Multi-agent rendezvous on street networks

Malika Meghjani and Gregory Dudek

***Abstract*— In this paper we present an algorithm for finding a distance optimal rendezvous location with respect to both initial and target locations of the mobile agents. These agents can be humans or robots, who need to meet and split while performing a collaborative task. Our aim is to embed the meeting process within a background activity such that the agents travel through the rendezvous location while taking the shortest paths to their respective target locations. We analyze this problem in a street network scenario with two agents who are given their individual scheduled routes to complete with an underlying common goal. The agents are allowed to select any combination of the waypoints along their routes as long as they travel the shortest path and pass through the same potential rendezvous location. The total number of path combinations that the agents need to evaluate for the shortest path increases rapidly with the number of waypoints along their routes. We address this computational cost by proposing a combination of Euclidean and street network distances for a trade-off between the number of queries and a distance optimal solution.**

I. Introduction

This paper examines multi-agent rendezvous problem in real-world environments, and specifically rendezvous in urban environments where mobility constraints must be taken into consideration. The key problem is to allow agents to execute self-selected trajectories defined by a set of waypoints, while also allowing them to meet (i.e. rendezvous) at some point during their travels. We propose an energy efficient rendezvous algorithm that combines this meeting process with the background activity which is represented by the sequential visitation of the waypoints. Examples of this kind of optimization are often encountered in everyday life, when a person would like to meet a friend on their way from office to home, or more hypothetically in the future where industrial robots are organizing the warehouse and would like to meet each other for load balancing, or when automated taxis need to load balance passengers. We propose a formalism for this problem, along with an algorithmic approach to solving it in a particular context, and then evaluate it using a web-based application framework that we have developed.

Our particular representation of the problem is characterized by a pair of agents that wish to meet, a sequence of waypoints for each agent, and an underlying road network that expresses the allowable paths which the agents may execute. While in prior work [1], we and others have examined this kind of problem using highly idealized topological models, in the present work, we examine an instance of the problem where the road network and navigation constraints are expressed by an actual traversability database for urban road networks embedded in a geographic information system (GIS), such as the one used by OpenStreetMap [2], Microsoft's Bing maps [3] or Google maps [4]. In this context, we seek not only to optimize the performance goal of the navigation problem, but we also seek to optimize the number of queries to the database server that we use for planning. In practice, query cost can be a significant factor in terms of both real time delays and actual costs since such queries are made using slow and costly cellular networks.

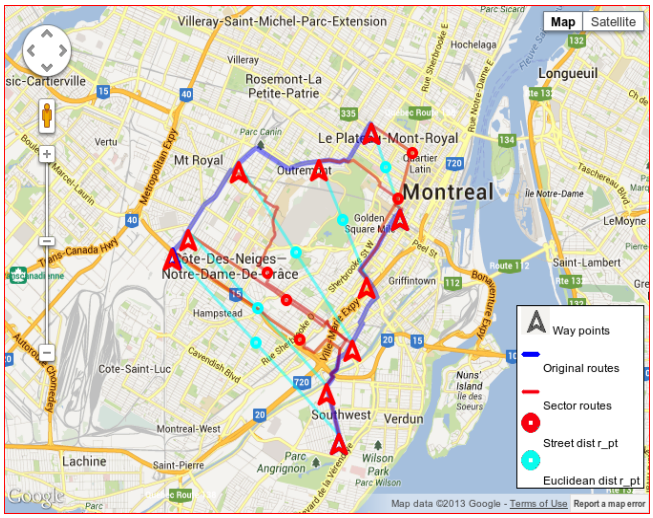


Fig. 1: An instance of rendezvous planner using Google maps API ©. The blue solid lines on the map indicates individual agent trajectories, red arrows indicate waypoints, cyan circles indicate rendezvous points computed using Euclidean metric and red circles indicate rendezvous points generated using actual travel times on the road network.

Thus, our problem entails three different classes of performance criteria, namely waypoint visiting efficiency, rendezvous efficiency and database query efficiency. The waypoint visiting efficiency can be seen largely in terms of the length of the path that covers the waypoints and skips as few of them as possible (with some skipping of waypoints permitted if a penalty is incurred). The rendezvous efficiency includes the consideration that both agents should arrive at the rendezvous point at about the same time in order to minimize waiting by one or the other (assuming that the first to arrive waits for the second agent). Lastly, the database query efficiency relates to the number of individual queries that need to be posed across the Internet. We will particularly address the waypoint efficiency and the database query efficiency in this paper.

The authors are with the Centre for Intelligent Machines, McGill University, Montréal, Québec, Canada.
email:{malika, dudek}@cim.mcgill.ca

II. Related Work

The multi-agent rendezvous problem was first introduced as a search problem in game theory by Alpern et al. [5] who extensively studied this problem on simple world models (e.g. lines, circles and polygons) and in metric environments [6]. The transition of this problem from theoretical environments to real world scenarios was addressed by Dudek et al. in [7]. Specifically they proposed rendezvous strategies for two robots to physically meet each other and share maps while exploring unknown environments. Our previous work [1], [8] is motivated by their rendezvous strategies for exploring random graphs for multiple agents without any prior knowledge. In the same work, we also introduced a cost efficient ranking criteria for combining exploration with rendezvous. The problem discussed in this paper however, assumes a known world which helps in planning the rendezvous off-line.

The prior work in multi-agent rendezvous literature finds the rendezvous locations only with respect to the initial locations and totally neglect the background activity and the individual target locations of the agents. One such example is the energy efficient rendezvous algorithm as proposed by Zebrowski et al. in [9] for a group of heterogeneous robots. The goal of this work was to minimize the total cost of traveling to the rendezvous location. The authors proposed a heuristic in which the local heading of individual robots is iteratively computed, based only on the starting location of the other robots. This solution was empirically shown to be near optimal by comparing it against the globally optimal solution.

An extension of the previous problem was addressed by the same authors in [10]. This work, discusses the problem of finding a set of meeting places for a tanker robot to rendezvous individually with multiple worker robots such that they can recharge with minimal energy expenditure. This problem is analyzed in two parts, (a) finding optimal rendezvous locations for meeting each robot and (b) obtaining an optimal order for visiting them. It was shown that the combinatorial problem of finding an optimal visiting order and meeting locations together is NP-hard. Therefore, the authors consider that the visiting order is given and solve for finding optimal meeting locations only using a numerical solution to sequentially resolve the optimal meeting locations for each robot. This solution is a generalized version of the *facility location problem*.

A similar application of using multi-agent rendezvous for charging mobile robots was addressed by Waslander et al. in [11]. The goal of this work was to plan routes for charging ground robots given the trajectory of the UAV working robots. This problem was formulated as a directed acyclic graph with vertex partitions containing sets of charging points where rendezvous can potentially occur for each working robot. The authors proposed a heuristic strategy that involves transforming the graph problem to a TSP and then solving a mixed integer linear program using a heuristic solver.

One of the related works in the context of route optimization on large GIS systems was proposed by Bast et al. in [12] where both routing and data access are relevant constraints (without considering the rendezvous process itself). Yan et al. [13] and Papadias et al. [14] worked on the joint problem of route optimization and rendezvous in large GIS systems and our work is motivated by their algorithms.

The former work, discusses an optimal meeting point algorithm on street networks applicable in the scenario where a tourist bus is deciding on an optimal location to pick up passengers who are at different locations. The optimal meeting point is then selected based on the minimum sum distance criterion. The target locations of the passengers are however, not taken into consideration. In the latter case, the target locations are considered and it is similar to the problem where a group of friends would like to meet for dinner and each one has a different restaurant preference. The proposed algorithm then selects a rendezvous location which minimizes the sum of distances to all the points from a given set of target locations. This problem is similar to our work with the only difference that our meeting locations are not the same as the target locations of the agents instead they are the midpoints of the agent locations along their routes.

III. Problem Formulation

We consider the following scenario for analyzing our rendezvous problem. Two agents, A_1 and A_2, are assigned two paths, $U(t)$ and $V(t)$ respectively, to follow as part of their task. Let $U(t) = \{u_1, u_2, ..., u_n\}$ and $V(t) = \{v_1, v_2, ..., v_n\}$ be the discrete representation, as sequences of waypoints, of these two paths. Let $R = \{r_1, r_2, ...r_n\}$ be the set of potential rendezvous locations. These locations are the midpoints of the waypoints that define the two paths. The agents are initially located at u_1 and v_1 and their desired target locations are u_n and v_n. We design our problem such that the agents depart from their original paths at locations u_k and v_k, at the same time, attempt to meet at a rendezvous point r_i and return to their respective routes at some waypoints, u'_k and v'_k. The total distance traveled by the two agents from their source to target locations is required to be minimized with respect to the rendezvous point. A graphical illustration of our problem is given in Figure 2.

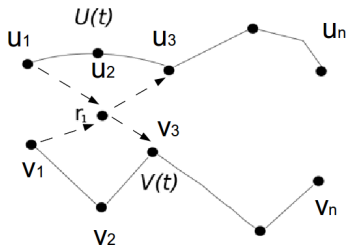


Fig. 2: $U(t)$ and $V(t)$ are the original paths of the agents with $\{u_1, v_1\}$ and $\{u_n, v_n\}$ as the source and target locations. In this example r_1 is the rendezvous location corresponding to the shortest path: $\{u_1, r_1, u_3, ..., u_n\}$ for agent A_1 where the waypoint u_2 is skipped.

For any particular agent, given a rendezvous location r_i, and a task path that consists of n waypoints, the number of possible paths that visit the rendezvous point is $\frac{n(n-1)}{2}$. This is explained by exhaustively considering all possible combinations of the rendezvous points and number of waypoints that the agent skips during its route. The agent can either choose to visit all of its waypoints and there will be $(n-1)$ possible paths to choose from or consider skipping one waypoint with $(n-2)$ options and so on until skipping all the waypoints with only one possible path from the source u_1 to rendezvous point r_i and returning to the original path at final waypoint u_n. Hence, the total number of paths considering one agent and all the potential rendezvous locations is $O(n^3) = \frac{n(n-1)}{2} * |R|$.

We would like to find the rendezvous point r^* that minimizes the total path length among all the combinations for the rendezvous points r_i where $i \in \{1, n\}$ and number of skips $j \in \{0, (n-1)\}$ i.e.

$$r^* = \underset{r_i}{\operatorname{argmin}} \quad D(r_i, j) \tag{1}$$

where $D(r_i, j)$ is a function that returns $p^*_{i,j}$ which is the minimum path length for a given rendezvous point r_i and number of skips j.

$$p^*_{i,j} = \min_k \quad p_{i,k} \tag{2}$$

$$p_{i,k} = b_{i,k} + b_{i,(j+k+1)} + d(u_1, u_k) + d(u_{(j+k+1)}, u_n) \tag{3}$$

where $k \in \{1, (n-1-j)\}$, is the waypoint index at which the agent takes a detour from its path, d is the distance function, $b_{i,k}$ represents the *bridge distance* between the rendezvous point r_i and waypoint u_k. In our formulation, the rendezvous points are selected as midpoints of the two paths and therefore, the *bridge distances* are symmetric for the two agents as given in Equation (4).

$$b_{i,k} = d(r_i, u_k) = d(r_i, v_k) \tag{4}$$

The cost function optimized in our above problem formulation is purely based on the distance measurements which does not account for the number of waypoints covered along the way. This can be incorporated by associating a reward with every waypoint and then optimizing the solution by using a weighted sum of distances and rewards. Similarly, the rendezvous locations can be prioritized based on the number of interesting locations around it and the order in which they occur along the agents' routes.

IV. Proposed Approach

We analyze our problem on street networks using the Google maps API [15]. We presuppose that the starting and ending points for each agent are given, along with a sequence of waypoints of interest between these points. In our actual implementation, we allow the user to select the starting and ending locations manually, and can synthesize an illustrative set of waypoints using either domain specific features (such as restaurants), uniform subdivision, or pre-specified locations based on a task of interest. In this paper, however, we consider the more idealized case where the waypoints are generated automatically by partitioning the most efficient route from start to end into uniform segments. This case where the waypoints arise naturally from starting to ending locations is inherent in some domains (e.g. highways) and is easier to evaluate experimentally than paths that are generated with a domain specific selection of waypoints, such as drop off points for deliveries in a courier task.

Once the waypoints are obtained for the two agents, we make their path lengths equal by truncating the longer path. This is justified by the requirement that that two agents need to meet before reaching their target locations. Next, we calculate a street network path between uniform segments of the two routes and we find the midpoint along this path. This midpoint is considered as one of the potential meeting locations. The process is repeated for all the segments along the path to obtain a list of potential rendezvous locations. For simplicity, we consider that the agents have the same speed. Also, to reduce the number of directional queries to the server we consider only the walking directions which are bidirectional and hence symmetric for the two agents.

Given the original routes divided into segments and the potential rendezvous locations, we exhaustively list $O(n^3)$ combinations of possible paths for one agent at a time as explained in Section III. Since the number of combinations increase very fast with number of waypoints, we propose a smart query reduction method. According to this method we only require n^2 queries to the server to enumerate all the $O(n^3)$ path lengths for one agent. Specifically, we query only the distances between the waypoints $U(t)$ and rendezvous points R as described in Equation (4). We combine these distances with the segment lengths from the original routes to obtain the total path length by parts for all the combinations.

Table I: Asymptotic bounds for worst case queries

Queries			
Total	Smart	Euclidean	Hybrid
$O(n^3)$	$O(n^2)$	0	$O(n^2)$

A further reduction in the number of queries can be achieved by considering an inherent property of the street networks. On the street networks, the Euclidean distances provide a lower bound on the street network distances. We leverage this fact and propose a hybrid combination of Euclidean and street network distance measures for calculating the path lengths. Specifically, we obtain a list of Euclidean distances for all the *bridge paths* (paths towards and away from rendezvous point as in Equation (4)) and arrange them in ascending order. The *bridge path* corresponding to the shortest path in Euclidean distance is re-evaluated for street network distance. If the street network distance is still the minimum in the list of *bridge path* lengths then the rendezvous location along this path is selected. Otherwise, the process is repeated until the shortest *bridge paths* are in street network distances. This process can reduce the number

of queries to as low as 2 for two *bridge distances* ($b_{i,k}$, $b_{i,(j+k+1)}$) in the best case and less than twice the number of smart queries in worst case. The asymptotic number of queries in worst case for all the methods is presented in Table I and a summary of our hybrid approach is presented in Algorithm 1.

- **Initialization**: endpoint selection:$\{u_1, u_n, v_1, v_n\}$, waypoint selection *(optional)*
- **Rendezvous selection**: generation of possible rendezvous points $R = \{r_1, r_2, ..., r_n\}$
- **Query synthesis**: output the shortest path, $p^* = \{u_1, ..., r_i, ..., u_n\}$ using hybrid queries as described below:
  - Find all path lengths in Euclidean space $P = \{p_{ik}\}$ with $i \in \{1, n\}$ and $k \in \{1, (n-1-j)\}$
  - Sort them in ascending order $Sort(P)$
  - If the shortest path $p^* = P[0]$ is not expressed in street network distance then make it so
  - Else p^* is the real optimal path and $r^* = r_i$ is the distance optimal rendezvous location
- **Path execution**

Algorithm 1: Hybrid algorithm

V. Experimental Results

We evaluated our algorithm on urban street network maps from 10 different cities around the world. The source and target locations for two agents, were manually selected in and around the city centers. The results are compared based on Euclidean distances, street network distances and hybrid combination of Euclidean and street network distances. The criteria for comparison are the shortest path length from source to target location passing through rendezvous location, number of queries made to the server and whether the suggested Euclidean distance based rendezvous location is accessible. A summary of the results is given in Table II and some of the graphical representations of the results are presented in Figures 3, 4 and 5.

We observed that the Euclidean based measures were a close approximation to the street network path lengths when used for calculating distances between the waypoints and the rendezvous locations as illustrated in Figure 3. This property of the street networks allowed us to significantly reduce the total number of smart queries to the server as observed in Table II.

The reduction in total number of queries can be compared using the data in the queries column of Table II. This column provides information regarding, the total number of possible paths between source and target locations for one agent via different rendezvous points which is $O(n^3)$, the number of smart queries which is fixed at n^2 and the number of street network distance queries for the hybrid algorithm along with number of queries saved by using the hybrid method instead of the smart queries. The hybrid algorithm had lesser queries than smart queries in all but one case and always provided the same length of the path as obtained by smart queries on street network distances. The average fraction of queries saved using the hybrid algorithm in 10 scenarios, that we illustrated is 40%.

An anomalous case can be observed for Hong Kong city, where the shortest path was obtained at the cost of larger number of queries than the smart queries but whithin its worst case $O(n^2)$. This result can be explained by the forest regions where there are no street network paths but Euclidean distances underestimate the street network path lengths. Another disadvantage of the Euclidean distance based measurements is that the rendezvous locations may not be accessible in some cases as illustrated in Figure 4. The rendezvous location for the shortest path in each of these scenarios are in the middle of a park, forest and lake respectively. In addition, as the original path lengths of the routes from source to target location increases, the Euclidean based measures deviate more from the street network distances as observed in the cases: Singapore (Fig.3(c)) and Mexico city.

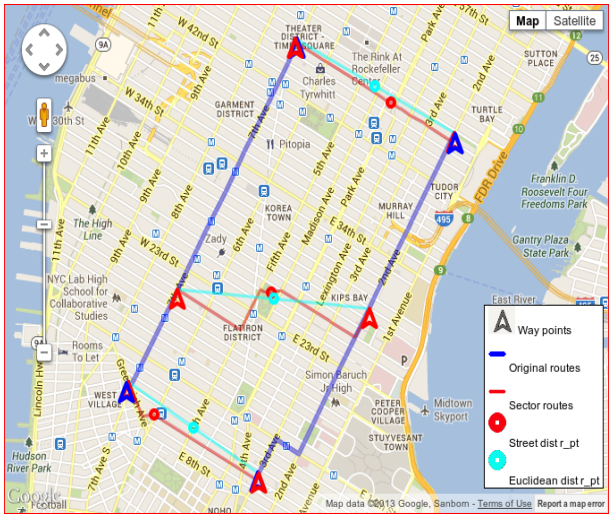
The results reported in this section consider the agents to be walking pedestrians. We also evaluated our algorithm in driving mode for one of the special scenarios in Brussels, as presented in Figure 5(a), where most of the roads are unidirectional. In this case, we can observe that though the path lengths were relatively short the street network distances were largely underestimated by the Euclidean measures. Another special case was evaluated in Paris city across a strait where both the Euclidean and street network distance based rendezvous points were relatively near each other but all the Euclidean rendezvous points were suggested in the water while all the street network rendezvous points were suggested across the bridge.

VI. Discussions and Conclusions

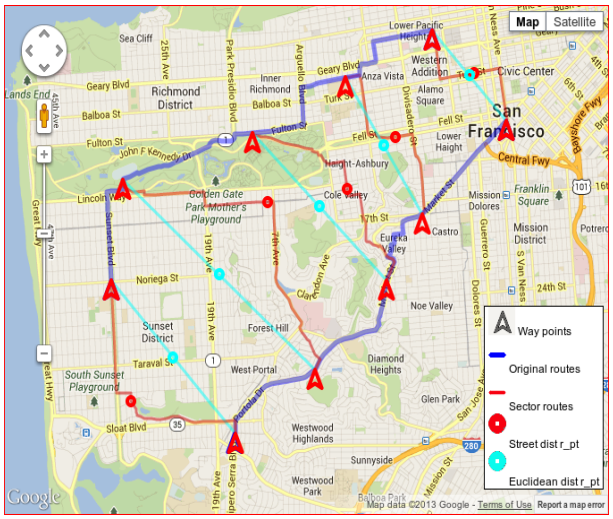
This paper provides a proof of concept for a novel multi-agent rendezvous problem in street networks. The previous work for mobile agent rendezvous has been very focused on idealized problems that, while illuminating, cannot be readily applied in practice. On the other extreme, the real world applications for human-human rendezvous using smart phones, have been very popular with the goal of gathering people to one single location. Typical practical solutions, however, fail to account for many of the constraints and specifically the joint optimization that characterizes our problem. In our work we have bridged the gap between pure theory based simulations and real applications using our web application framework. In addition, we also addressed the issue of reducing expensive query cost to the server for finding the shortest path by combining Euclidean and street network distance measurements. Our experimental results suggest that the Euclidean based measurements can provide a good first approximation of the street network distances hence making the planning cost very reasonable.

VII. Future Work

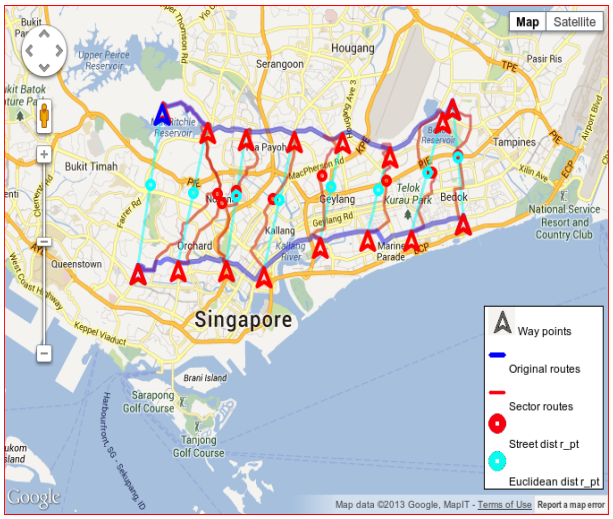
The rendezvous planning problem can potentially be formulated as an on-line problem rather than, or in addition



(a) New York



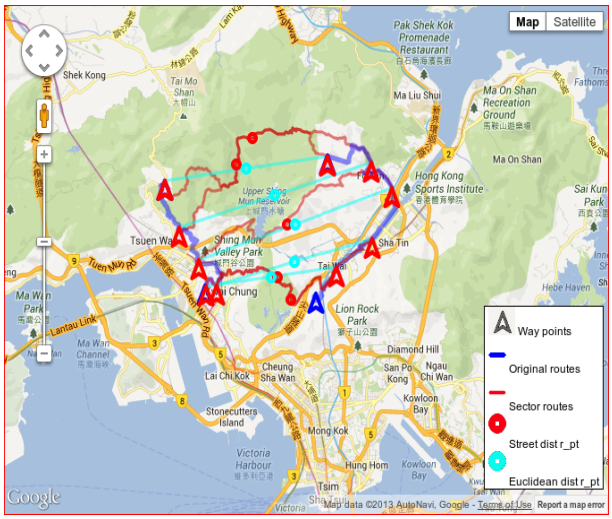
(b) San Francisco



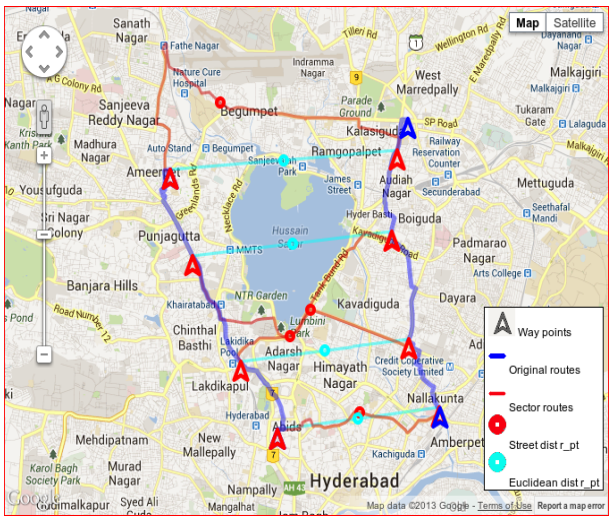
(c) Singapore

Fig. 3: Examples of good approximation of Euclidean to street network distance based rendezvous locations. Google maps API ©

(a) Montreal



(b) Hong Kong

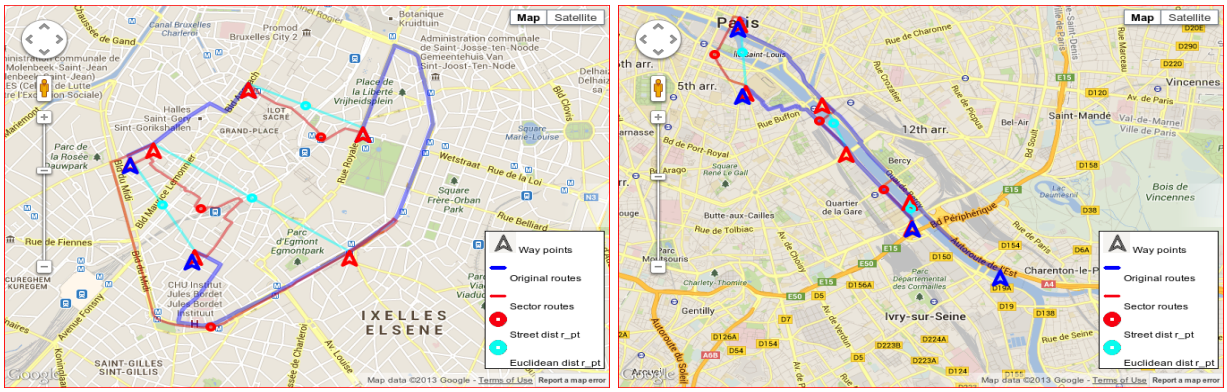


(c) Hyderabad

Fig. 4: Examples of inaccessible Euclidean distance based rendezvous locations. Google maps API ©

Table II: A comparison of distances and number of queries for Euclidean, smart and hybrid methods

Cities	Euclidean distance (km)	Real distance (km)	Hybrid distance (km)	No. of queries				Euclidean rpt accessible?
				Total	Smart	Hybrid	Saved	
Montreal	2.8	3.4	3.4	27	9	4	5	No (Park)
New York	2.9	3.3	3.3	27	9	6	3	Yes
San Francisco	6.0	7.4	7.4	125	25	17	8	Yes
Paris	7.3	8.0	8.0	64	16	6	10	Yes
Singapore	12.9	15.0	15.0	512	64	50	14	Yes
Hong Kong	6.9	11.5	11.5	125	25	31	-6	No (Forest)
Tokyo	8.3	9.2	9.2	125	25	5	20	Yes
Sydney	7.9	9.0	9.0	125	25	11	14	No (Harbor)
Hyderabad	5.5	6.9	6.9	64	16	9	7	No (Lake)
Mexico City	19.2	21.4	21.4	729	81	38	43	Yes
Average	7.9	9.5	9.5	192.3	29.5	17.7	40%	—



(a) Brussels with driving directions

(b) Paris with rendezvous locations on the bridge

Fig. 5: Examples not included in Table II. Google maps API ©

to, the off-line optimization which we have considered in this paper. The on-line planning extension can take into consideration uncertainty in traffic patterns that is only determined as the trajectory is executed, and thus can trade off the expected impact of future uncertainty against rendezvous efficiency. The on-line problem is a natural successor to the off-line base problem considered in this paper.

VIII. Acknowledgement

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) through the NSERC Canadian Field Robotics Network (NCFRN). The authors would specially like to acknowledge all the members of the Mobile Robotics Lab at McGill University for their technical discussions and motivation.

References

[1] M. Meghjani and G. Dudek. Multi-robot exploration and rendezvous on graphs. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages –. IEEE, 2012.

[2] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.

[3] Joe Schwartz. Bing maps tile system. *Microsoft Developer network Available: http://msdn. microsoft. com/en-us/library/bb259689. aspx*, 2009.

[4] Daniel J Turco. Auto routing computer for eliminating the need for maps or travel instructions, November 17 1981. US Patent 4,301,506.

[5] S. Alpern and S. Gal. The theory of search games and rendezvous. pages 165–178, 2003.

[6] A. Dessmark, P. Fraigniaud, and A. Pelc. Deterministic rendezvous in graphs. *Algorithms-ESA 2003*, pages 184–195, 2003.

[7] N. Roy and G. Dudek. Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations. *Autonomous Robots*, 11(2):117–136, 2001.

[8] M. Meghjani and G. Dudek. Combining multi-robot exploration and rendezvous. In *CRV '11: Proceedings of the 2011 Canadian Conference on Computer and Robot Vision*, pages 80–85. IEEE Computer Society, May 2011.

[9] P. Zebrowski, Y. Litus, and R.T. Vaughan. Energy efficient robot rendezvous. In *Computer and Robot Vision, 2007. CRV'07. Fourth Canadian Conference on*, pages 139–148. IEEE, 2007.

[10] Y. Litus, R.T. Vaughan, and P. Zebrowski. The frugal feeding problem: Energy-efficient, multi-robot, multi-place rendezvous. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 27–32. IEEE, 2007.

[11] Neil Mathew, Stephen L Smith, and Steven L Waslander. A graph-based approach to multi-robot rendezvous for recharging in persistent tasks. In *IEEE Int. Conf. on Robotics and Automation, Karlsruhe, Germany*, 2013.

[12] Hannah Bast, Erik Carlsson, Arno Eigenwillig, Robert Geisberger, Chris Harrelson, Veselin Raychev, and Fabien Viger. Fast routing in very large public transportation networks using transfer patterns. In *Algorithms–ESA 2010*, pages 290–301. Springer, 2010.

[13] D. Yan, Z. Zhao, and W. Ng. Efficient algorithms for finding optimal meeting point on road networks. *Proceedings of the VLDB Endowment*, 4(11), 2011.

[14] D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis. Group nearest neighbor queries. In *Data Engineering, 2004. Proceedings. 20th International Conference on*, pages 301–312. IEEE, 2004.

[15] Gabriel Svennerberg. *Beginning Google Maps API 3*. Apress, 2010.