# Selection of Rendezvous Points for Multi-Robot Exploration in Dynamic Environments

Julian de Hoog
Computing Laboratory
Oxford University
Wolfson Building, Parks Road
Oxford OX1 3QD, UK
julian.dehoog
@comlab.ox.ac.uk

Stephen Cameron
Computing Laboratory
Oxford University
Wolfson Building, Parks Road
Oxford OX1 3QD, UK
stephen.cameron
@comlab.ox.ac.uk

Arnoud Visser
Intelligent Systems Laboratory
Universiteit van Amsterdam
Science Park 107
NL 1098 XG Amsterdam,
Netherlands
a.visser@uva.nl

## ABSTRACT

For many robotics applications (such as robotic search and rescue), information about the environment must be gathered by a team of robots and returned to a single, specific location. Coordination of robots and sharing of information is vital, and when environments have severe communication limitations, approaches must be robust to communication drop-out and failure. The difficulties are compounded in dynamic environments, where paths previously believed to be free can suddenly become blocked.

In this paper, we introduce a novel way of calculating rendezvous points for robots to meet and share information. Using role-based exploration, some robots continuously explore the environment while others ferry information back and forth to a central command centre. Optimal rendezvous point selection leads to more efficient exploration, and allows robots to replan when one of them has unexpected obstacles in its path.

## Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics—*Autonomous vehicles*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence —*Intelligent agents, Multiagent systems*; I.4.10 [**Image Processing and Computer Vision**]: Image Representation—*Morphological*

## General Terms

Algorithms, Experimentation, Performance

## Keywords

Robotics, exploration, multi-robot cooperation, limited communication, search and rescue robots, role-based exploration, rendezvous points, dynamic environments

## 1. INTRODUCTION

Advances in robotics and multi-agent systems mean that robots will be used for an ever wider range of applications in the near future. Such tasks include reconnaissance, surveillance, exploration of environments inaccessible to humans (*e.g.* underwater or in space), and missions in potentially dangerous environments (*e.g.* bomb disposal or search-and-rescue).

In this paper we are particularly interested in the robotic search-and-rescue task, although our results are applicable to various robotic tasks. Search-and-rescue robots are used to explore environments after disaster scenarios (such as earthquakes) that are otherwise not accessible due to risks of secondary disaster, environmental hazards, or a lack of spatial access for humans or dogs. The hope is that robots will be able to efficiently explore and map disaster environments and find locations and statuses of human victims, so that human responders know where to focus their rescue efforts.

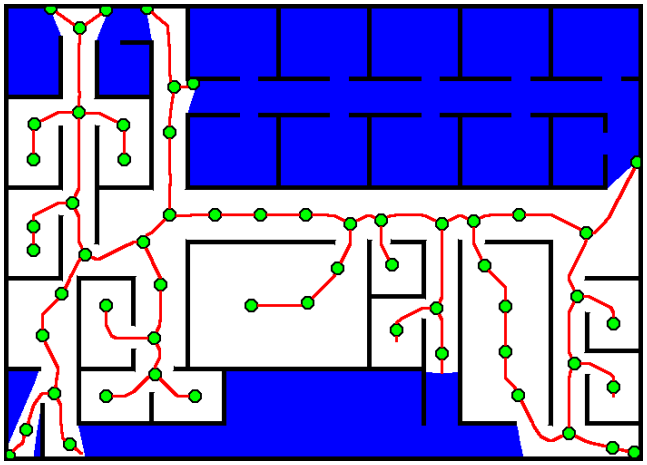


Figure 1: A partially explored environment: walls are black, unexplored space is blue, explored (free) space is white. Thinning on the free space allows for calculation of possible rendezvous points (green dots). By choosing the best rendezvous points, robots can meet to exchange information more efficiently, and can replan to meet at another rendezvous point if one of them encounters unexpected obstacles.

Currently, search-and-rescue robots are typically at least 40 - 60cm wide, long, and high, and are usually controlled directly by a human operator. Promising approaches include track-based robots, snake robots, and flying robots. As technologies improve and miniaturise, we believe that future robotic search-and-rescue efforts will involve teams of small rolling, crawling, or flying robots that autonomously explore environments of interest together. Such teams will require robust strategies for typical multi-robot team problems: team coordination, sharing of information, and limited communication.

In search-and-rescue environments, the limited communication problem is particularly relevant as disaster environments are likely to be full of obstacles and interference. Moreover, the additional problem of dynamic environments must be considered: unstable and burning rubble, for example, may well shift or change as the exploration effort unfolds.

In this paper we hope to make first steps in the direction of solving the following problem: how can a team of agents, subject to limited communication, be coordinated to (i) explore an unknown and possibly dynamic environment as quickly as possible while (ii) relaying known information back to a central command centre as quickly as possible?

Central to our approach is the use of **role-based exploration**, and the determination of optimal **rendezvous points**. In role-based exploration, team members assume one of two roles: *exploring* the far reaches of the environment, or *relaying* known information from explorers back to the command centre. To coordinate efficient meetings between explorers and relays (for information exchange), calculation of optimal rendezvous (*i.e.*, meeting) points is crucial, and can significantly speed up the exploration effort.

While our experiments are an abstraction from the real-world problem of robotic search-and-rescue, we hope that our ideas and conclusions will be applicable to future robot rescue teams, possibly as an extension of existing multi-robot exploration algorithms. The results are also applicable to other problems where information by a team of communicating robots must be consolidated at a single location, such as for example in underwater or planetary exploration.

This paper is structured as follows: In Section 2 we discuss related work. Section 3 describes in detail our approach, including how we determine rendezvous points. Our simulation framework and communication model are described in Section 4, while our experimental results are outlined in Section 5. Finally we discuss the ramifications of our work in Section 6, and conclude in Section 7.

## 2. RELATED WORK

**Multi-robot Exploration**

Multi-robot exploration has received considerable attention in recent years but only a small number of approaches have taken limited communication into account.

In early approaches, a line-of-sight constraint was used to keep robots within communication range [3, 12]. This has been extended to robots reactively choosing a direction that will most likely keep them within sight of the rest of the team [17].

Several authors propose multi-robot exploration strategies based on market principles, in which robots place bids on subtasks of the exploration effort [22, 8, 28, 21]. These bids are typically based on values such as expected information gain and travel cost to a particular location in the environment, and may be assigned in a distributed fashion among team members, or by a central agent. When strength of communication is factored into the bids, robots avoid areas outside of communication range.

Another common strategy for robotic exploration is to use frontiers [27], which can easily be extended for use by multiple robots [5, 10, 18, 24]. Similar to bids described above, utilities of individual frontiers may include a factor related to likelihood of communication success, so robots are less likely to explore areas that take them out of the team communication range.

Further approaches include the use of 'energy fundamentals' to maintain network connectivity [20], results from graph theory to keep individual robots in 'comfort zones' [23] and the application of synthetic 'spring forces' to keep robots close to one another [16].

While several of these approaches have proven successful in maintaining team connectivity during the exploration effort, they are usually limited by the constraint of having to keep team members within communication range. Even if members of a team are dispersed to the maximum extent that their communication ranges allow, in large and complex environments unexplored areas will remain.

A solution to this problem is to allow robots to autonomously explore beyond communication range limits. This can be implemented in terms of 'robot pack' or clustering behaviour, in which groups of robots stay close together as they explore the environment [18, 21, 10].

However, little work has been done towards the typical search-and-rescue problem of gathering information in a severely communication-limited environment at a single location as efficiently as possible.

**Rendezvous points**

In several robotic exploration approaches, shared knowledge communicated at meetings between multiple robots is used for multi-robot localisation, although such meetings are not explicitly planned [9, 13]. The term *rendezvous* itself was introduced by Roy and Dudek in 2001 [19]. In their approach, robots wander through the environment and choose suitable landmarks for rendezvous, returning to the most suitable at a pre-arranged time.

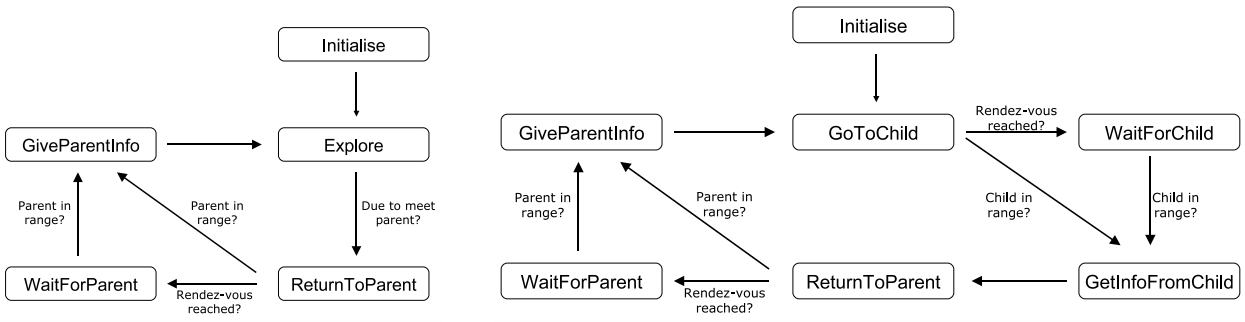
Several authors have worked on the problem of efficiently gathering multiple agents with limited visibility at a single meeting point [2, 15], but in these approaches, exploration of the environment is not a goal. The rendezvous problem has also been phrased in terms of two agents entering a known environment at separate locations and having to find one another in minimal time [1]. We do not cover this problem here, however, since we are interested in applications where agents know one another's locations at the start of the exploration effort, but the environment is unknown.

Robot rendezvous is most relevant to exploration approaches in which individual robots (or groups of robots) are out of one another's communication range for extended periods of time. Since little work has been done in this direction, rendezvous selection and use remains a young field of study.

## 3. OUR APPROACH

### 3.1 Role-based Exploration

While most frontier-based exploration approaches lead to quick and efficient exploration, they do not take into account the need to relay new information back to a central command centre in communication-limited environments. To take advantage of frontier exploration's strengths while still maintaining as well connected a robot team as possible, we propose role-based exploration. We present a brief overview here; interested readers are referred to [7] for a more thorough description.




(a) Explorers (b) Relays

Figure 2: State transition diagrams

In role-based exploration each member of the team is assigned one of two roles:

1. *Explorer.* Explorers are meant to explore the farthest reaches of the environment. To communicate their findings, they return periodically to previously agreed rendezvous points where they pass their knowledge to a relay.

2. *Relay.* Relays ferry information back and forth between explorers and the command centre. This is achieved by meeting the explorer periodically at aforementioned rendezvous points, exchanging all relevant knowledge, and then returning to the command centre. If a relay discovers information about the environment while relaying, this is added to the team knowledge, but exploration is only a by-product of the relay's movement.

## 3.2 Team Hierarchy and State Transitions

The team hierarchy is determined in advance. There may be multiple relays between the command centre and an explorer, and a relay may serve more than one explorer (see Figure 3). We are interested in dynamic team hierarchies as well, but leave this as future work for now.

State transition diagrams for Explorers and Relays are presented in Figure 2. Explorers' `GiveParentInfo` state (which coincides with Relays' `GetInfoFromChild` state) is particularly relevant to this paper: it is in this state that an Explorer plans next exploration steps, recalculates possible rendezvous points, and tells his parent relay where to rendezvous next.

Note that an Explorer and Relay do not need to reach rendezvous to transition to the next state. If there is a chance meeting between the two earlier than expected, it is advantageous to replan at that moment, rather than wait until both reach rendezvous.

## 3.3 Teammate Modeling

When two teammates, an Explorer and a Relay, meet, they exchange all relevant knowledge of the environment. After exchange, each robot will have the same map, and know exactly what its teammate knows at that point in time. Since relays' movement is highly predictable and both robots use the same path planner, the Explorer can calculate exactly how long the Relay will need to return to the Command Centre (or its parent relay), turn around, and make its way back to the next jointly agreed rendezvous point. Thus the Explorer knows exactly how much time it has to continue exploring before having to turn around and rendezvous once again, and subsequent meetings can be timed in such a manner that neither Relay nor Explorer waste time waiting for the other to return to the rendezvous point – both should reach the rendezvous point at almost the same time.

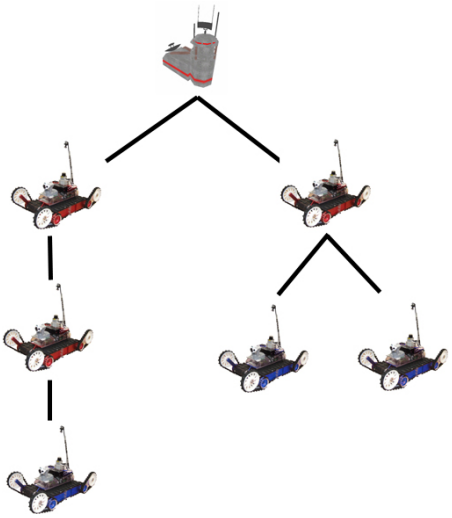


Figure 3: A possible hierarchy for role-based exploration. Explorers are blue, relays are red. The command centre (top) is the root of the hierarchy tree.

Moreover, if the Explorer stores the map exchanged at rendezvous separately from its own evolving map, then it can at any point predict the Relay's likely position, even when not in communication range (since the Relay's map is unlikely to change much). Explorer and Relay can also agree on fallback rendezvous points, in case the preferred rendezvous point can unexpectedly not be reached. This has significant implications for rendezvous in dynamic environments, discussed in more detail in section 3.6.

## 3.4 Frontier Assignment

Assuming that the team hierarchy has been determined and each robot assigned a role, how does exploration actually take place? For this, we apply simple frontier exploration [27], which is among the most popular and promising

approaches today. Frontier exploration is heavily influenced by how utilities are calculated for individual frontiers. For every frontier $f$ we calculate a utility $U(f)$ as follows:

$$U(f) = A(f)/C^n(f)$$

where $A(f)$ is the area of frontier $f$, $C(f)$ is the path cost from the robot to that frontier, and exponent $n$ determines the exploration behaviour. High values of $n$ lead to exploration of nearby frontiers (such as rooms) whereas low values mean that robots are more likely to pursue larger frontiers (such as hallways) [25]. For experiments reported later in this paper we use $n = 2$.

An additional consideration is that it is undesirable to send two robots into the same frontier. Elsewhere segmentation and the Hungarian method have been proposed [26], but we use a simple agent-frontier assignment algorithm detailed in [25]; in short, every robot determines frontier utilities for itself and its nearby teammates, and iteratively calculates a robot to frontier assignment that maximises joint utility. While this method is not necessarily optimal, it is fast, and in our experience entirely sufficient for distributed exploration.

## 3.5 Selection of Rendezvous Points

It turns out that this rendezvous point selection is an important factor in the exploration effort, and good rendezvous point selection both drives the exploration effort deep into the environment while minimising time required to communicate information up the communication chain. In our previous work, the Explorer stored its own current location at the moment that it turned to meet a Relay for use as the following rendezvous point. This did lead to deeper and deeper exploration of the environment, as with each rendezvous the Relay had to come deeper and deeper into the environment to meet the Explorer. However, in certain circumstances, rendezvous points chosen in this manner were less than favourable and led to inefficiencies (for example, when an Explorer chose a rendezvous point in an already fully explored part of the environment and had to backtrack unnecessarily to meet the Relay).

Here we propose a novel approach: subsequent rendezvous is calculated by the Explorer while it is in communication range of the Relay, and uses thinning on the free space in the map. Thinning is a technique from digital image processing that is meant to reduce a shape to its skeleton by making the shape as thin as possible while keeping it connected and centred. There are many parallels between thinning, skeletonisation, and Voronoi diagrams. A wide range of thinning techniques have been proposed since the 1960's, having various advantages or disadvantages (for a review, see [14]).

In our approach we use Hilditch's algorithm [11][1], since it is fast, returns a connected skeleton, and is easy to implement. A typical skeleton calculated using Hilditch's algorithm is presented in Figure 1.

Hilditch's algorithm requires the calculation of a *neighbour traversal* function $T(p_1)$, described in Figure 4. This function can also be used to find junction points in the skeleton: any point $p_1$ that is a junction in the skeleton will have $T(p_1) \geq 3$. A skeleton may contain long stretches without junction points, for example along a hallway – to fill out the resulting graph, we iterate over all points in the skeleton and add those that are a minimum distance from all existing rendezvous points (*filling*). On the other hand, complex parts of the environment may contain a large number of junction points in a small area – to simplify calculations we choose only one point per given density (*pruning*). This gives a nice set of possible rendezvous points, distributed fairly evenly over the known environment and including all junctions. The full algorithm for rendezvous point calculation is presented in Algorithm 1.

[1]For a useful tutorial on how to implement Hilditch's algorithm, see `http://cgm.cs.mcgill.ca/~godfried/teaching/projects97/azar/skeleton.html`.

| | | |
|---|---|---|
| $p_9$ | $p_2$ | $p_3$ |
| $p_8$ | $p_1$ | $p_4$ |
| $p_7$ | $p_6$ | $p_5$ |

Figure 4: Traversal function $T(p_1)$ is the number of 0,1 patterns in the sequence $p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_2$

```
List skeletonPoints = hilditchThinning(map);
List rendezvousPoints = new List;
foreach sp ∈ skeletonPoints do
    if neighbourTraversal(sp) ≥ 3 then
        rendezvousPoints.add(sp);
    end
end
foreach sp ∈ skeletonPoints do
    boolean addToList = true;
    foreach rp ∈ rendezvousPoints do
        if sp.distanceTo(rp) < threshold T1 then
            addToList = false;
            break;
        end
    end
    if addToList then
        rendezvousPoints.add(sp);
    end
end
foreach rp1 ∈ rendezvousPoints do
    foreach rp2 ∈ rendezvousPoints, rp2 ≠ rp1 do
        if rp1.distanceTo(rp2) < threshold T2 then
            rendezvousPoints.remove(rp1);
        end
    end
end
Return rendezvousPoints;
```

**Algorithm 1**: Calculation of rendezvous points.

Now that we have a list of potential rendezvous points, which is the best one? We examined a number of different utilities and combinations thereof: estimated communication range at the rendezvous point, proximity to nearest frontiers, and path cost. Since we want the Relay to follow the Explorer, however, it turned out that the most important consideration is the Explorer's next choice of frontier. In other words, placing the next rendezvous deep into the next frontier that the Explorer plans to enter, while ensuring that the rendezvous point has a strong communication range, gave the best results. (A large communication range is a desirable characteristic for a rendezvous point since as two robots approach it, they will be able to detect and com-

municate with one another earlier. Communication range at a particular point can be easily estimated using the communication model described in section 4.2).

More specifically, in our implementation we choose a rendezvous point by considering only a small number of points near the Explorer's next frontier of choice and choosing the one having highest neighbourTraversal value (since this is the most important junction). If multiple points have equal neighbourTraversal values, we choose the one with the best estimated communication range. The full process is outlined in Algorithm 2.

```
List rendezvousPoints = rvCalculation (Algorithm 1);
List chosenPoints = new List;
Point frontierCentre =
agent.chosenFrontier.getCentre();
int highestDegree = 0;
foreach rp ∈ rendezvousPoints do
    if rp.pathCost(frontierCentre) < threshold T3 then
        if rp.degree > highestDegree then
            chosenPoints = new List;
            chosenPoints.add(rp);
            highestDegree = rp.degree;
        end
        else if rp.degree = highestDegree then
            chosenPoints.add(rp);
        end
    end
end
double bestRange = 0;
Point bestPoint = new Point;
foreach cp ∈ chosenPoints do
    if CommModel.rangeEstimate(agent.occupancyGrid,
    cp) > bestRange then
        bestRange = Comm-
        Model.rangeEstimate(agent.occupancyGrid,
        cp);
        bestPoint = cp;
    end
end
Return bestPoint;
```

**Algorithm 2**: Choosing the best rendezvous point.

## 3.6 Replanning in Real-time and Dynamic Environments

**Real-time**

On a modern computer, selection of a rendezvous point typically takes a few hundred milliseconds (see Table 1). Hilditch thinning, generation of a list of rendezvous points, and selection of the final point take longer as the effort progresses, as there is more free space, a larger skeleton, and longer paths to compute. Choosing a frontier takes less time as the effort progresses since the number of open frontiers decreases.

Exact computation times are highly dependent on the specific approach, and numerous optimisations are possible (for example, considering only a subset of all frontiers when choosing a frontier). Nevertheless, while real search-and-rescue scenarios are likely to involve larger occupancy grids and environments extending across multiple levels, we believe that as robots are equipped with better and better processors, this method would scale well and be possible to compute in real-time.

| Early in the exploration effort (20% of env. explored) | |
|---|---|
| Hilditch thinning | 168ms |
| Generation of rendezvousPoints list | 15ms |
| Choosing a frontier | 255ms |
| Deciding on the exact rendezvous point | 128ms |
| **Total** | 566ms |

| Late in the exploration effort (80% of env. explored) | |
|---|---|
| Hilditch thinning | 380ms |
| Generation of rendezvousPoints list | 30ms |
| Choosing a frontier | 98ms |
| Deciding on the exact rendezvous point | 253ms |
| **Total** | 761ms |

Table 1: Typical computation times for elements of the rendezvous point selection process in an 800 x 600 occupancy grid involving four robots (two explorers, two relays), using a 2.4GHz, 2GB machine

**Dynamic Environments**

At this point we have an efficient method for calculation of rendezvous points, and each robot has a fairly accurate picture of where its parent relay is likely to be. What happens in dynamic environments? Among the possible problems that arise in dynamic environments, we consider two cases:

1. A parent finds that the path to rendezvous with its child becomes blocked
2. A child finds that the path to rendezvous with its parent becomes blocked

We will look at the case where the parent is a Relay and the child is an Explorer (although interaction between two relays connected in the team hierarchy would be the same).

**Case 1: Relay cannot reach rendezvous**

In the first case, the Relay finds that it cannot reach rendezvous. It recomputes to find the next best rendezvous point (within a maximum distance threshold), reaches this point, and waits, hoping that the Explorer will find it.

The Explorer reaches the originally agreed rendezvous point, and waits. If after a specific amount of time the parent Relay has not arrived, the Explorer must assume that it could not reach rendezvous, and must replan. The Explorer will have stored the map known to its parent Relay at the previous rendezvous between the two. Since a Relay ferries back and forth between rendezvous and the command centre, its map will not have changed significantly. Thus the Explorer can predict which new rendezvous point the Relay is likely to choose (or if a fallback rendezvous point has been agreed on, it can choose this as a target).

Now, the Explorer must reach this new rendezvous point. If it is reachable by path planning on the Explorer's map, the problem is solved and the Explorer goes to the new rendezvous. If, however, there is currently no path to the new rendezvous, the Explorer continues with frontier exploration, but significantly favouring frontiers that are closer to the new rendezvous point. This can be integrated into the frontier utility equation in terms of a proximity factor.

This does not guarantee that Relay and Explorer will meet

again (indeed, in some cases a robot may become entirely trapped), but it does allow them to make a second attempt.

An alternative solution is for both Relay and Explorer to find their way to the next highest point in the communication chain (the Relay's parent relay) – we hope to examine this scenario in future work.

**Case 2: Explorer cannot reach rendezvous**
Let's assume that the Explorer has had his return path to the rendezvous point blocked. In this case, the Relay can still reach the originally agreed rendezvous point, and waits there.

The Explorer, on the other hand, does not have a path to the rendezvous. Again, it tries to find an alternative route, by favouring frontiers that are closer to the originally agreed point.

Hopefully the Explorer finds an alternative route. If not, and if a significant amount of time has passed without rendezvous, the Relay can either return up the chain of communication (to its parent), or convert to becoming an explorer itself.

# 4. SIMULATION ENVIRONMENT

## 4.1 Simulator Framework

To implement our multi-robot exploration approach and compare it with other existing approaches, we have developed our own JAVA-based simulation environment, the Multi-Robot Exploration Simulator (**MRESim**). MRESim allows for full configuration of environments, either manually or by import of binary image. The simulation framework handles collisions, sensor data and communication as follows: At every time step, the simulation framework requests from each agent a new desired location. If the location is valid, the agent is moved to this location, and new sensor data is simulated and sent to the agent. Following the movement of all agents, the communication model is used to determine whether any agents are within range of one another, either directly or via multi-hop. If yes, all relevant knowledge of the environment is shared between all communicating agents.

At any point a simulation may be paused and agents' individual knowledge bases may be examined. This includes all known free space, safe space, frontiers, calculated paths, communication ranges, map skeleton and rendezvous points.

## 4.2 Communication Model

We have implemented and tested a variety of communication models in our simulations. For experiments reported here we use a standard path loss model with a wall attenuation factor as described in [4]:

$$S = P_{d_0} - 10 \times N \times log_{10}(\frac{d_m}{d_0}) \begin{cases} nW \times \text{WAF} & nW < C \\ C \times \text{WAF} & nW \geq C \end{cases}$$

where $P_{d_0}$ is the reference signal strength, $N$ is the path loss rate, $d_m$ is the distance, $d_0$ is the reference distance, $nW$ is the number of obstructing walls, WAF is the wall attenuation factor and $C$ is the maximum number of walls to consider. This model is widely used in simulation, including the popular USARSim simulator [6]. A typical communication range for an agent is displayed in Figure 5.

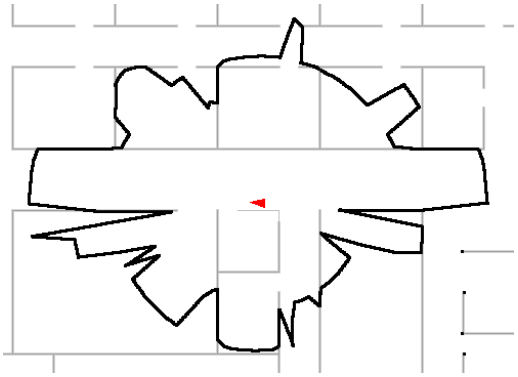


Figure 5: Typical communication range for an agent using the communication model described in section 4.2

## 4.3 Noise

Currently, we assume perfect sensor data and localisation. We are well aware that this is not realistic and real-world systems need to cope with sensor noise and inaccurate maps. However, we believe that our results are useful nevertheless, because:

- steady advances in robotic mapping are leading to ever more accurate mapping techniques
- even with imperfect localisation, the approach is likely to work. When rendezvous points with large communication range are chosen, teammates do not need to rendezvous at precise locations, they merely need to enter the communication range of the rendezvous point, which leaves room for error. Once they meet, they can relocalise based on one another's maps. The shared map itself does not need to be perfect; it's more important is that both robots share the same frame of reference.
- this is an early work examining some of the high-level aspects of multi-robot exploration under limited communication. We hope to take the results from these simulations and test the successful methods in more noisy, realistic simulation environments (such as USARSim [6]) in the near future.

# 5. RESULTS

## 5.1 Static environment

To examine whether the new method for calculating rendezvous points improves exploration efficiency, we ran three algorithms and compared results. The three approaches were:

A. **Frontier Exploration.**
Normal frontier-based exploration without concern for communication range limits. Robots choose frontiers according to the method in section 3.4, and don't return to the ComStation until the whole environment has been explored.

B. **Role-based Exploration, simple rendezvous point calculation.**
Role-based exploration, without the novel method for

rendezvous calculation. Explorers choose their current location as subsequent rendezvous points.

C. **Role-based Exploration, advanced rendezvous point calculation.**
Role-based exploration, using the method for rendezvous calculation detailed in section 3.5.

We compared these three approaches in office-like, open, and cluttered environments. We use two performance metrics to compare the methods:

1. *Total area explored.* We use the union of the area explored by each robot.
2. *Total knowledge of the environment at the command centre.* Since the goal is to return all information to human responders at the point of entry in a search-and-rescue scenario (and since known information that doesn't reach human responders is useless), this metric is of particular interest.

The full details of these experiments for runs involving a team of four robots (two explorers, two relays) are presented in Figure 6. Frontier exploration leads to faster coverage of the environment in both office-like and cluttered environments. However, this advantage is not noticeable at the command centre – it is clear that both role-based approaches are significantly better at relaying new information back to the command centre in all three types of environments.

The novel rendezvous point calculation method proposed in this paper leads to significantly more efficient exploration than the previous rendezvous point calculation – the percent knowledge gain is outlined in Table 2.

| Environment type | % gain |
|---|---|
| Office-like | 9.43 |
| Open | 9.64 |
| Cluttered | 1.12 |

Table 2: Overview of improvement of novel rendezvous point calculation over previous method, in terms of percentage of exploration known at the command centre. More extensive experimental results are presented in Figure 6.

## 5.2 Dynamic environment

To examine our proposed solution to problems of dynamic environments, we spontaneously let obstacles and walls appear in our testing environments. An example of a 'Case 1' situation (Relay cannot reach rendezvous) is presented in Figure 7.

In this case the Relay reverted to the junction point as expected from Algorithm 2. Beta could predict Alpha's choice and pursued frontiers close to the new rendezvous point. In all of our initial experiments, teammates were able to find one another again after being blocked off. We hope to examine these ideas in more detail in future work.

# 6. DISCUSSION AND FUTURE WORK

While results presented here are preliminary, the role-based approach and rendezvous point selection methods presented here show promise regarding future robotic applications such as robotic search-and-rescue.

The novel rendezvous point selection method means that role-based exploration almost matches frontier-based exploration in terms of speed of exploration, while significantly outperforming it in terms of returning knowledge to a central location. Most of today's multi-robot exploration approaches keep team members within range of one another, but for severely communication limited environments, other methods such as these will be crucial.

As robots are likely to be used for exploration of larger and larger areas, methods need to scale up in terms of memory requirements and computation time. The rendezvous selection process presented here is easy to implement and fast, and we believe that it could be applied to multi-level maps as well.

However, much work remains to be done: the noise and localisation assumptions must be relaxed, and more realistic simulations conducted. Experiments need to be conducted with larger numbers of robots. There are numerous potential extensions to the role-based approach, such as the development of dynamic hierarchies or the deployment of RFID tags or motes to aid in the exploration process. We are also interested in the fusion of aerial data (e.g. overhead cameras) and ground data (e.g. rangefinders), and the use of aerial robots to create a communication infrastructure for ground robots. We hope to apply some of the ideas presented here to such scenarios in the future.

# 7. CONCLUSIONS

For many robotic applications, information about an environment must be gathered by a team of robots and returned to a central location. In robotic search-and-rescue, this corresponds to the human responders' point of entry. Using a team of robots for such a task brings up problems of team coordination, knowledge sharing, and communication. Particularly in search-and-rescue scenarios, communication can be severely limited and robust strategies must be devised that take this into account.

We have proposed Role-based Exploration, in which robots either explore the deep reaches of the environment, or ferry information from explorers up the communication chain to the central command centre. While purely frontier based methods cover an environment faster, role-based approaches allow for information to reach human responders more quickly and more often.

Explorers and relays must rendezvous to exchange information and the selection of rendezvous points turns out to have a significant effect on the exploration effort. We propose a method from digital image processing, thinning, to skeletonize the map. This skeleton can be used to find an even distribution of possible rendezvous points, including those found at junctions. By careful selection of rendezvous points (close to future exploration areas), exploration becomes significantly more efficient. These rendezvous points can be used for replanning when unexpected changes in the environment occur. This could be of great use in dynamic environments such as those encountered in search-and-rescue scenarios.

While role-based exploration is in an early stage, we believe that certain applications will require explicitly planning for autonomous exploration beyond communication range limits. We hope the ideas presented here are an early step in that direction, and may be used on their own or as extensions of existing approaches in the near future.

## 8. REFERENCES


[1] S. Alpern. The rendezvous search problem. *SIAM J. Control Optim.*, 33(3):673–683, 1995.

[2] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *Robotics and Automation, IEEE Transactions on*, 15(5):818–828, Oct 1999.

[3] R. Arkin and J. Diaz. Line-of-sight constrained exploration for reactive multiagent robotic teams. *Advanced Motion Control, 2002. 7th International Workshop on*, pages 455–461, 2002.

[4] P. Bahl and V. N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. *IEEE Infocom 2000, Tel-Aviv, Israel*, 2:775–784, March 2000.

[5] W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376 – 378, 2005.

[6] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper. Usarsim: a robot simulator for research and education. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1400–1405, April 2007.

[7] J. de Hoog, S. Cameron, and A. Visser. Role-based autonomous multi-robot exploration. In *International Conference on Advanced Cognitive Technologies and Applications (COGNITIVE)*, November 2009.

[8] M. B. Dias and A. T. Stentz. A free market architecture for distributed control of a multirobot system. In *6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pages 115–122, July 2000.

[9] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 2000.

[10] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, 94(7):1325–1339, July 2006.

[11] C. Hilditch. Linear skeletons from square cupboards. *Machine Intelligence*, 4:403–420, 1969.

[12] A. Howard, M. Mataric, and G. Sukhatme. An incremental deployment algorithm for mobile robot teams. *Intelligent Robots and System, 2002. IEEE/RSJ International Conference on*, 3:2849–2854 vol.3, 2002.

[13] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 4, pages 3232–3238 vol.3, Oct. 2003.

[14] L. Lam, S. Lee, and C. Suen. Thinning methodologies: A comprehensive survey. 14(9):869–885, September 1992.

[15] J. Lin, A. S. Morse, and B. D. O. Anderson. The multi-agent rendezvous problem. part 1: The synchronous case. *SIAM J. Control Optim.*, 46(6):2096–2119, 2007.

[16] A. Mosteo, L. Montano, and M. Lagoudakis. Multi-robot routing under limited communication range. *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1531–1536, May 2008.

[17] M. Powers and T. Balch. Value-based communication preservation for mobile robots. In *7th International Symposium on Distributed Autonomous Robotic Systems*, 2004.

[18] M. N. Rooker and A. Birk. Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, 15(4):435–445, 2007.

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

[20] B. Santos Pimentel and M. Fernando Montenegro Campos. Multi-robot exploration with limited-range communication. In *Anais do XIV Congresso Brasileiro de Automática (CBA'02)*, 2002.

[21] W. Sheng, Q. Yang, J. Tan, and N. Xi. Distributed multi-robot coordination in area exploration. *Robot. Auton. Syst.*, 54(12):945 – 955, 2006.

[22] R. G. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. L. S. Younes. Coordination for multi-robot exploration and mapping. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 852–858. AAAI Press / The MIT Press, 2000.

[23] J. Vazquez and C. Malcolm. Distributed multirobot exploration maintaining a mobile network. *Intelligent Systems, 2004. Proceedings. 2004 2nd International IEEE Conference*, 3:113–118 Vol.3, June 2004.

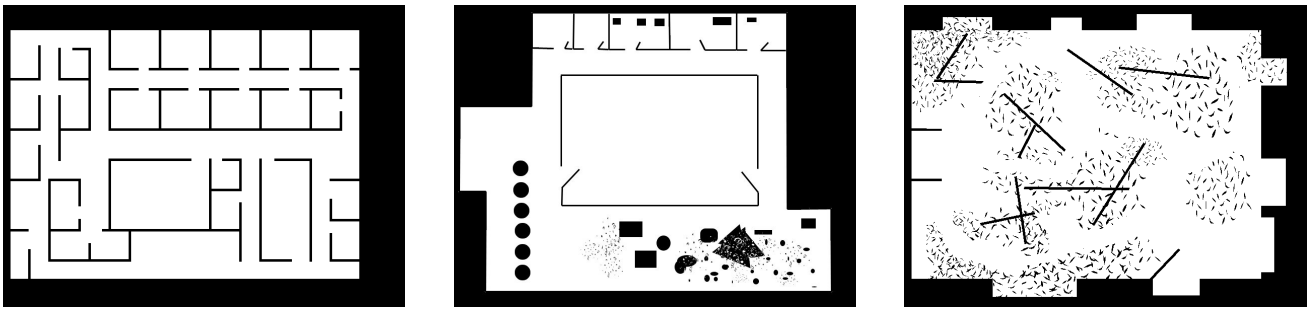
[24] A. Visser and B. Slamet. Including communication success in the estimation of information gain for multi-robot exploration. *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops, 2008. WiOPT 2008. 6th International Symposium on*, pages 680–687, April 2008.

[25] A. Visser and B. A. Slamet. Balancing the Information Gain Against the Movement Cost for Multi-robot Frontier Exploration. In *European Robotics Symposium 2008*, Springer Tracts in Advanced Robotics, pages 43–52. Springer-Verlag, February 2008.

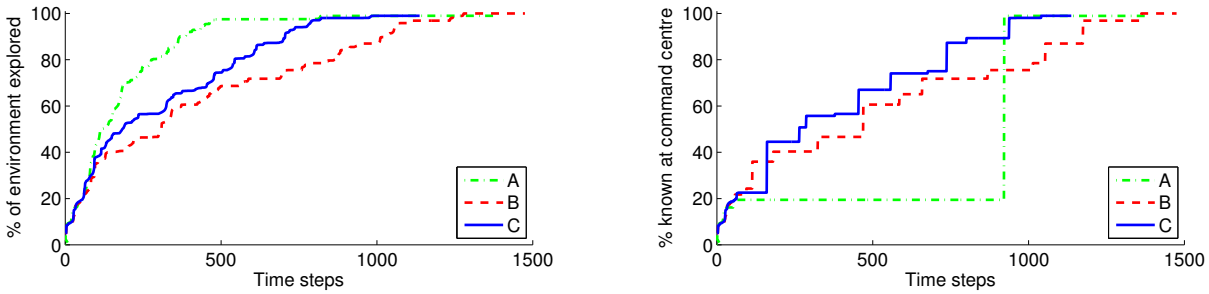
[26] K. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1160–1165, Sept. 2008.

[27] B. Yamauchi. Frontier-based exploration using multiple robots. In *AGENTS '98: Proceedings of the second international conference on Autonomous agents*, pages 47 – 53, New York, NY, USA, 1998. ACM.

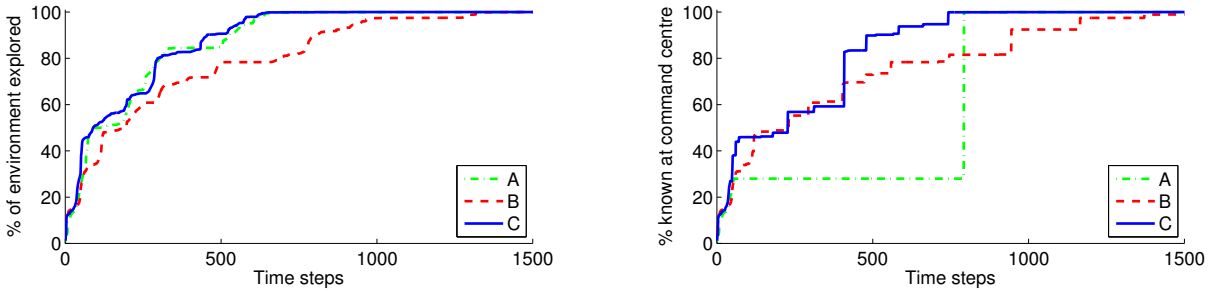
[28] R. Zlot, A. Stentz, M. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, 3:3016 – 3023, 2002.



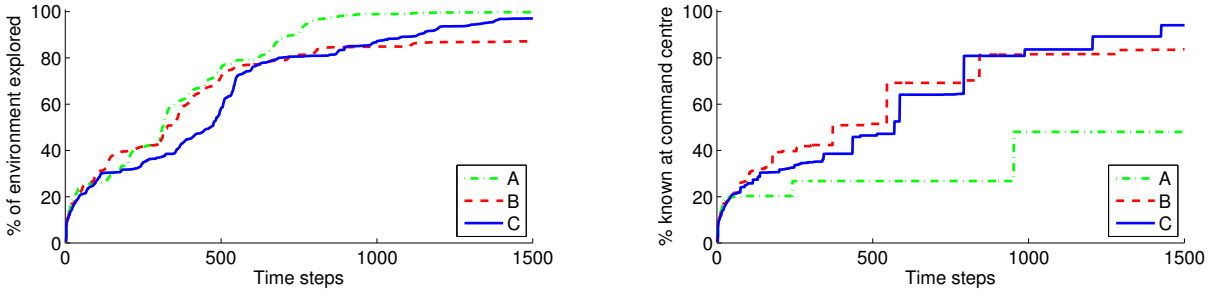
(a) Three environments used for testing: office-like, open, and cluttered




(b) Results in an office-like environment: % of environment explored (left) and % known at command centre (right)

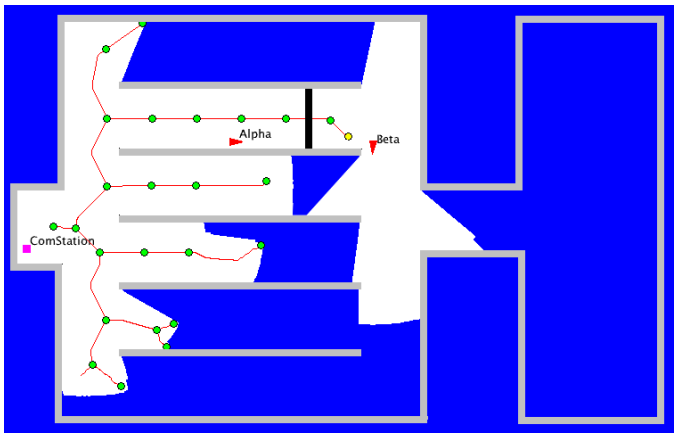



(c) Results in an open environment: % of environment explored (left) and % known at command centre (right)

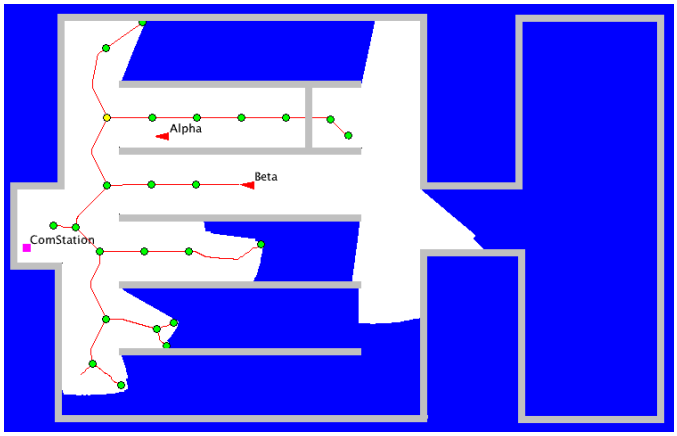



(d) Results in a cluttered environment: % of environment explored (left) and % known at command centre (right)

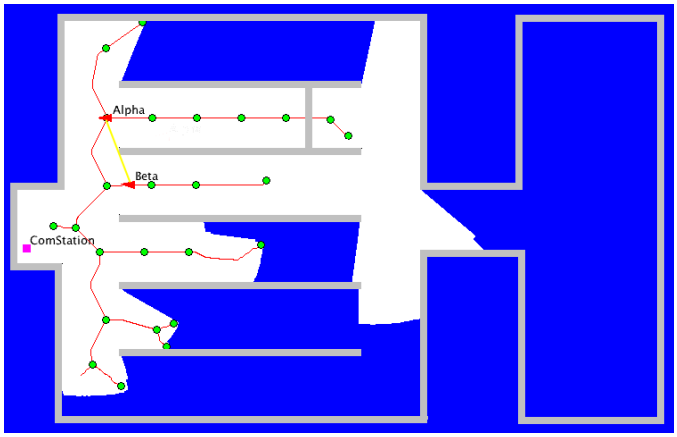
Figure 6: Evaluation of performance metrics after running algorithms A, B and C in 3 different types of environments




(a) A sudden wall (previously not there) blocks Alpha from reaching the rendezvous point (yellow). Beta waits a specific amount of time for Alpha to appear.




(b) Alpha recalculates, and chooses the rendezvous point having highest degree (a junction point). Beta recalculates, assumes Alpha will head for the same junction point, and pursues frontier exploration with a preference for frontiers near the new rendezvous point.




(c) Alpha and Beta meet at the new rendezvous point and exploration can proceed as normal.

Figure 7: Use of rendezvous points to deal with unexpected obstacles in dynamic environments