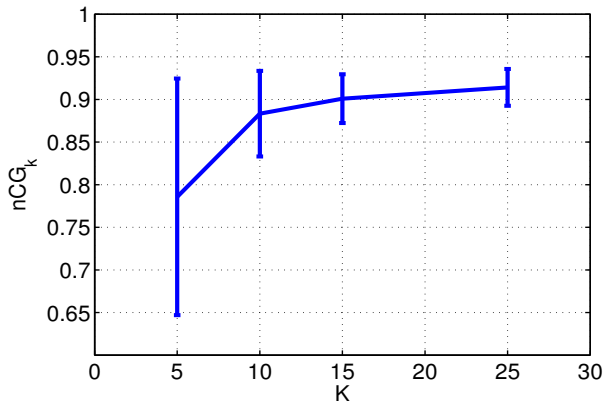


Fig. 5. Normalized Cumulative Gain $\mathbf{nCG}_k$ obtained by varying $K$ in the *crowd A* scenario. As $K$ increases, our approach converges to the optimal value of the normalized cumulative gain $\mathbf{nCG}_k$.

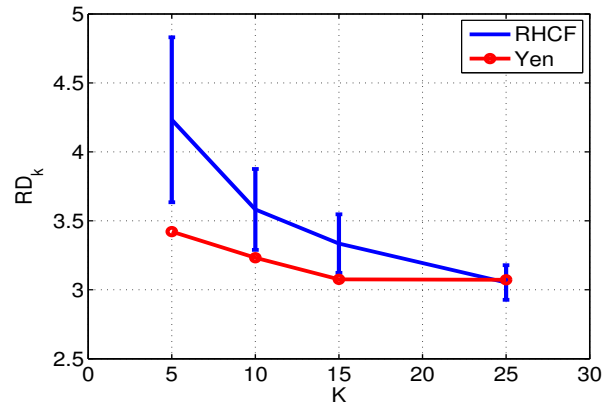


Fig. 6. Robust diversity $\mathbf{RD}_k$ obtained by varying $K$ in the *crowd A* scenario. The paths produced by our approach are more diverse than the one generated by Yen's for small values of $K$.

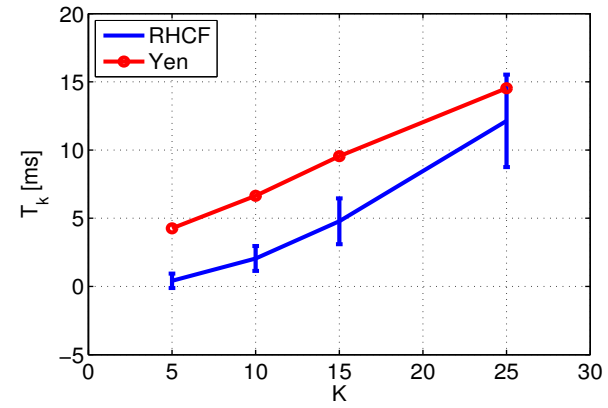


Fig. 7. Planning time $\mathbf{T}_k$ obtained with different values for $K$ in the *crowd A* scenario. For small values of $K$, our approach is faster than Yen's algorithm in very complex scenarios (with hundreds of different homotopy classes).

| $\mathbf{RD}_k$ | | |
|---|---|---|
| **Scenarios** | RHCF | Yen |
| Crowd A | **3.76813** | 2.79823 |
| Crowd B | **4.23284** | 3.4217 |
| Wall of People | 3.7924 | **5.19881** |
| Surrounded | **3.76813** | 2.79823 |

TABLE I
ROBUST DIVERSITY RESULTS

| $\mathbf{nCG_k}$ | |
|---|---|
| **Scenarios** | RHCF |
| Crowd A | 0.7857 |
| Crowd B | 0.7142 |
| Wall of People | 0.771 |
| Surrounded | 0.7461 |

TABLE II
CUMULATIVE GAIN RESULTS

magnitude of social force to pedestrians $a_j$ to 2, range of social force to pedestrians $b_j$ to 1, strength of the anisotropic factor $\lambda$ to 0.1, $r_{ij}$ equal to 0.4 $m$.

## IV. RESULTS AND DISCUSSION

Table I, Table II and Table III collect the results generated for all the described scenarios, considering the number of homotopy classes to find equal to five ($K = 5$). Table III shows the planning time results: our approach is at least five times faster than Yen's approach. Table II details the results related to the cumulative gain, *RHCF* while being faster than Yen's it also finds homotopy classes with a gain close the optimal one (i.e 1): the costs of the solutions generated by the two approaches are only slightly different, the solutions' quality of the two approaches are very similar. Table I details the diversity of the paths generated by both approaches: our method outperforms Yen's in three environments of the four. Only in one scenario, *wall of people*, Yen's finds more diverse paths.

Fig.5-7 show the metrics trends for different values of $K$ in the *crowd A* scenario (the same trends are visible in the other scenarios too). For small values of $K$, our approach is faster than Yen's algorithm in very complex scenarios (with hundreds of different homotopy classes), as it is showed in Fig.7. In average, when $K$ is greater than one fourth of the total possible homotopy classes in the scenario, the algorithm is slower than Yen's. Our approach has a better robust diversity $RD_k$, considering $K$ up to 25 see Fig.6: the paths produced are more diverse than the one generated by Yen's for small values of $K$. With a higher value for $K$, see Fig.5, our approach converges to the optimal value of the normalized cumulative gain $nCG_k$ values, the one associated to Yen's rankings.

## V. CONCLUSIONS

In this paper, we introduce the Randomized Homotopy Classes Finder, that finds homotopy classes in

| $\mathbf{T_K}$ $[ms]$ | | |
|---|---|---|
| **Scenarios** | RHCF | Yen |
| Crowd A | **0.41** | 4.26 |
| Crowd B | **1.53** | 4.42 |
| Wall of People | **0.035** | 2.07 |
| Surrounded | **1.05** | 5.61 |

TABLE III
PLANNING TIME RESULTS

an undirected weighted graph built from a Voronoi diagram. We use the algorithm to find a set of $k$ distinct socially-aware paths from which the robot chooses the best one to follow in terms of a social cost function. Our experimental evaluation shows that our approach is faster than Yen's approach. Moreover, as the cumulative gains results show, the paths produced by our approach are not far from the ones generated by Yen's that finds the true $k$ best paths. A key property is that our approach computes a set of more diverse paths respect to the baseline: usually different paths share few edges which make them robust to invalidation due to unexpected obstacles.

In future work, we intend to further improve the time performance of the *RHCF* by introducing a discounting factor that biases the search towards a not frequently visited subset of the state space, therefore increasing the probability to generate paths not yet found. Moreover we are interested to couple our approach with an informed (weighted) Voronoi diagram that implicitly encodes the social context of the scene. Finally, we plan to integrate our algorithm in a hierarchical framework where an optimal sampling-based motion planner generates (locally) optimal kinodynamic trajectories in the best homotopy class found by *RHCF*.

## ACKNOWLEDGEMENTS

The authors thank Markus Kuderer and Christoph Sprunk for valuable discussions and feedback. This work has partly been supported by the European Commission under contract number FP7-ICT-600877 (SPENCER)

## REFERENCES

[1] R. Triebel, K. Arras, R. Alami, L. Beyer, S. Breuers, R. Chatila, M. Chetouani, D. Cremers, V. Evers, M. Fiore, H. Hung, O. A. I. Ramírez, M. Joosse, H. Khambhaita, T. Kucner, B. Leibe, A. J. Lilienthal, T. Linder, M. Lohse, M. Magnusson, B. Okal, L. Palmieri, U. Rafi, M. van Rooij, and L. Zhang, "Spencer: A socially aware service robot for passenger guidance and help in busy airports," in *Proc. Field and Service Robotics (FSR)*, 2015.

[2] N. J. Nilsson, "Problem-solving methods in artificial intelligence," 1971.

[3] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," in *Int. Conf. on Robotics and Automation (ICRA)*, Detroit, USA, 1999.

[4] L. E. Kavraki, P. Švestka, 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, no. 4, pp. 566–580, 1996.

[5] S. Koenig and M. Likhachev, "D* lite." in *AAAI/IAAI*, 2002, pp. 476–483.

[6] D. Demyen and M. Buro, "Efficient triangulation-based pathfinding," in *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*, 2006.

[7] S. Eriksson-Bique, J. Hershberger, V. Polishchuk, B. Speckmann, S. Suri, T. Talvitie, K. Verbeek, and H. Yıldız, "Geometric k shortest paths," in *26th Symposium on Discrete Algorithms (SODA)*. SIAM, 2015.

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

[9] S. Bhattacharya, V. Kumar, and M. Likhachev, "Search-based path planning with homotopy class constraints," in *Third Annual Symposium on Combinatorial Search*, 2010.

[10] M. Kuderer, C. Sprunk, H. Kretzschmar, and W. Burgard, "Online generation of homotopically distinct navigation paths," in *Int. Conf. on Robotics and Automation (ICRA)*, 2014.

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

[12] P. Vela, A. Vela, and G. Ogunmakin, "Topologically based decision support tools for aircraft routing," in *Digital Avionics Systems Conference (DASC), 2010 IEEE/AIAA 29th*, 2010.

[13] C. Voss, M. Moll, and L. E. Kavraki, "A heuristic approach to finding diverse short paths," in *Int. Conf. on Robotics and Automation (ICRA)*, 2015.

[14] D. Eppstein, "Finding the $k$ shortest paths," *SIAM J. Computing*, vol. 28, no. 2, 1998.

[15] E. Hernandez, M. Carreras, and P. Ridao, "A comparison of homotopic path planning algorithms for robotic applications," *Robotics and Autonomous Systems*, vol. 64, no. 0, 2015.

[16] K. D. Jenkins, "The shortest path problem in the plane with obstacles: A graph modeling approach to producing finite search lists of homotopy classes, master's thesis," in *Monterey, California, Naval Postgraduate School*, 1991.

[17] A. W. Brander and M. C. Sinclair, "A comparative study of k-shortest path algorithms," *Proc. of 11th UK Performance Engineering Workshop*, 1996.

[18] T. Linder and K. O. Arras, "Multi-model hypothesis tracking of groups of people in RGB-D data," in *IEEE Int. Conf. on Information Fusion (FUSION'14)*, Salamanca, Spain, 2014.

[19] M. Luber, G. D. Tipaldi, and K. O. Arras, "Place-dependent people tracking," *International Journal of Robotics Research*, vol. 30, no. 3, March 2011.

[20] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[21] D. Vasquez, B. Okal, and K. O. Arras, "Inverse reinforcement learning algorithms and features for robot navigation in crowds: an experimental comparison," 2014.

[22] T. Kruse, A. Kirsch, H. Khambhaita, and R. Alami, "Evaluating directional cost models in navigation," in *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*. ACM, 2014, pp. 350–357.

[23] E. Q. Martins and M. M. Pascoal, "A new implementation of Yen's ranking loopless paths algorithm," *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, vol. 1, no. 2, pp. 121–133, 2003.