Bar-Ilan University
אוניברסיטת בר-אילן

Intelligent User Interface for Multi-Robot Search

Shahar Kosti

Submitted in partial fulfillment of the requirements for the Master's Degree in the Department of Computer Science, Bar-Ilan University

Ramat Gan, Israel 2013

This work was carried out under the supervision of Prof. Gal A. Kaminka and Dr. David Sarne.
Department of Computer Science, Bar-Ilan University.

ACKNOWLEDGMENTS

First, I would like to thank my advisors Prof. Gal A. Kaminka and Dr. David Sarne for their excellent guidance and constant support in the past two years. I learned quite a bit about scientific research, all thanks to their great instruction. Working with them was a real pleasure, not only on the professional level, but on the personal level as well.

Thanks to my colleagues and friends from the Bar-Ilan University in general, and MAVERICK lab in particular, for the great time we had together.

Special thanks to Gabriella Melamed, MAVERICK lab manager, for assisting with administrative issues, and in particular for helping with the arrangements for the experiment.

I also wish to thank Tal-Oron Gilad from Ben-Gurion University, for providing us with a Hebrew translation of the NASA-TLX questionnaire.

Thanks to my dear family for their support and encouragement, and for enabling me to complete this work. Thanks to my girlfriend for her patience and understanding.

Finally, I would like to thank my grandfather, who never ceased to encourage me throughout this journey.

Contents

ABSTRACT

Human-Robot Interaction (HRI) is an important part of many robotic systems, in particular systems designed for search or exploration missions. The traditional HRI methods of controlling a single robot by an operator have their shortcomings, for example the need to continuously monitor a live (synchronous) video display. When the number of robots in the system increases, the traditional methods are even harder to apply, especially when attempting to display camera imagery from multiple robots. In this work, we present an asynchronous and user-guided interface for operators of a robotic system, supporting exploration (search) of an unknown area. Focusing on Unmanned Ground Vehicles and Urban Search and Rescue missions, we describe a novel asynchronous interface and operation scheme. With our interface, the operator is able to cover the recorded imagery of a given indoor environment, by selecting points of interest on a map and watching highly-relevant images that cover it. We evaluated our interface by comparing it with the state-of-the-art asynchronous interface, and found it to perform better under certain conditions, in terms of found targets.

Chapter 1

Introduction

Recently there is a growing interest amongst Human-Robot Interaction (HRI) researchers in the usage of multi-robot systems for exploration or search missions [13, 23, 36, 39]. Exploration is a fundamental problem of robotics, in which one or more robots are used to gain knowledge of an unknown area. Applications of the problem include Urban Search and Rescue (USAR) [9], military uses [21] and planetary exploration [3].

Human operators are frequently included in the control loop of existing robotic systems, for several reasons. While navigating and mapping an environment could be done autonomously by robots, interpretation of live camera images requires visual perception skills. To date, visual perception in such missions is still done better by the human brain [41]. In addition, robots navigating autonomously might get stuck and need human intervention in order to get back on track. Finally, humans must take the commander role and monitor the mission progress, while changing objectives and priorities if necessary.

The traditional approach to single robot supervision, particularly in search tasks, is commonly referred to as *camera guided teleoperation* [36]. This approach has several caveats. It requires constant attention of the operator, a high-bandwidth and low-latency communication method and is difficult to scale due to increased operator workload and degraded performance [11].

Scaling up the number of robots imposes challenges to HRI. For proper operation, we need to provide users with the means to understand the robots status and obtain sensory information. In common multi-robot search interfaces, video interpretation is done by observing live video feeds from multiple robots, while the robots have some level of autonomy. Having to view multiple live video feeds, while navigating the robots and following their status, increases the workload on the operator [39]. Automating the navigation task could help to reduce the operation workload [40], and is the basis for asynchronous interfaces.

Recent work [36] on multi-robot interfaces investigated the usage of asynchronous operation schemes in the context of Urban Search And Rescue (USAR) missions [28]. With the asynchronous approach, the robots are exploring the environment autonomously, while the operator is presented with recorded imagery (*asynchronous*) from the robots' camera, rather than live imagery (*synchronous*).

Two major concerns of asynchronous interfaces for multi-robot search, are how to select most relevant imagery of display, and how to display it to the operator. In a recent asynchronous interface [39], images are stored in a database and ordered by a utility value, which is computed by the image area that wasn't already seen. Users can then request the next image to view, and navigate through images that were recorded near the selected image.

Several questions may arise in this context: can we allow users to have more freedom in image selection? can we provide users with the ability to control the mission progress? We suggest an asynchronous and user-guided interface, in which the operator is requested to cover the recorded imagery of a given indoor environment, by selecting points of interest (POI) on a map. The interface provides the operator with highly-relevant images of the POI from several view points, after applying a dynamic filtering and ranking process over the recorded images.

We evaluated our interface by comparing it with the state-of-the-art asynchronous interface. The experiment was conducted in a simulated environment, generated by the USARSim simula-

tion package. Experimental conditions consisted of two USAR environments, based on an office environment from the 2006 RoboCup Rescue competition finals. 32 adult students performed a USAR mission in one of the interfaces. We found our interface to perform better than the existing interface, under certain conditions, in terms of found targets. We also found our interface to be less sensitive to changes in map size, suggesting that it might scale better.

An important part of our interface is the ability to retrieve images that cover a given point of interest. This is not a trivial task, as the interface should respond quickly to user queries, but the total number of images may be large even for small environments. We describe methods for efficiently storing the images in a database, and finding images that cover the point of interest. As the number of images that cover a certain location may also be large, we describe a method to rank images according to a computed utility value.

Implementing the methods to store and quickly retrieve images that match the points of interest, is not a trivial task as well. There are several decisions to be made, and an incorrect choice could affect the user experience drastically. To evaluate these methods, we conducted an additional experiment that compared different methods for finding images, using real-world data from the interface evaluation experiment. We found out that the chosen method in our implementation was efficient enough for our data set.

List of Publications The work reported in this thesis has been published in:

- S. Kosti, D. Sarne, and G. A. Kaminka. Intelligent user interface for multi-robot search [extended abstract]. In *HRI 2012 Workshop on Human-Agent-Robot-Teamwork (HART 2012)*, 2012.

- S. Kosti, D. Sarne and G. A. Kaminka. An Effective Collaborative Interface For Multi-Robot Search. In *The First Israeli Human-Computer Interaction Research Conference (IsraHCI)*, 2013.

Chapter 2

Related Work

Our work is related to the Human-Robot Interaction (HRI) research area [16], which combines several fields of studies, including Robotics and Human-Computer Interaction (HCI). In this chapter we review related work from the HRI literature. In Section 2.1, we review some work on single-robot search interfaces, and discuss the difficulties of designing a robust interface for such systems. In Section 2.2, we review work on multi-robot interfaces, and discuss how various researchers have chosen to solve the problem of scaling from a single robot to, including synchronous and asynchronous interfaces.

2.1 Single-robot interfaces

Darken et al. [12] tested the effect of various environmental conditions on live video feedback, viewed by a remote observer. They concluded that unless a video stream is augmented with additional information, for example spatial data, a remote observer cannot be expected to extract useful information from it and maintain spatial orientation. They also suggested augmenting the video stream with symbolic information, such as the output of an Object Recognition system.

Bruemmer et al. [5] evaluated a 3D interface for robot teleoperation, that included both a map

and video on the same display. A 3D environment was built in real-time from laser scans, and video snapshots were placed in the appropriate positions. Their results suggest that there are alternatives to video-based interfaces, and that using snapshots (instead of live video) degrade performance significantly. We did not expect performance to degrade in our case, considering later work on asynchronous interfaces in this area.

Nielsen et al. [27] compared the importance of map and video displays for navigation tasks. They evaluated 2D and 3D interfaces with different configurations of map and video displays: map and video side-by-side, map-only and video-only. Experiments were conducted with a single robot, both in simulation and with a real robot. The results were not definitive, suggesting that sometimes a map is more helpful than video, but sometimes it is the opposite. They did suggest, however, that integrated designs might be more useful for navigation than side-by-side displays.

Yanco et al. [45] described the evolution process of two interfaces, originally developed by the University of Massachusetts Lowell and the Idaho National Laboratory. The interfaces were meant to control a single robot in an Urban Search and Rescue (USAR) mission. The evolution process consisted of usability tests that suggested to combine features from both interfaces. They also described the usage of HCI principles and techniques for evolving User Interface (UI) designs for single-robot control.

Yanco et al. [44] also discussed the AAAI Robot Rescue Competition and how HRI applies at that context. Their report covers a few competitions, held between the years 2002 and 2004, and several participating teams. They studied HRI in run-time, by videotaping the operator, robots and monitors. Overall performance was measured and each team (interface) was scored according to that. They concluded with a few suggestions for designing effective interfaces for HRI, for example using a single monitor with large video display and avoiding a design that requires switching between windows.

Yanco and Drury [43] discussed acquiring Situational Awareness (SA) with a single robot.

Through trials, they found out that users could spend more than 30% of their time maintaining SA. Later on [14], they developed a technique for analysing SA in five awareness categories. They presented the analysis of two interfaces, and showed which interface excels in which category.

Hughes et al. [22] discussed how camera properties such as poor camera placement and narrow field of view, can significantly impair the operator performance. They have conducted two user studies in simulation, and in with various camera settings. Their results suggested providing a camera that can be controlled independently from the robot, and that providing multiple cameras may be beneficial for navigation tasks.

2.2 Multi-robot interfaces

We now review related work on multi-robot interfaces for search missions. Naturally, we would expect a robotic system to perform better when the number of robots increases, but additional concerns arise when when transitioning from a single-robot system to a multi-robot system. The main questions related to the human operator side, are how to control the robots, and how to provide the operator with the means to understand the robots status and obtain sensory information. Automating some of the tasks may be beneficial, but deciding which task to automate and how to automate it, is not trivial.

2.2.1 Synchronous (live) video

Wang et al. [42] presented a system called MrCS (Multi-Robot Control System), which provided means to navigate multiple simulated robots, show live video from multiple robots on a dedicated display, and view laser data on the map. Their research compared various control schemes for a team of three simulated robots. Participants interacted with the robots by either providing them with waypoints, performing direct teleoperation or by controlling a PTZ camera.

Humphrey et al. [23] presented an interface design for multi-robot bomb defusing task. Their approach was video-centric: a large video display, augmented by status information to increase the operator's Situational Awareness (SA) and improve multi-robot management. A "halo" display around the video feed contained the additional status information. Results showed that increasing the number of robots increased SA, but also increased operator workload significantly. In our interface, the main display is map-centric, rather than video-centric.

Valero et al. [33] described the design, evolution and evaluation of an interface for multi-robot exploration. They focused on providing SA using a 2D map and a 3D map combined with video projection, and supporting control of multiple levels of autonomy within that interface.

The most extensive and related work, was carried out as part of a joint project between Carnegie-Melon University and the University of Pittsburgh. From here until the end of this chapter, the domain of the presented experiments is USAR missions, and participants were required to mark disaster victims on a map. Experiments were conducted in a simulated environment.

In one approach [25, 40], the authors introduced the concept of a *fulltask*, which is composed of sub-tasks *exploration* (cover as much area as possible) and *perceptual search* (locate victims on video screens). Their goal was to determine which sub-task is a better candidate for automation, by requesting participants to perform them together and separately. Results were measured by comparing the relevant metrics between the sub-task and the *fulltask*. Their results showed that while performing each sub-task separately exhibited improved performance over the *fulltask*, the difference in the *perceptual search* sub-task was much higher. They suggested that automating the *exploration* task should provide improved performance for similar tasks as well. We suggest to automate the *exploration* task as well, but also automate some of the *perceptual search* task. Instead of observing video feeds from all robots continuously, we display only the relevant imagery.

In the approach described in [25, 40], the robot paths for the automated *exploration* sub-tasks were pre-recorded, meaning that participants did not have to deal with situations such as stuck

robots. A later work than this approach [31], used the Carnegie-Mellon Robot Navigation Toolkit (CARMEN) [26] to perform autonomous exploration in real-time. Results support the previous findings by showing an increased performance when *exploration* is automated.

Another work [37] investigated various ways to organize a team of two operators, to perform a search task using 24 robots. Robots were controlled by placing waypoints on a map, or by direct teleoperation. In the *Individual* condition, each of the participants controlled 12 robots individually, while in the *Call Center* condition, participants shared control of the 24 robots. In the *Call Center* condition, 24 live video streams were presented on the screen.

2.2.2 Asynchronous video

One major disadvantage of *synchronous* video for multi-robot interfaces, is that the operator has to constantly monitor multiple live video feeds, and perform mission-related tasks at the same time. Automating the robot navigation task, as suggested in [40], is one necessary step to lower operator workload. Another step forward is to eliminate the need to view live video, thus turning the interface to use *asynchronous* video. Two major concerns of asynchronous interfaces for multi-robot search, are how to select most relevant imagery of display, and how to display it to the operator.

Velagapudi et al. [34–36] compared two interfaces: *synchronous* and *asynchronous*. The former was similar to the MrCS interface from an earlier study [42], while the latter had no live video display. Instead, an operator directed the search task by assigning robots with waypoints. After reaching the final waypoint, the robot would take a panoramic image of that location, which appeared as a new symbol on a map. The results showed better performance for the *asynchronous* interface when the number of robots increased. Our work also suggests an asynchronous interface and uses the map as a primary tool. However, in our system the robots are completely autonomous, and we make a far more extensive usage of the map.

A recent paper by Wang et al. [38,39] compared two interfaces for multi-robot search: *streaming video* and *image queue*. The robots were autonomous but manual control was allowed. In the *image queue* interface, images from all robots are stored in a database and sorted by utility, which is calculated by the size of visible area that wasn't already seen. The images with the highest utility are presented on a "filmstrip", along with a local map at that time. The user can select an image and view a "sub-queue" of nearby images (to get a better perspective). Results showed that the performance did differ significantly, in terms of the total number of victims found. However, with the "image queue" interface there were less errors (false positives and negatives) and the workload was lower. We refer to this work in chapters 5 and 6, by comparing our interface with the *image queue* interface.

Brooks et al. [4] further developed the *image queue* approach, by introducing an Automatic Target Recognition (ATR) mechanism to the image display. The ATR mechanism assisted users with detecting victims in images, by drawing a red box around the supposed victim in the image. To provide better realism, a false rectangle was drawn in 20 percent of the images that contained no victims. Researchers compared *streaming video*, *no-cued image queue* (no ATR) and *cued image queue* (with ATR), and did not find a significant advantage to any of the methods in terms of found victims, but found a significant advantage in terms of marking errors.

Chapter 3

User Interface based on Point of Interest (POI)

In this chapter, we present an asynchronous user interface system, that enables users to view recorded imagery and to mark locations on a map. Navigating between recorded images is done in a user-guided manner, by selecting a Point of Interest (POI) on a map (Section 3.1). Section 3.2 describes generic user interface components that are not necessarily specific to our method. We then describe the specific details of our interface in Section 3.3. Finally, in Section 3.4 we briefly summarize the POI interface.

3.1 Overview

The system is designed to perform two main tasks: view recorded imagery and mark locations on a map. In the context of a USAR mission, the marked locations can specify the approximate positions of disaster victims. The system is composed of two parts, *User Interface* and *Image Retrieval*. The former is responsible for getting user requests and displaying images, while the latter is responsible for processing user input, searching through the recorded images and ranking

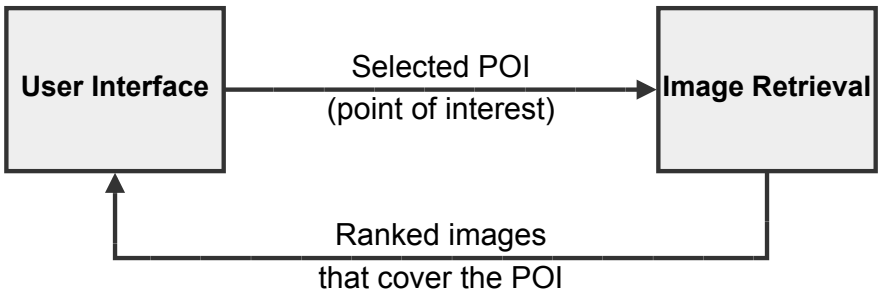


Figure 3.1 Data flow in the system.

them. Figure 3.1 illustrates the data flow in the system. In this chapter we focus on the *User Interface* part, and treat the *Image Retrieval* part as a black box. The next chapter describes *Image Retrieval* in more detail.

The user interface is consists of three main components, *Map*, *Current Image* and *Relevant Images*. Figure 3.2 illustrates the general screen layout. The general role of the user interface components, and how they are used with *Image Retrieval*, is described below.

- *Map* - responsible for receiving user requests for POIs (input mechanism).
- *Current Image* and *Relevant Images* - display the returned ranked images (output mechanism).

3.2 Generic components

The interface input is an image database collected by one or more autonomous robots, that provide an on-going stream of camera images and range scanner readings. Incoming range scanner readings are used to create a map using a technique called Simultaneous Localization and Mapping (SLAM) [15]. The Field Of View (FOV) polygon of a particular camera image is the map area that is covered by that image.

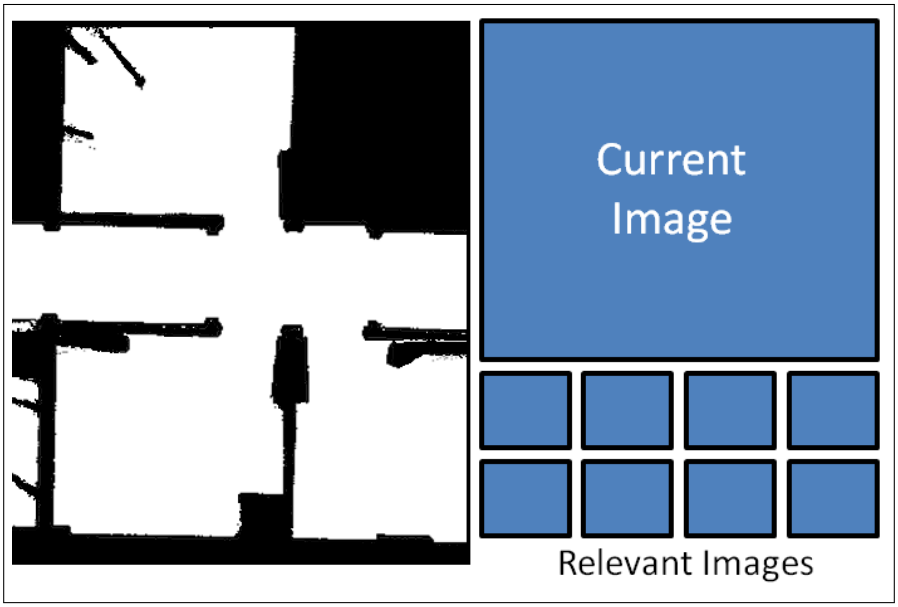


Figure 3.2 User interface layout.

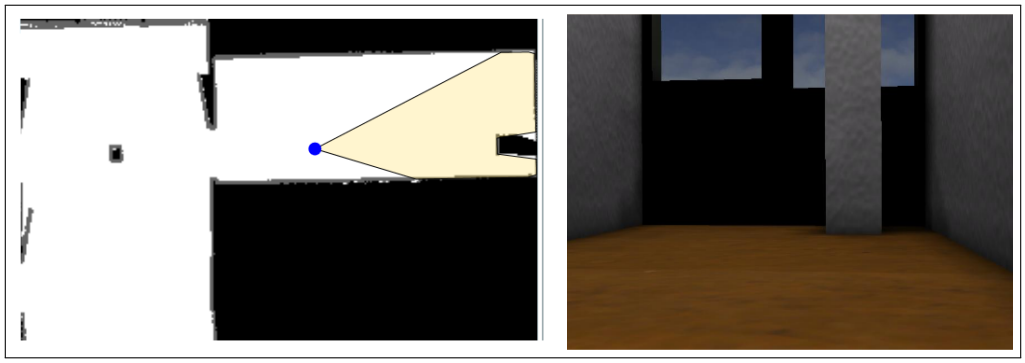


Figure 3.3 Basic interface components. Here we see an image of room that consists of walls, windows and a column, along with the corresponding map. Note how the column, which is placed in the right side of the image, appears in the map.

In the maps that will be presented from now on, white represents areas that were visible to the robot while recording, and black represents occupied areas that were obscured by objects (walls, obstacles, etc.). A dark dot represents the robot position while recording the image, and a light polygon represents the FOV polygon (map area that the image covers). Figure 3.3 is an example of an image and the corresponding map.

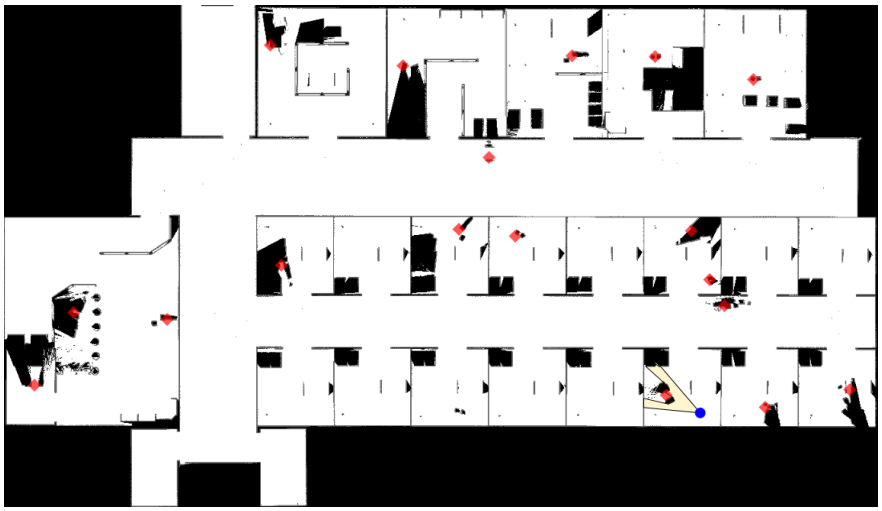


Figure 3.4 A map after placing a few marks. Marks appear as red symbols.

Marking A key task of users of our interface is to mark locations on a map, for example the position of disaster victims. Marking is done by a simple right-click on the desired location, which results in a special symbol appearing at that location. Right-clicking on the mark symbol removes it from the map. Figure 3.4 illustrates the map after placing a few marks. Marks can be hidden and restored using a dedicated button.

Map control Map control is done using a mouse device, with panning done by holding a button and moving the mouse ("drag"), and zoom done with the mouse wheel. In addition, dedicated zoom buttons are available above the map. This kind of map interface is similar to systems such as Google Maps, and thus should be familiar to most users.

3.3 POI interface components

3.3.1 Navigating between images

Navigating between recorded images is done in the following manner. The user selects a POI (point of interest) on the map by clicking on it. The system finds all camera images that cover the selected point, and ranks them. The highest-ranked image is displayed as the *Current Image*. In most cases, the highest-ranked image should provide a good view of the POI. To provide other perspectives of the POI, a subset of the highest-ranked images is displayed on the map. Clicking on one of them replaces the *Current Image* with the clicked image. All other images that cover the POI are also available for display, and presented as thumbnails (*Relevant Images*). Again, clicking on one of them will present it as the *Current Image*.

Figure 3.5 illustrates the interface after a POI was selected. The user can refine the automatic image selection, to get a better perspective of the POI, by selecting one of the other images that cover the POI. This can be done by either clicking on one of the smaller dots on the map, or by choosing it from the "Relevant images" display.

To conclude, the flowchart in Figure 3.6 illustrates how the user navigates between images with our interface. Compare the flowchart with the high-level data flow description from Figure 3.1; all items in the flowchart are from the *User Interface* part, except for the *Image Retrieval* part, which is represented by the upper rectangle ("Find images ..."). Note how the refinement of automatic image selection is done entirely in the user interface, as the images are already ranked.

It is important to mention that the interface does not employ a specific map coverage strategy. It is up to the user to determine which areas to visit first and in what way to do so.

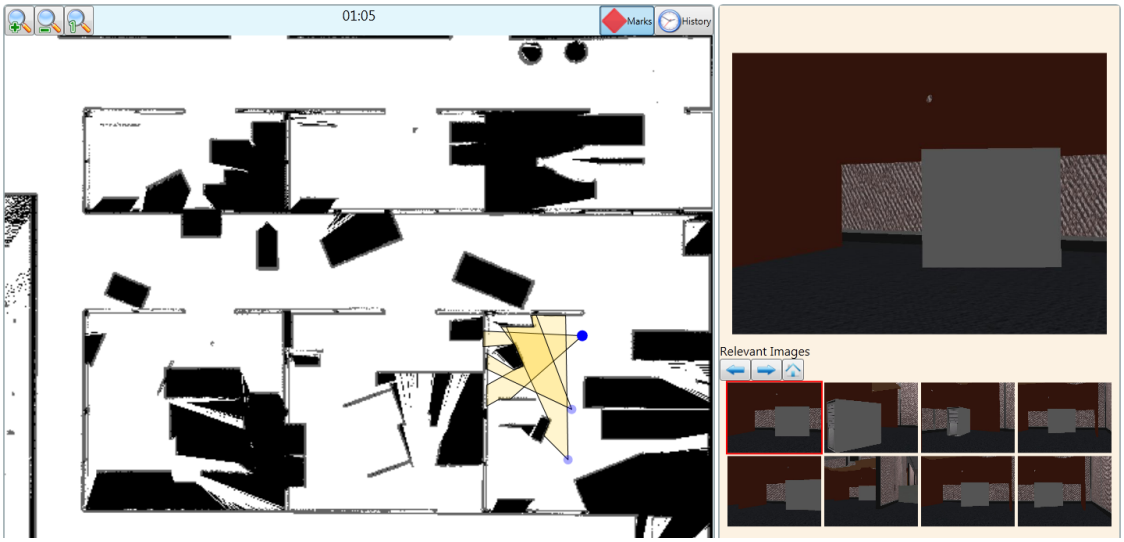


Figure 3.5 The interface after a point of interest was selected. The highest-ranked image is displayed in the upper-right corner. The bold dot on the map represents the robot location while recording current image. Some other images are available for selection on the map, and displayed as smaller dots. All other images that cover the POI are displayed as thumbnails, below the current image.

3.3.2 Image selection history

In order to distinguish between visited and unvisited map areas, users are provided with a dedicated map layer to show previously seen images. As illustrated by figure 3.7, seen images are displayed on the map as light-blue polygons. Viewing the same area multiple times will show up as darker color. The layer is activated by default, and can be deactivated if necessary.

3.4 Summary

The system is composed of two main components, *User Interface* and *Image Retrieval* (Figure 3.1). The user provides input to the system mainly by clicking on the map, and receives the output through the image display and the map. After selecting a POI on the map and clicking on it, the system finds all camera images that cover the selected point and ranks them.

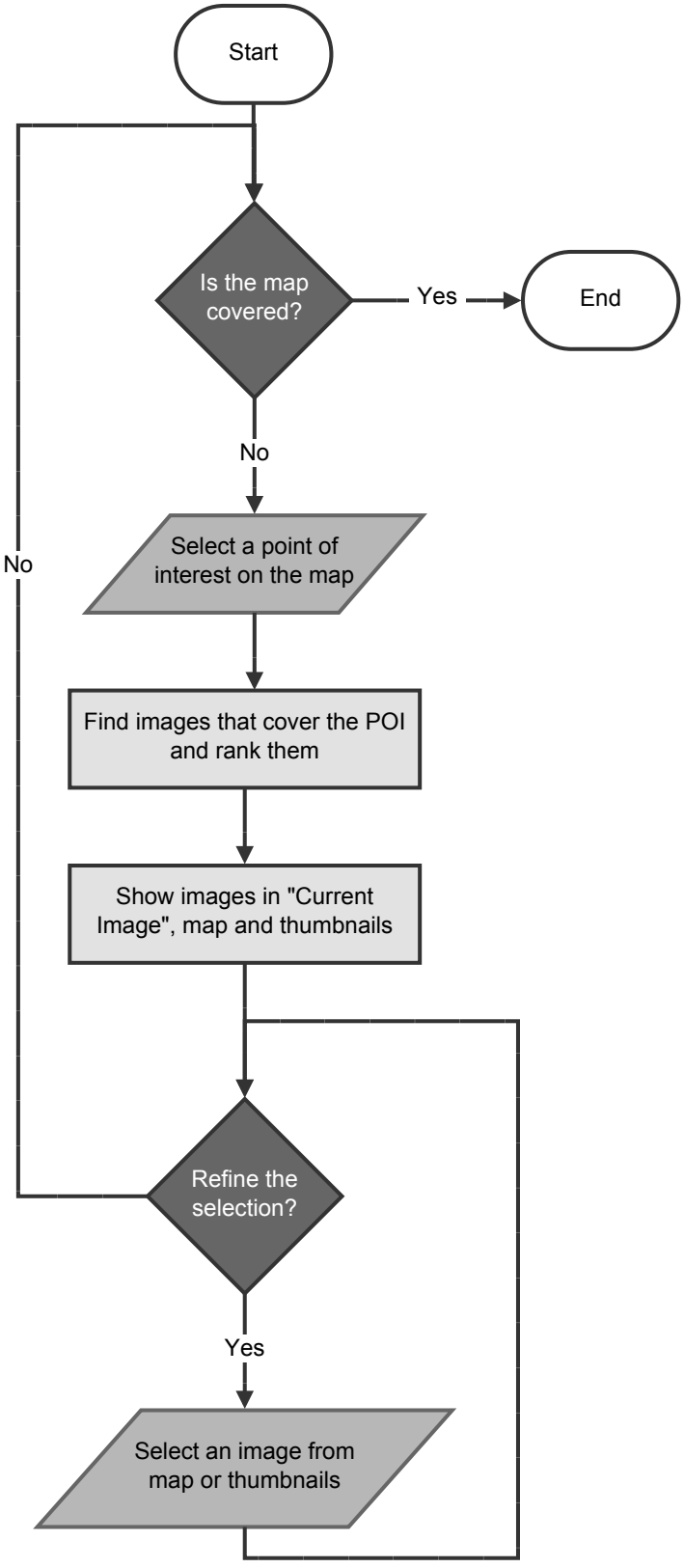


Figure 3.6 Image navigation flowchart. User input is represented by a parallelogram, a loop is represented by a rhombus, and (automatic) processing steps are represented by rectangles.

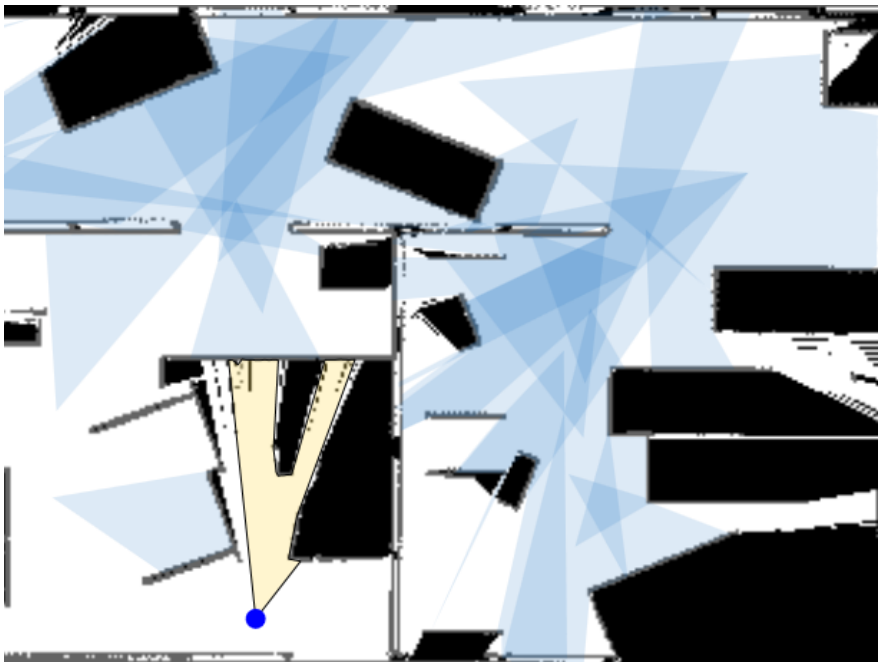


Figure 3.7 History Layer

The ability to retrieve recorded images according to a user-selected POI, is a crucial part of the system. From a user perspective, image retrieval should be transparent; clicking on a point of interest should result in a good view of the chosen location, appearing immediately. From the implementation perspective, however, this is not trivial task. The *Image Retrieval* component must be implemented carefully, so that users can focus on performing the mission. The next chapter deals with the implementation details of this component.

Chapter 4

Image Storage and Retrieval

In this chapter, we discuss the storage mechanism and image retrieval methods, that lie at the core of our system. As described in the previous chapter, one part of the system is responsible for finding and ranking all camera images that cover the user-selected point of interest. Figure 4.1 provides an outline for this chapter, by illustrating the data flow in the system with a focus on the *Image Retrieval* part.

In Section 4.1, we describe how the images are stored and what additional data is required. We also discuss the process of combining sensory data from multiple sensors, to be stored as a single entry in the image database. Section 4.2 discusses three methods for finding all images the contain the user-requested POI, as well as methods for addressing image requests when the POI is outside the visible map area. In Section 4.3 we describe a framework for ranking the images that cover the selected point. Finally, in Section 4.4 we discuss the image utility function that was chosen for this study, followed by examples of how it performs.

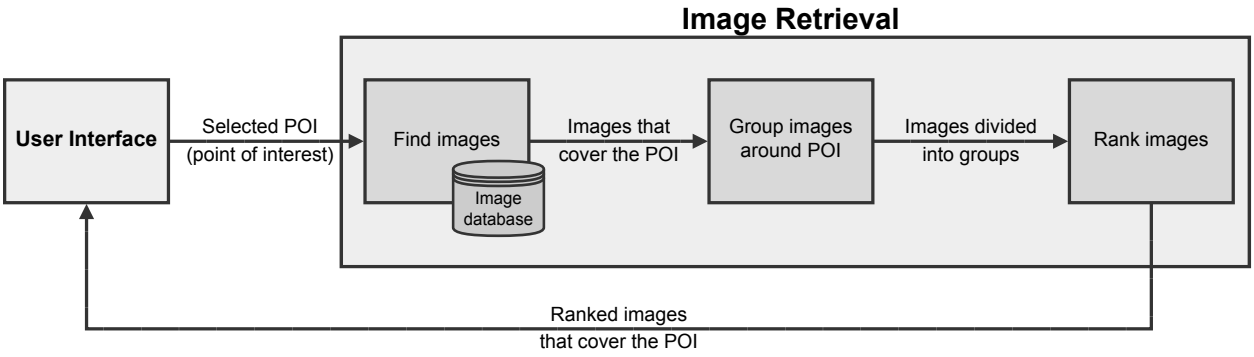


Figure 4.1 Data flow in the system, with a focus on *Image Retrieval*.

4.1 Storing images

Recorded images and range scanner from all robots, are saved in an image database and as a map. Figure 4.2 illustrates the storage-related data flow in the system. Each database entry contains auxiliary information regarding the robot at the time of recording. By storing the auxiliary information with each image, we save significant computation time when responding to queries. The following data is kept for each image:

- Camera ID, a unique identifier for the robot-camera pair.
- Timestamp, the time in which the image was taken.
- The image itself.
- Robot location and orientation (heading) at the image recording time.
- Field of View (FOV) polygon, range scanner readings from the recording time, kept as a polygon that begins and ends at the robot location. The polygon indicates which part of the map is covered by the recorded image.
- Additional data for the “point in polygon” query, as suggested in Section 4.2.1.

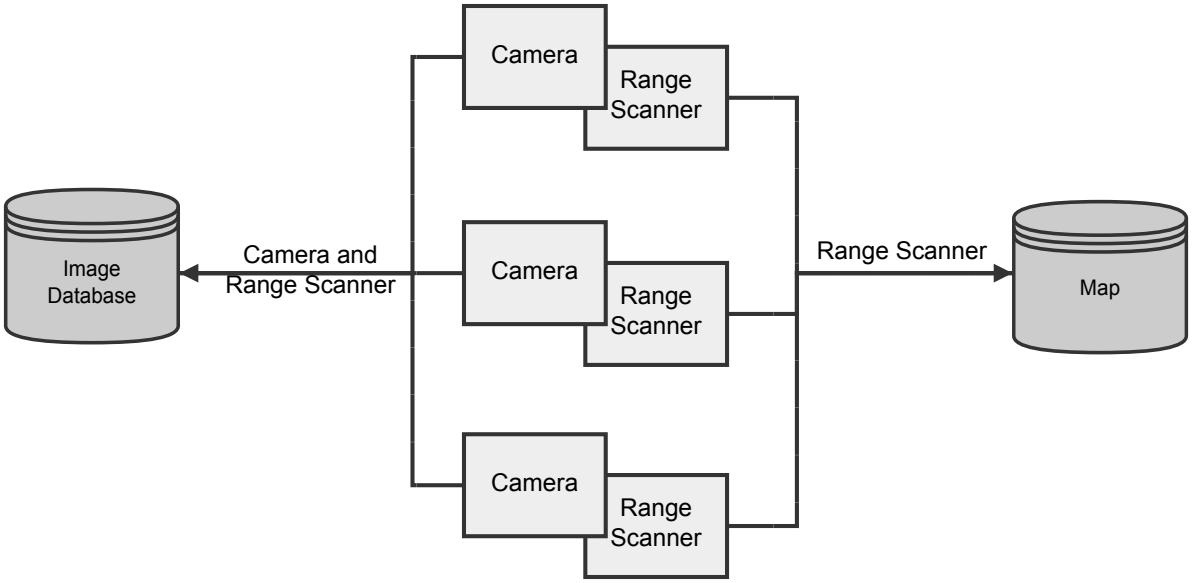


Figure 4.2 Storage-related data flow. This is an example for one possible configuration, with a group of three robots that carry a single camera and a range scanner. Other configurations are also possible.

Constructing the FOV polygon from a robot position rp, heading h and a set of n range scanner readings $\langle r_1, \ldots, r_n \rangle$, given in polar coordinates:

1. First, we transform the readings to Cartesian coordinates, resulting in a list of points in local coordinate system, where the origin is the robot position location rp.

2. We then project the points to the global coordinate system by rotating by h and adding rp to each point, resulting in a list of points $\langle p_1, \ldots, p_n \rangle$.

3. The resulting FOV polygon vertices are $\langle rp, p_1, \ldots, p_n, rp \rangle$.

Some adjustments have to be made to the FOV polygon, so that it better represents the recorded image. A range scanner will typically have a wider field of view than a camera. To make sure we only use the map area that is covered by the image, the polygon is clipped to the actual camera FOV. Obviously, this is only an approximation of the visible map area, but in reality it provides adequate results.

In some cases the FOV polygon has to be clipped even more. The clipping approach described above ignores other camera-related factors, which could reduce the map area the camera is able to capture, compared to the area covered by the range scanner. Such conditions may include light conditions (if the image is very dark, the visible area is reduced), image resolution (could affect the maximal visible distance), and more. In this work we do not consider these camera-related factors, and assume that the approach described earlier produces a good approximation of the visible map area.

We note that the proposed method requires input from multiple sensors, in particular a camera and a laser range scanner. For simplicity, we refer to the major data categories: image (camera) and position (position, orientation, FOV polygon). We assume that the sensors are either producing output in fixed intervals (together), or generated in an unsynchronized manner but use a synchronized clock to label the output data with timestamps. In the former case, the sensory data can be used as-is. In the latter case, however, the sensory data has to be combined before it can be used within the interface. Note that having a synchronized clock is a reasonable assumption, as the sensors are normally placed on the same platform (the robot).

One simple approach to combine the two data categories, is to find which sensor is produced in a lower rate (the “slow" sensor). Then, we match each reading of the “slow" sensor with the nearest (in terms of timestamp) “fast" sensor reading. The unmatched remaining items from the “fast" sensor are discarded.

4.2 Finding images in the database

4.2.1 Point in polygon

An important task required by the system, is to find all polygons that cover (contain) the point of interest, p, provided by the user. We begin by describing common methods for testing whether a

point is inside a polygon. Then, we describe three methods to retrieve all polygons that contain the POI from the image database. Section 7.2 describes experimental results for these three methods on our main data set.

Testing whether a point is inside a polygon is a basic problem in computational geometry, and has several known standard algorithms [18,29,32]. We use the *crossing number* (CN) algorithm as described by O'Rourke [29], also known as the *ray-casting* method or the *even-odd rule*. CN was chosen due to ease of implementation and decent performance as compared to other algorithms [18].

Our initial, naïve, approach to find which of the polygons contain the point of interest, consisted of simply running the CN algorithm on all polygons. This method suffers from performance issues, as we test all polygons regardless of their relative position to the point of interest.

An improvement over this method requires some form of preprocessing. For each polygon we additionally keep the bounding rectangle, aligned to the vertical (y or "north") axis. Now, instead of running CN on all images, we run it only on polygons where the bounding rectangle contains p. This approach still requires testing of all polygons, but it performs better than the naïve approach. To conclude, for each polygon, we test whether p relies within the bounding rectangle. If it does, we continue testing with the CN algorithm.

An even further improvement is to use a spatial access method as the basis for data lookup. This reduces the number of polygons that are processed in response to each query. This approach is more efficient, and thus more appropriate as the number of images to be stored is larger. One example of a spatial data structure is R-Tree [17], a search tree that allows indexing of data by rectangles. When constructing the tree from polygons, we use their bounding rectangle as the key. Upon query for a point p, we first use the R-Tree to retrieve all polygons whose bounding rectangles contain p. Then, the CN algorithm from above is used in a similar manner to test whether p is actually contained in the polygon.

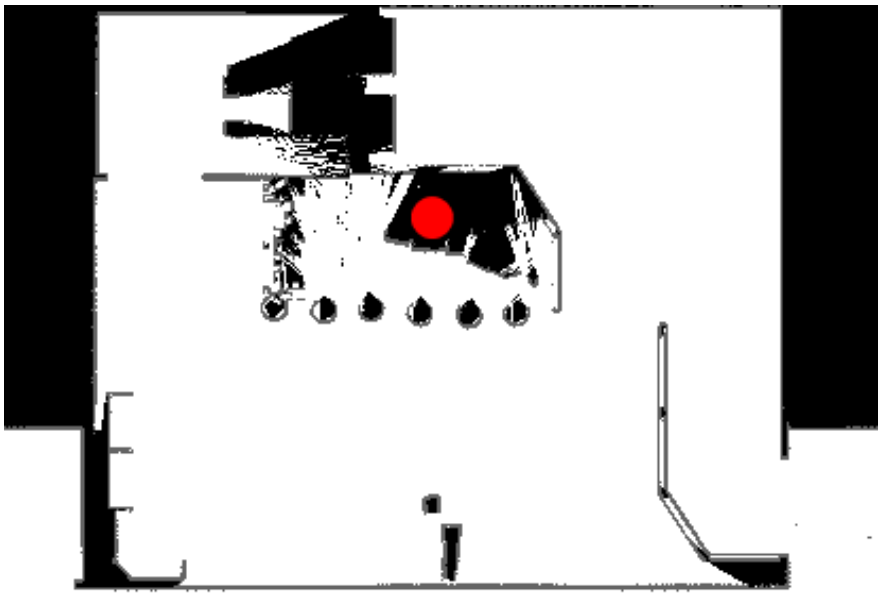


Figure 4.3 Point of interest (in light color) lies in an unknown area.

4.2.2 Dealing with unknown areas

Consider the case where a user requests images from a location, but no image covers it. This could happen, for example, if while recording the images the POI was obstructed by an object. Figure 4.3 illustrates this situation - the red dot represents the POI. The straightforward approach in this case would be to return no images, and let the user fine tune the point selection. However, this would only be one way of responding to this. There are others, and we demonstrate by discussing another.

One approach would be to scale up all FOV polygons by constant factor, to allow a larger area to be included in the "point in polygon" test. The scaling is done in advance, to allow using query methods that require preprocessing, such as the "bounding rectangle" method. This allows more flexibility when selecting unknown areas. This approach is easy to implement, but could cause unwanted images to be returned for a search query, images that do not actually cover the point of interest.

4.3 Ranking retrieved images

4.3.1 Ranking process

Normally, the selected point of interest would be covered by many images, possibly too many. For example, in the reported experiment a given point was covered by a range of 10–300 images. Most of these images are redundant, as they cover overlapping areas. In order to decrease the number of images the operator has to view, to conclude whether a victim is present at the point of interest, we apply a ranking process. The whole process may be described as follows.

1. Find all images that cover the point of interest (p).
2. Group the images in sectors, by robot heading and sector resolution r.
3. For each sector, compute the utility value u of all images, and select the image with best u.

The ranking process produces two sets of images: *best*, which contains the highest-ranked images of each sector, and *other*, which contains all other images that cover p. The highest-ranked image of *best* is displayed automatically, while the rest of the images are available for selection on the map. All images, including the *other* images, are displayed as thumbnails and ordered by their utility value. Figure 4.4 illustrates how the *best* and *other* groups fit within the interface.

4.3.2 Grouping

To provide the user with different perspectives of the POI, we group the images by their view angle. For each image we compute an angle θ, between the POI and the robot location while recording that image. The images are then grouped in sectors, considering the θ value and the resolution parameter r.

The grouping process is illustrated in Figure 4.5. The red circle is the point of interest, selected by the user. The blue dots represent the images that cover the POI. Images are grouped in sectors

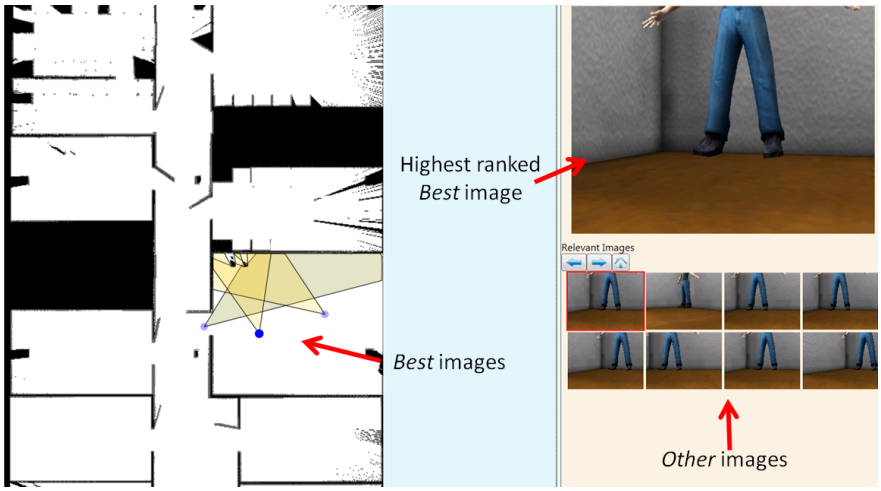


Figure 4.4 Ranking process output example

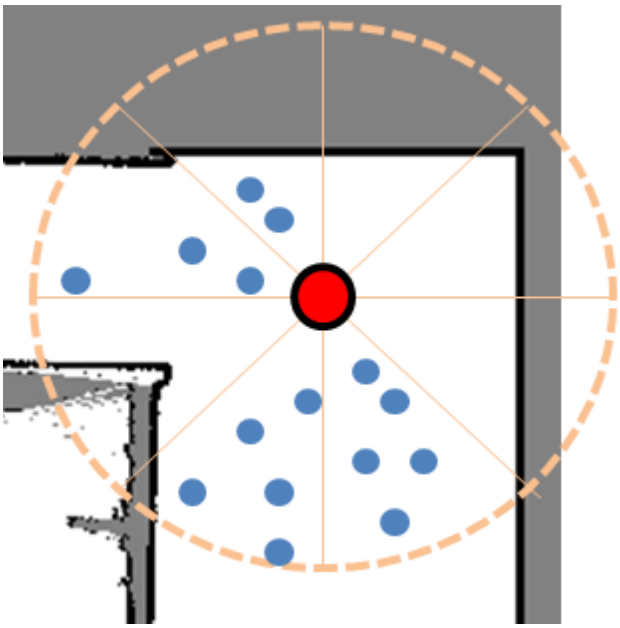


Figure 4.5 Grouping Images

by their θ angle, and the actual size of each sector is determined by r. We then pick one image from each group to the *best* group, and gather the rest in the *other* group.

4.4 Image utility

4.4.1 Utility value computation

For this study, we have chosen to consider the following attributes when computing the utility value:

1. Maximize the image area (formed by the range scanner readings).
2. Minimize the distance from the POI.
3. The image should be centered as much as possible. We define an image to be fully *centered*, if the robot faced directly towards the POI while recording. We try to minimize the following value:

$$centered[image] = |heading[image] - \theta|$$

Finally, we use a linear combination of the attributes:

$$u[image] = w_1 \frac{area\,[image]}{area_{max}} + w_2 \frac{distance_{min}}{distance\,[image]} + w_3 \frac{centered_{min}}{centered\,[image]}$$

Appropriate weight values for the reported experiment were set in a pilot session. Note that the minima and maxima are computed over each group (sector) separately.

This is only one example of a utility function for our method. Other functions, that consider different environmental parameters or are calculated differently, can be used without major changes to the other components of the system.

4.4.2 Performance

The utility function that was presented above is a heuristic, and can fail under certain conditions. Figures 4.6 and 4.7 show two examples where the *best* image, that was automatically selected by

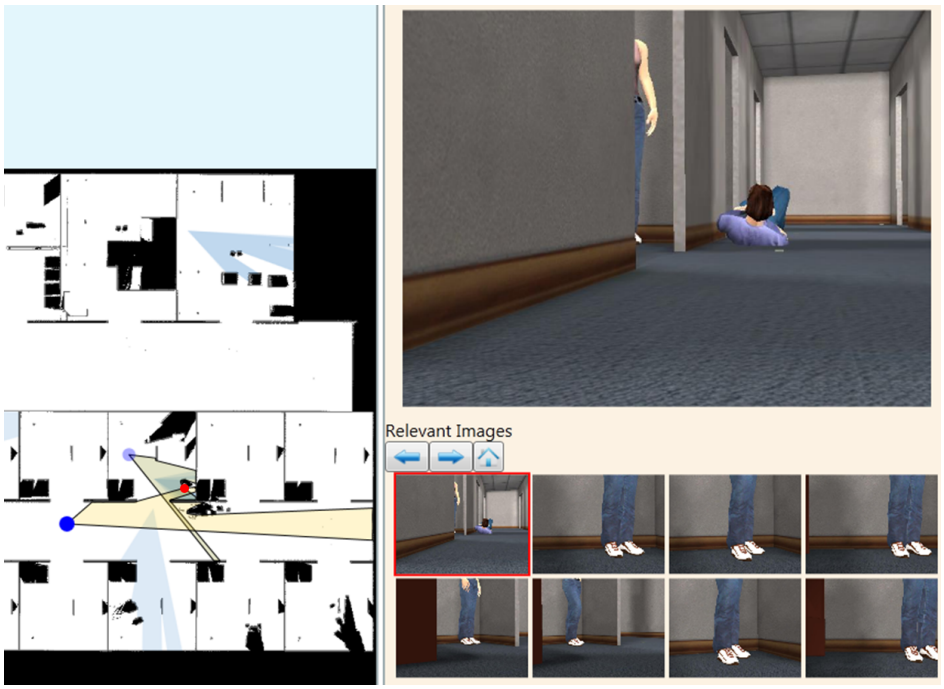


Figure 4.6 The *best* image in this case is the upper one (on the map), but another image was selected instead. Note how the image is centered, covers a large area, but does not provide a very good view of the POI. In fact, all "relevant images" in this figure are actually better than the *best* image.

the system, is not what the user would expect as the best image that cover the POI. In each of the examples, the light dot represents the POI.

Note that even if the *best* image does not provide a good view of the POI, the user can still select another image that covers it, from the map or from the "Relevant Images". Section 8.2 contains some practical suggestions for improving the prioritization of images.

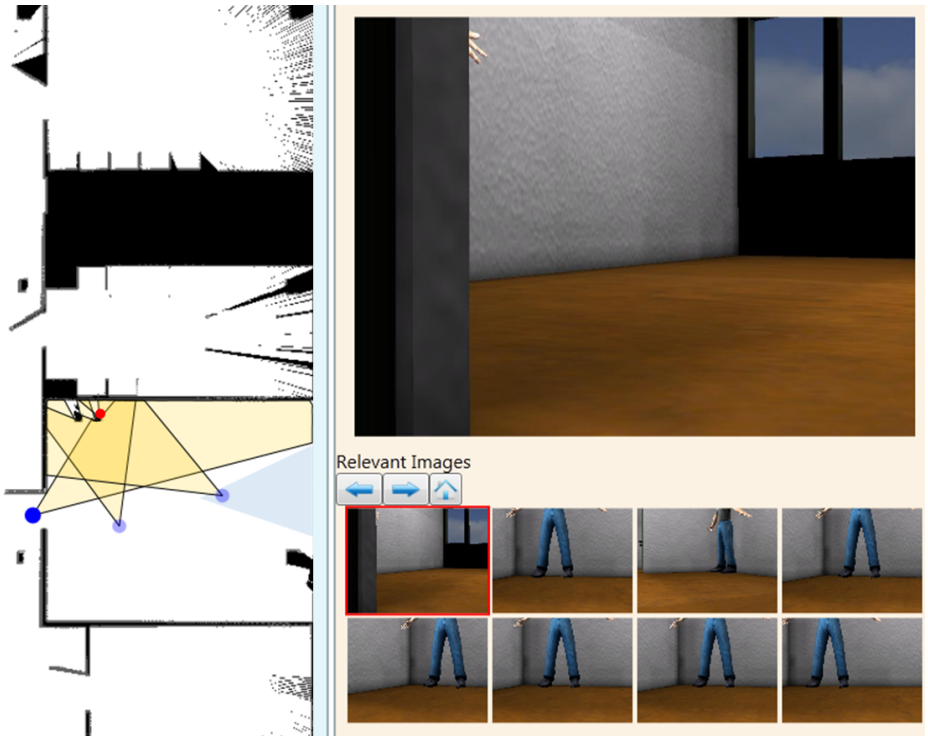


Figure 4.7 The *best* image in this case is probably the middle one (on the map), but another image was selected instead. This is probably due to the *area* consideration, taking over the *distance* and *centered* components.

Chapter 5

Interface Evaluation Experiment

In this chapter we describe an experiment that was conducted in order to evaluate our interface. The experiment compared performance in a search mission, of users that used our interface (*POI*) and an existing asynchronous interface that we call *Best-First*. In Section 5.1, we describe a simulated environment that was generated using the USARSim simulator. The experiment was conducted using pre-recorded data. Section 5.2 describes what data was recorded from the simulator and how it was processed. In Section 5.3, we discuss the experimental conditions: the interfaces that were compared, recruiting participants, and how the experiment was structured. Finally, Section 5.4 describes specific implementation details and decisions.

5.1 Simulated environment

The reported experiment was conducted in a simulated environment, generated by USARSim. USARSim is a high-fidelity simulation package of robots and environments, based on the commercial Unreal Tournament game engine. It has been shown to accurately model robot behavior and sensors, in particular camera video and laser range scanner [7, 8]. This is the same environment in which the *Best-First* interface was tested [39].

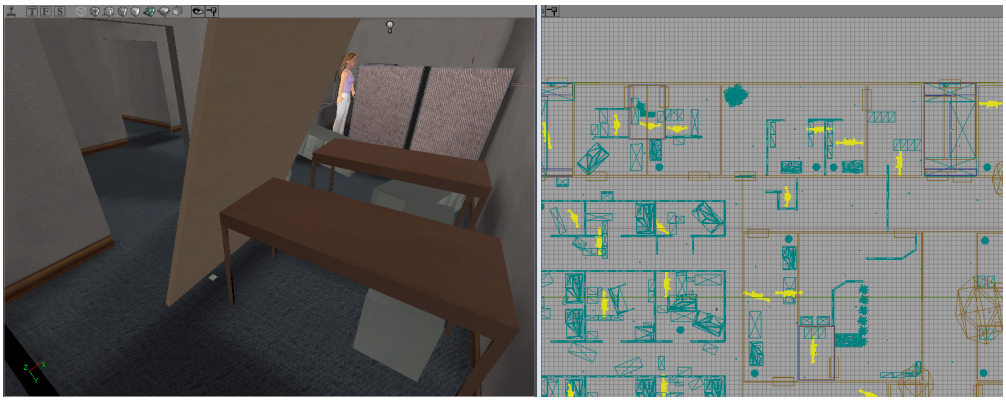


Figure 5.1 Placing objects and victims using the environment editor.

Two USAR environments were created, based on an office environment from the 2006 RoboCup Rescue Virtual Robots competition finals [1]. The RoboCup Rescue project promotes research and development of robots for disaster management, and Virtual Robots is part of the Simulation project.

The original environments were modified using the built-in *UnrealEd* environment editor, by placing additional objects such as desks and cabinets, in order to make them look more like a disaster scene. 20 human-like characters ("victims") were placed in each environment, in rooms and corridors, and in various postures. Up to two victims were placed in each room, most were put behind objects, but some were placed in open spaces. Visual inspection verified that each victim is visible from the height of a robot camera. Figure 5.1 illustrates placing objects and victims using the environment editor.

5.2 Data recording and processing

Since both of the interfaces we compared are asynchronous, and robots are assumed to be fully autonomous, we could change the search mission to use pre-recorded data instead of "live" robots.

[1] Map *compWorldDay5b* from the *DM-Robocup06Worlds V3.31* package

Participants were given the entire map and image database of an environment, after a robot explored it. In addition, means for direct teleoperation were not provided, as it would be needed mainly for contingencies (e.g. stuck robots). An advantage of this evaluation approach is that it precludes differences between operator decisions on robot movements, and between different runs where the autonomous robots may make different decisions.

5.2.1 Recording

In order to generate data for the experiment, a simulated robot was manually steered in each of the environments, while recording sensory data. Steering was done at a modest pace, with each environment taking 60-90 minutes. Robot steering was done using *Iridium*, a tool which is bundled with USARSim and allows joystick control of robots. The simulated robot was modeled after the EyeDrive [1] by ODF Optronics. Although the real EyeDrive has additional four side cameras, we used only the front camera, for compatibility with other (more standard) robots.

The sensory data that had to be recorded from the simulator, consisted of camera imagery, range scanner readings and wheel odometry. In USARSim, camera imagery is available through a different interface than the rest of the sensors. In order to synchronize the recorded data later, we set a timestamp on each sensor reading upon arrival. The timestamps were generated using the operating system clock, at the highest resolution available. Sensory values were not generated at the same rate; camera images were produced at 10Hz, while range scanner readings and wheels odometry were sent together at about 4Hz. To conclude, the recording sessions resulted in two data sets: one that contained timestamped images, and another that contained timestamped position sensors - range scanner readings and wheels odometry.

Environments differed in area size, the number of recorded images and the average image space covered by a victim ("visibility", explained in section 5.2.3). The differences are summarized in Table 5.1.

	Victims	Images	Area (m^2)	Mean visibility	Visibility SD
Environment 1	20	3209	327	5.01	16.76
Environment 2	20	4579	483	9.91	30.45

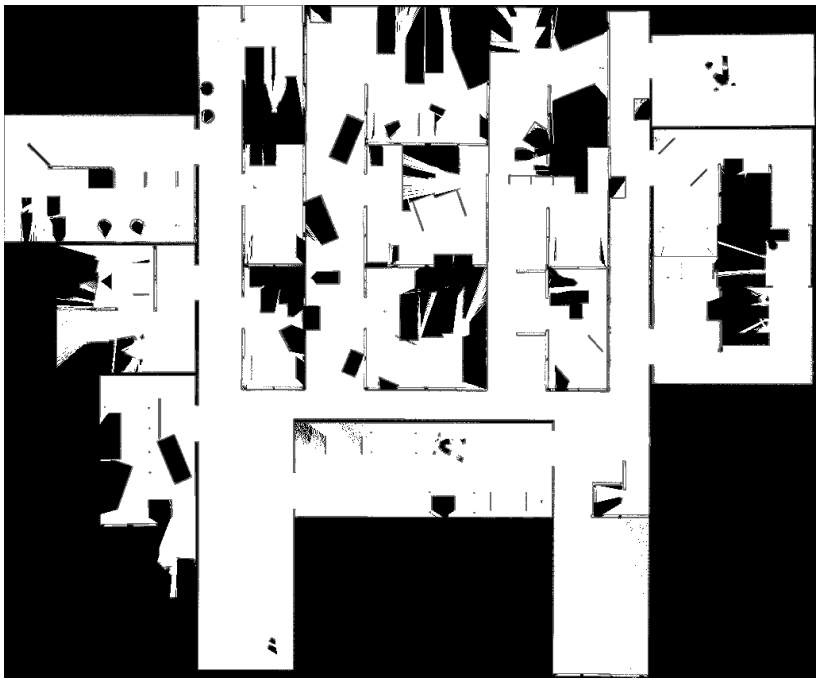
Table 5.1 Summary of environments parameters

5.2.2 Map generation and image database

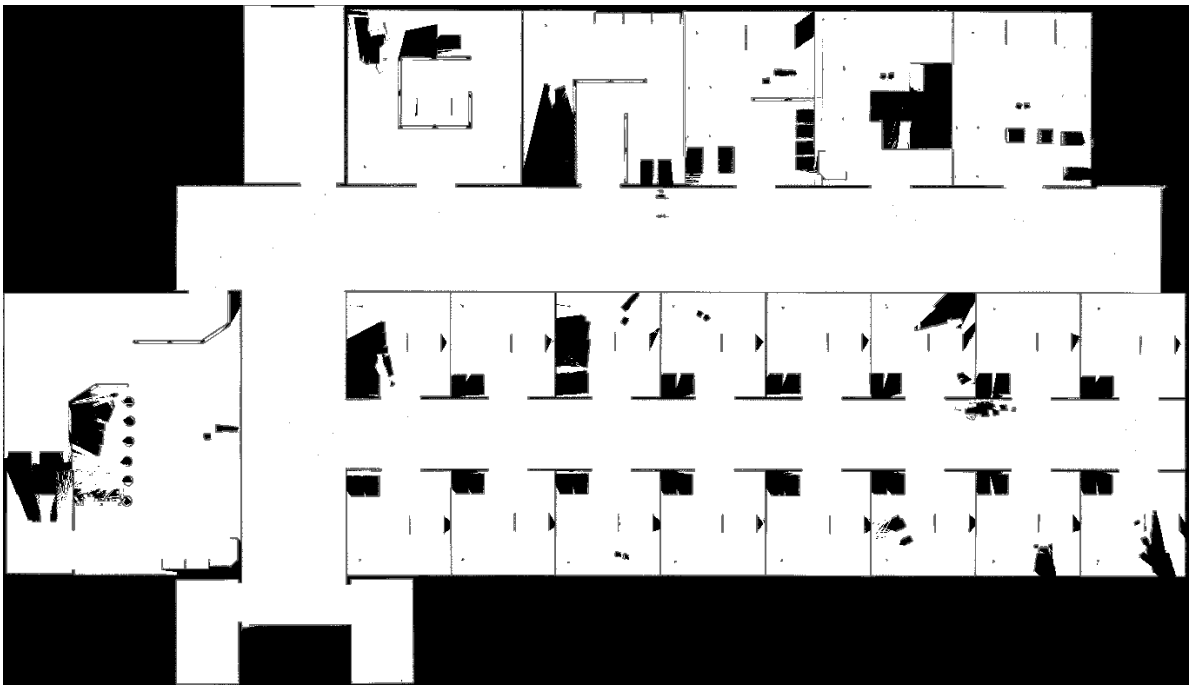
In order to transform the received position sensors data to a 2D map, we used an off-the-shelf software library called Karto SDK [2], by SRI International. Karto provides a SLAM [15] mapper module, which creates a map given a set of localized range scans. The mapper is implemented using a scan-matching algorithm and can correct noisy sensory data. For each set of readings, we produced a map image, along with a list of robot positions, while maintaining the timestamps from the recording session. The maps were generated with a resolution of 2.5cm/pixel, and are presented in figures 5.2a and 5.2b.

The next step was to combine the camera and position sensory data, to form the image database, as discussed in Section 4.1. As camera imagery was generated at a faster rate, we had to discard some of the images. Range scanner polygons were projected from local coordinates to map coordinates (robot position + point, for each polygon vertex), and then clipped to the camera FOV (45°). The final image database contained entries with the following data:

- The image itself.
- FOV polygon, in map coordinates and clipped to camera FOV.
- FOV polygon area and bounding box.



(a) Environment 1 (smaller) - rotated 90°



(b) Environment 2 (larger)

Figure 5.2 Generated maps. White indicates clear space, while black indicates occupied space (walls, obstacles, etc.).

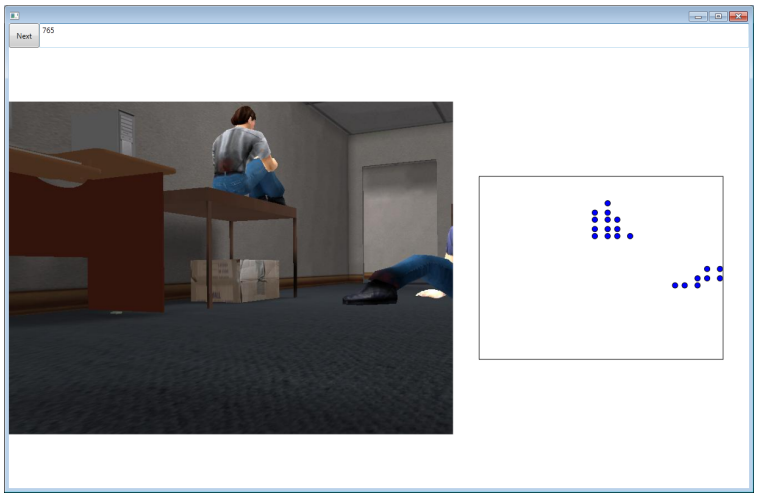


Figure 5.3 Visual inspection of victim hit points. The right square shows all victims hit points that were found for a particular image. Hit points appear as dark dots.

5.2.3 Victims visibility

To conclude whether a certain victim was seen but not marked, we calculated a *visibility* value for each image, to determine which parts of an image are covered by a victim. Image visibility was computed by performing ray casting [30], starting from the camera center position and aiming at all directions within the camera field of view. Ray casting was performed at a resolution of two degrees, since lower resolution values resulted in severe simulator performance issues. The actual ray casting implementation was based on the USARSim *VictSensor* module, with minor modifications to allow for better fidelity in the reported hit points. The first hit for each ray was tested for whether it represents a victim or another object (e.g. a wall). The image *visibility* value was defined as the number of victims hit points. In total, $23 \cdot 23 = 529$ rays were cast for each image, obtained by dividing the camera FOV (45° for both the horizontal and vertical axis) by the resolution of 2°. To verify the results, we manually inspected all images and their corresponding victim hit points. Figure 5.3 shows the tool that was used to inspect the images.

Due to simulator performance issues, ray-casting could not be run during the data recording sessions, and had to be performed separately from the actual recording of data. To support record-

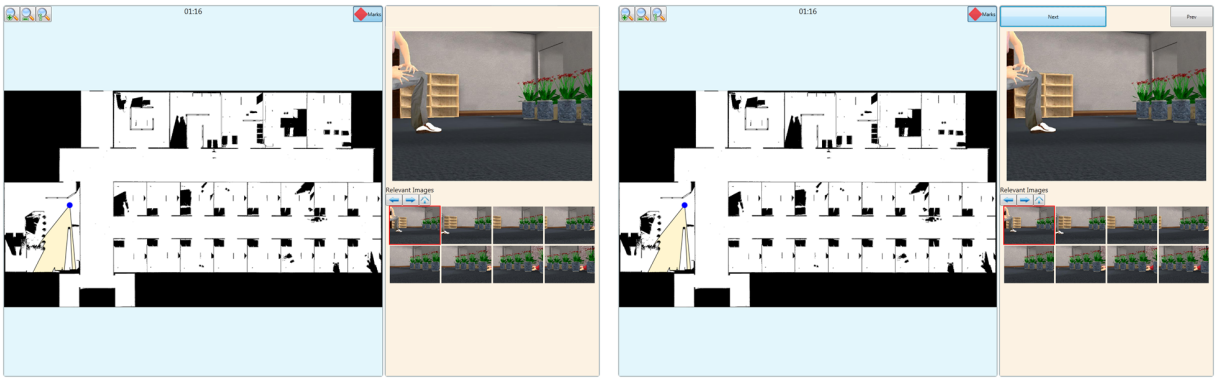
ing of additional data after the original recording session was over, we saved the *ground truth* position of each image while recording it. In USARSim, the *ground truth* position is the actual robot position in simulator coordinates. Later on, this value was used to place the robot in the exact position from which an image was recorded, and then run the ray-casting process for that image. Since the environments had no moving objects, and the *ground truth* position represents the real simulated robot position (verified by visual inspection), we can conclude that the recording of additional data was made under the same conditions as the original recording session.

5.3 Experiment design

5.3.1 Asynchronous interfaces

To evaluate our interface for USAR missions, we compared it with an existing asynchronous interface by Wang et al. [39], originally called *Image Queue*. We consider *Image Queue* to be the current state of the art of asynchronous interfaces for USAR missions, intended for a fully-autonomous multi-robot system. We have implemented this interface as described by the authors, without any major changes. The main differences from our interface include:

- The map was not used to navigate between images, but only to identify and mark victims.
- Navigating between images was done with two "Next" and "Previous" buttons, and the images were ordered by a utility value. The utility value was computed by the image area that wasn't already seen.
- The thumbnails display images were ordered chronologically, to provide contextual information for the currently selected image.
- The *history* map layer (mentioned in the previous section) was not provided, in order to stay



(a) *POI* interface **(b)** *Best-First* interface

Figure 5.4 The evaluated interfaces, presented side-by-side. The only visible difference between the interfaces is the presence of the "Next" and "Prev" buttons, at the top right corner.

as close as possible to the original interface. The layer is less significant for the *Best-First* interface, because visited areas are ranked lower than unvisited areas.

We refer to this interface as *Best-First*, since the next image to display is the "best" image, in terms of high utility value. Figure 5.4 shows our implementation of the *Best-First* interface, along with the *POI* interface.

5.3.2 Conditions and procedure

The experiment followed a between-subjects design, with each participant using only one of the interfaces for both of the environments. The order of environments was counterbalanced, for the sake of minimizing order effect [10] between sessions.

To begin the experiment, participants read an instructions manual for the chosen interface, and performed a training session using a dedicated training map. Participants had to successfully mark three victims in the training session before advancing to the other sessions, in order to ensure they understand how to perform the missions to follow. Afterwards, participants were given 10

Figure 5.5 Four participants during the experiment, some of them were still reading the instructions manual.

minutes in each of the environments to locate and mark as much victims as possible. The order of environments for each participant was determined prior to the experiment.

After each of the two sessions (one for each environment), participants were asked to complete the NASA-TLX [20] questionnaire, which is a subjective assessment tool for rating perceived workload. We have chosen to use a modified version of the test, known as Raw TLX (RTLX) [19], since it is easier to apply and requires less time to complete, amongst other advantages [6]. Upon completion of all sessions, participants were given a questionnaire to conclude their experience with interface.

The experiment was conducted in a dedicated room, free from distractions. Up to four similar (in terms of hardware) machines were used during the experiment, each running the Windows operating system. The machines, monitors and input devices were placed next to each other, but great care has been taken to ensure that participants who used the same interface did not sit next to each other. Figure 5.5 illustrates the room during the experiment.

5.3.3 Participants

32 adult students were recruited from Bar-Ilan University by placing advertisements throughout the campus. Students were asked to fill an online form and select available dates for participation. Participants were to be divided by three factors: interface to use (*POI* or *Best-First*), environments order (1,2 or 2,1) and gender. There were eight groups in total, as each of these factors consisted of two values.

Students who asked to participate in the experiment were assigned to a list of 32 "slots", with each slot specifying one of the eight groups from above. The list of slots was created by randomly shuffling a collection of the eight groups, with each group appearing four times. The actual assignment of a participant to a slot was done on a first-come basis, and according to the gender.

32 participants were recruited to this study. The average participant age was 23.2 ($SD = 3.8$). Participants reported using a computer for an average of 6 hours per day ($SD = 3.2$), and a smartphone device for an average of 4.4 hours per day ($SD = 4.7$). The average reported daily duration for playing video games was 0.5 hours ($SD = 1.2$). None of the participants had prior experience with robot systems.

Participants were compensated in cash for their time and effort. They received a fixed show-up fee, along with a variable bonus based on the number of found victims. In total, 40 New Israeli Shekels (NIS) were given as the show-up fee, and up to 20 more NIS as the variable bonus. The bonus was computed from a table of five categories for the number of found victims, each specifying the portion of the maximal variable bonus to hand out. Table 5.2 illustrates how the bonus was allocated. After a participant completed the experiment, the number of found victims was computed and the bonus was given according to the table. The technique for computing the number of found victims is described in Section 6.1.1.

During the experiment, five of the participants who used the *Best-First* interface did not perform the missions as instructed. Instead of using the "Next" and "Prev" buttons to navigate between

Found victims	% of maximal bonus (20 NIS)
0-3	0%
4-7	25%
8-11	50%
12-15	75%
16-20	100%

Table 5.2 Variable bonus for participation

images, they have relied almost exclusively on "Relevant images" by rapidly switching between nearby images. This is the equivalent of watching a video stream from one robot, and was not a part of the experiment. Therefore, we have discarded these results and recruited the same number of people to replace them, using the same method and under the same conditions as described in this chapter (5). The newly-recruited participants performed the missions as instructed.

5.4 Interface implementation

5.4.1 POI implementation

In chapters 3 and 4, we presented several components and methods that can be implemented in more than one way. We now describe the choices that were made for implementation of these components.

- Point in polygon - the Bounding Rectangle method was used, in which the polygons are kept in a list, and points are first tested to be contained in a polygon bounding rectangle, before further testing. Section 7.2 discusses the performance of point in polygon methods on our data set.

- Combining data from multiple sensors - the simulated sensors were generated at a different rate, with the camera being generated faster than the laser range scanner. Each laser scan was combined with the nearest (time-wise) image, and excessive images were discarded.
- Unknown areas - to deal with POIs requested from unknown areas, all FOV polygons were increased by a small constant factor. We found this method to perform adequately on our data set, while not producing too many wanted images.

5.4.2 Logging of user operations

All meaningful user operations within the UI were recorded for analysis purposes. The following list summarizes the operations that were recorded, along with the relevant interface:

- Mark operations - adding and removing marks (both interfaces).
- Image selection - selected image from all UI components: map (*POI*), next/prev buttons (*Best-First*) and "relevant images" (both interfaces).
- Map operations - pan and zoom (both interfaces).
- POI selection - *POI* only.

Chapter 6

Interface Evaluation Results

In this chapter, we present results from the interface evaluation experiment that was described in the previous chapter. In Section 6.1, we discuss some methods that were used to analyze the results, and define several metrics to compare them. Section 6.2 discusses results that are related to victim marking. In Section 6.3 we present results related to image selection. Section 6.4 contains results related to how participants navigated between images, depending on the interface. Section 6.5 presents results from the subjective assessment tools that were used - perceived workload and general experience. Finally, we discuss our findings in Section 6.6.

Unless specified otherwise, the statistical tests in this chapter that compare the *POI* and *Best-First* interfaces, were performed using a two-sample t-test. Box plots in this chapter follow the following convention: the upper and bottom lines are the first and third quartile (the 25th and 75th percentiles), and the line inside the box represents the median.

Figure 6.1 Map with bounding boxes around victims

6.1 Measuring success and failure

6.1.1 Map marks

Prior to the experiment, a list of bounding rectangles for victims was made for each environment. The bounding rectangles indicated the map areas considered as victims, and were created by matching victim locations in the USARSim environment editor with the 2D map generated by Karto. Figure 6.1 shows a map with the respective bounding rectangles around victims. Note that the victims were placed in different postures (standing, crouching, lying), which explains why the map area they occupy is different.

Each mark made by the participants was assigned to the nearest bounding box. If the distance was less than 1m, the mark was considered successful for that victim. We chose an accuracy value of 1m, and verified that it does not produce any anomalies with the experiment data. Marks were categorized into one of the following categories:

1. *found*, first successful mark for a victim.

2. *duplicate*, successful mark for a victim that was already marked.

3. *false positive*, a mark that could not be assigned to any of the victims.

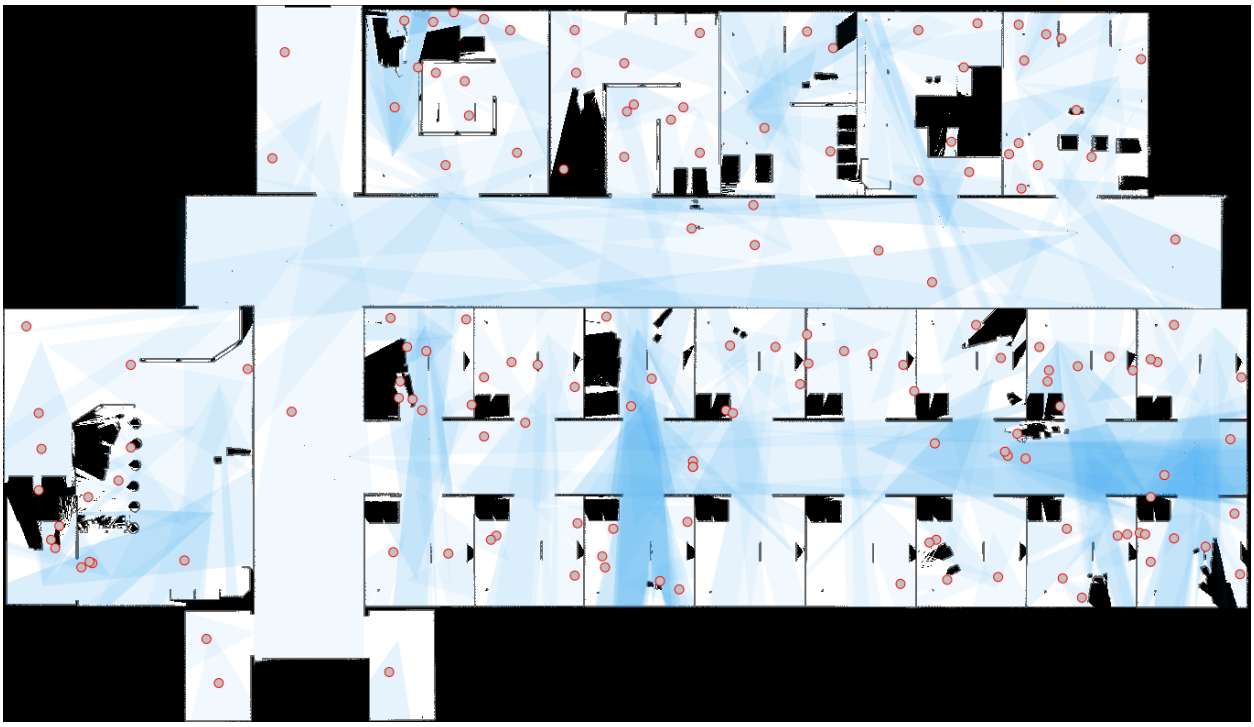
4. *false negatives*, which considers victims that were seen by the participant sometime throughout the session, but were never marked. To compute this measure, we used the victim visibility data (described in Section 5.2.3) of the images that were seen by the participant. With that data we were able to determine which victims were present in images, but not marked.

6.1.2 Analyzing individual participants

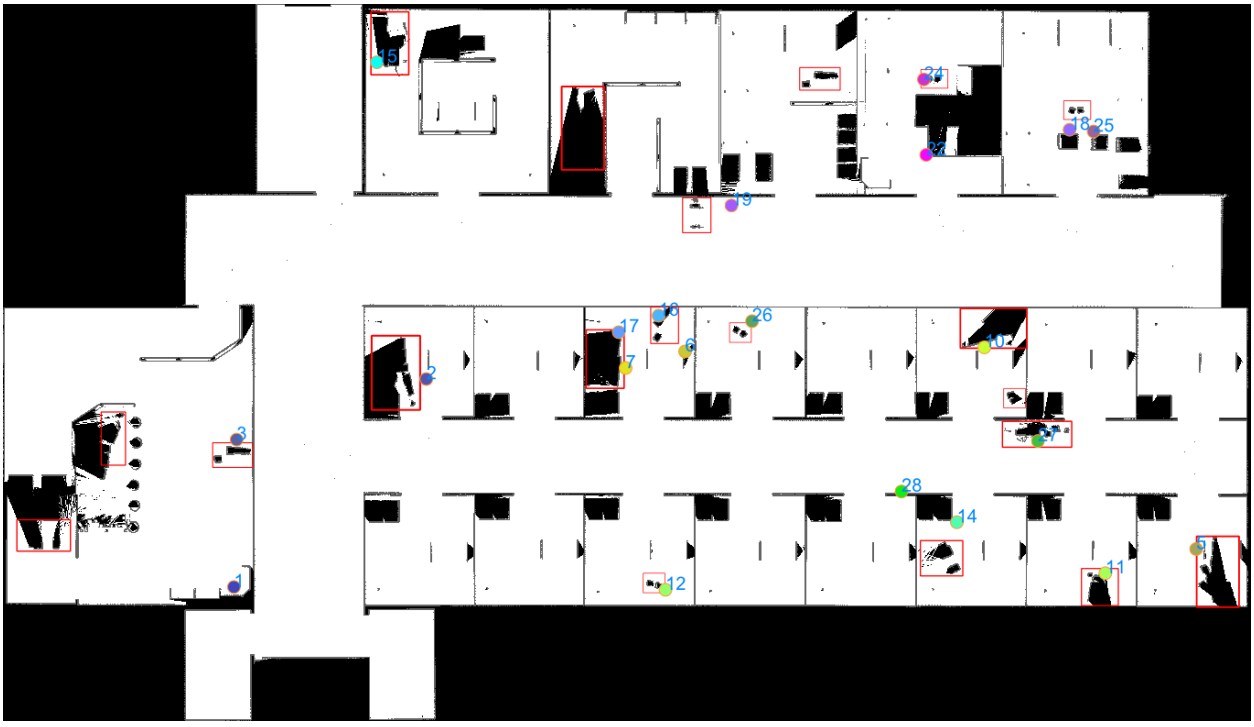
For analysis purposes, we have looked into the results and mission progress of individual participants. This was done in order to find additional methods of comparing between different participants, rather than just their marks results. In particular, we visually analyzed individual participants in two ways:

1. Area coverage (Figure 6.2a) - shows which areas have been viewed by a certain participant, and how many times they were viewed. For participants who used the *POI* interface, it also includes the points of interest. This was used mainly to compare the performance of different participants.

2. Marks (Figure 6.2b) - shows all marks that were made by the participant. This was used mainly to ensure that the *found*, *duplicates* and *false positives* counts are correct. However, it can also be used to learn about a certain participant marking behavior, which might be useful during training.



(a) Area coverage for an individual participant that used the *POI* interface. The grey dots represent all points of interest throughout the session. Dark areas were viewed more than light areas.



(b) All marks made by an individual participant. A circle represents one mark, and the color represent the time of marking.

Figure 6.2 Analyzing individual participants

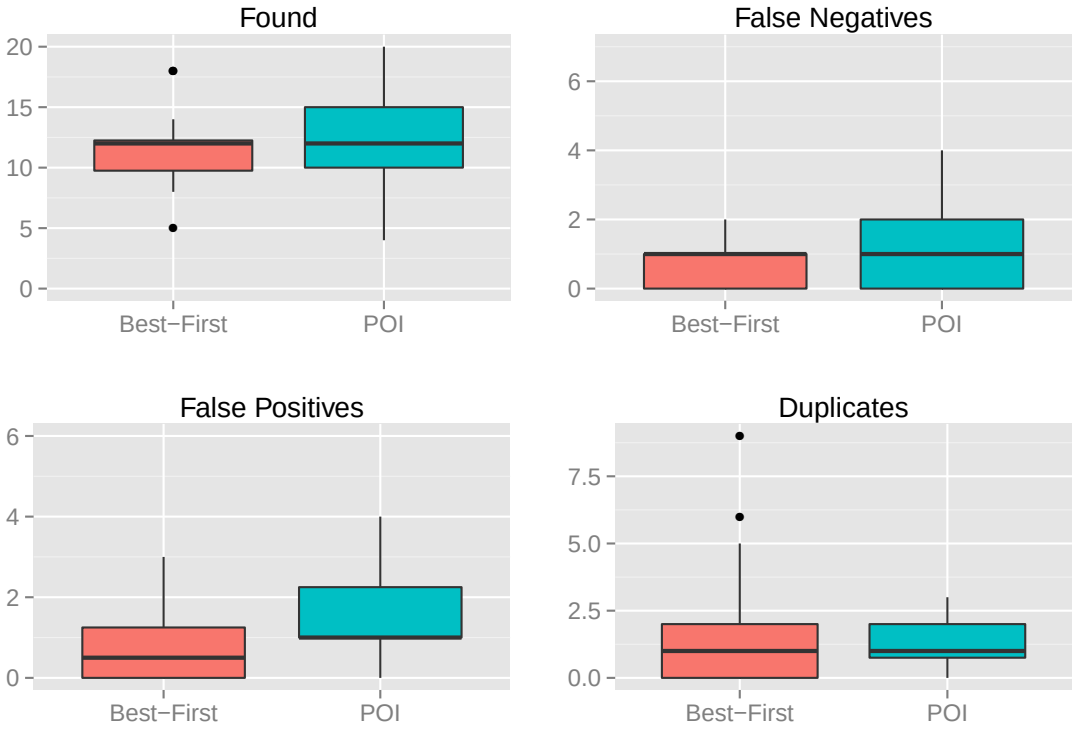
6.2 Results

The overall aggregate results are shown in Figure 6.3. In the figure, we see the measures from Section 6.1 for both of the environments. The horizontal axis specifies the interface (*POI* or *Best-First*), and the vertical axis specifies the number of marks. For example, in the "Found" chart, the vertical axis specifies the number of marks that were considered as the first successful mark for a victim.

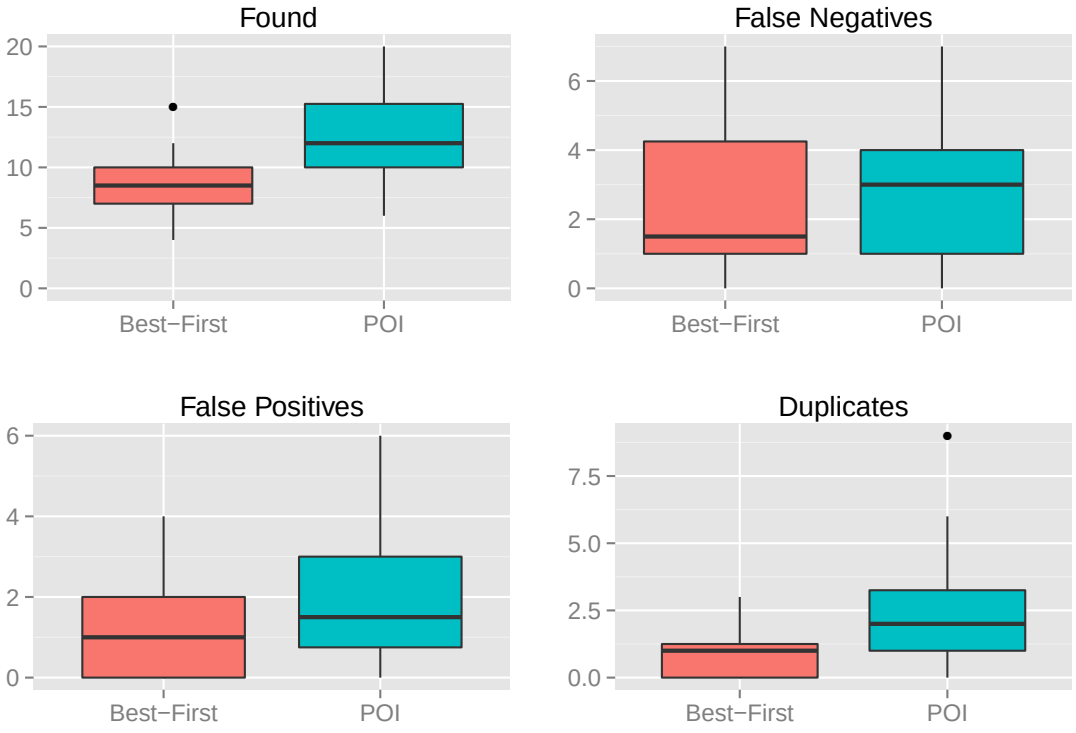
Environment 1 Participants in the *POI* and *Best-First* conditions found on average 12.4 and 11.5 victims respectively. We found no significant difference ($p > 0.1$) in any of the measures, as can be seen in Figure 6.3a.

Environment 2 Overall, participants in the *POI* condition had more success marking the victims, than those in the *Best-First* condition. As shown in Figure 6.3b, the number of *found* victims in the *POI* condition was significantly higher $(t(30) = -3.24, p < 0.003)$ finding 12.7 and 8.6 victims on average respectively. There were also significantly less *missed* victims $(t(30) = 4.00, p < 0.001)$, averaging at 4.3 and 8.5 respectively, meaning that participants in the *POI* condition have seen more victims. However, the number of duplicate marks was significantly higher $(t(30) = -2.31, p < 0.028)$, averaging at 2.4 and 0.9 respectively. The number of false positives did not differ significantly $(t(30) = -1.44, p > 0.16)$. Note that the number of degrees of freedom (30), is due to the number of participants (and samples), which was 32.

Task progress Figure 6.4 illustrates task progress over time for both of the environments, by displaying the number of correct marks throughout the session. Each colored line represents the number of found victims for a different participant, as a function of time. The black line is a second order polynomial regression line. In the second environment, the number of correct marks for *POI* condition increased in a higher rate than the *Best-First* condition, as illustrated by the regression



(a) Mark results - Environment 1



(b) Mark results - Environment 2

Figure 6.3 Marks results

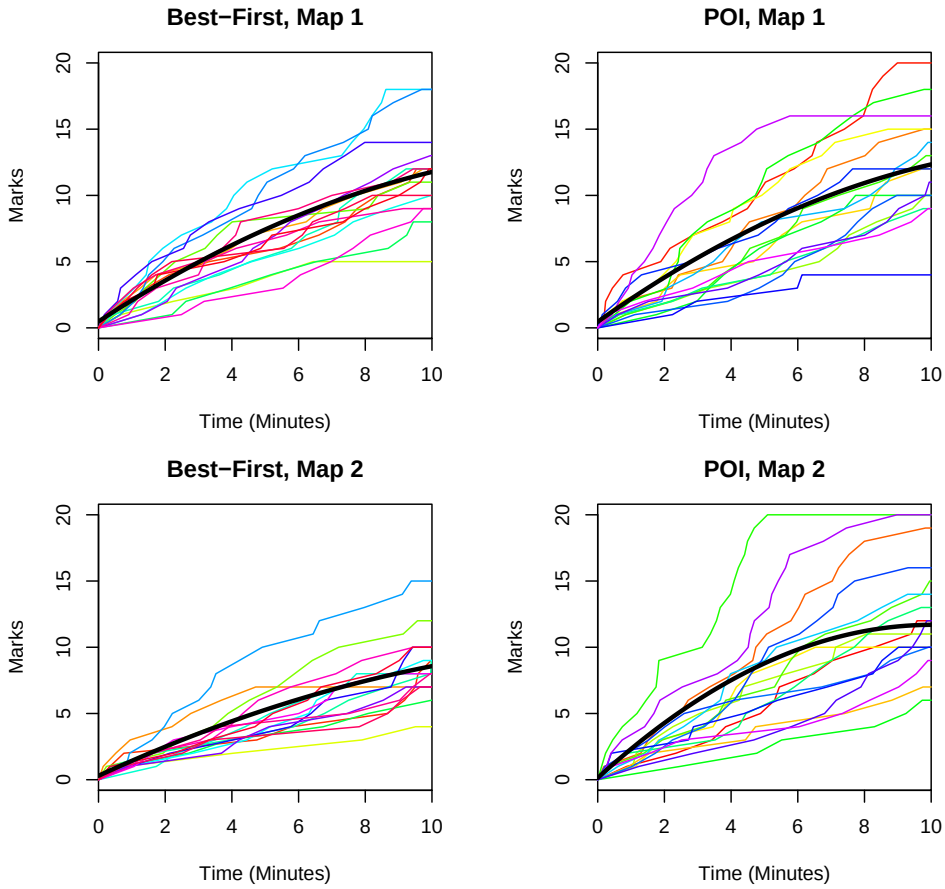
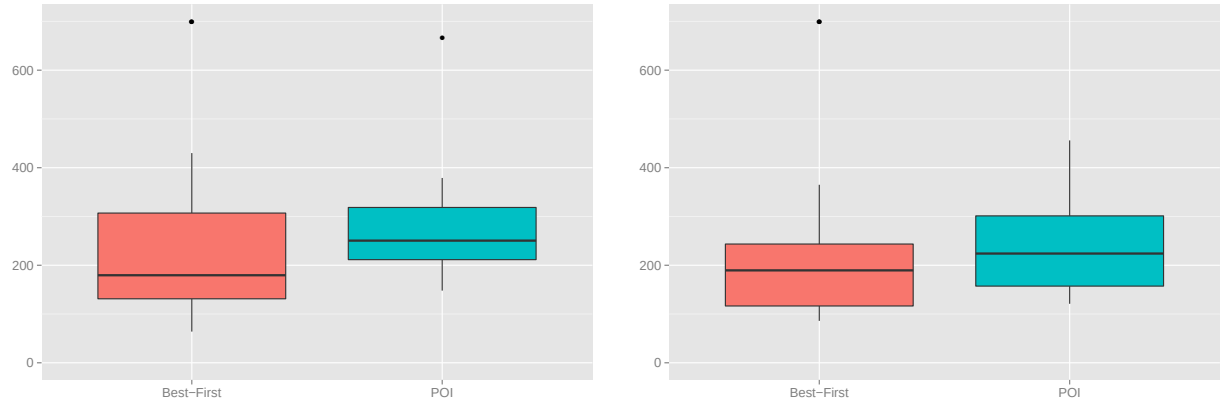


Figure 6.4 Correct marks over session time. Each colored line represents a different participant, while the black line is a second order polynomial regression line.

line. In addition, correct marks over time of participants in the *POI* condition had higher variance, compared to *Best-First* participants.



(a) Environment 1 **(b)** Environment 2

Figure 6.5 Number of seen images

6.3 Image selection

6.3.1 Seen images

Figure 6.5 shows the number of seen images for each of the interfaces. We found no significant difference between the interfaces in the number of seen images ($p > 0.7$). In general, most participants watched 150-300 images

Another measure which is related to image selection is the time spent on each image, counted from the moment an image was displayed and until another image was selected. Table 6.1 shows the mean and standard deviation of the time spent on images, considering both environments and interfaces. We found a significant difference between *POI* and *Best-First* in the second environment $(t(1325) = 4.85, p < 0.00001)$, but found no significant difference for the first environment $(t(1439) = 1.29, p < 0.196)$.

Environment	Interface	Mean	SD
1	*POI*	12.4	19.2
1	*Best-First*	13.7	19.3
2	*POI*	11.9	17.3
2	*Best-First*	17.9	27.7

Table 6.1 Summary of time spent (in seconds) per image.

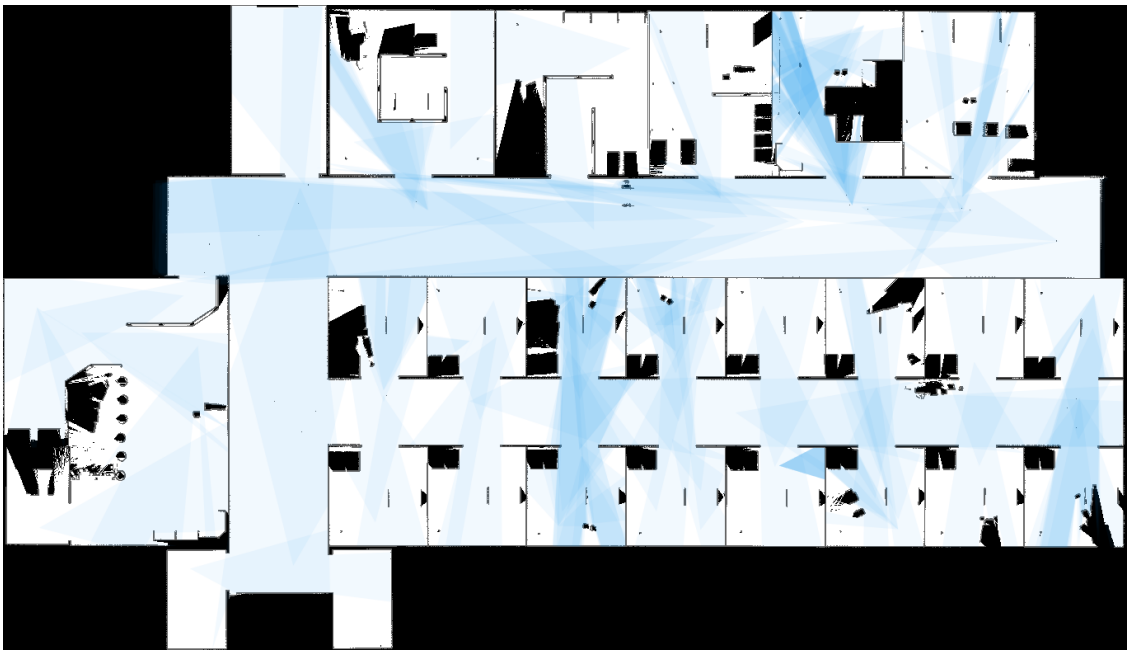
6.3.2 Map coverage quality

Consider Figures 6.6b and 6.6a, each representing the map coverage of two different participants. Covered areas appear in blue color, and the color intensity represents the number of times an area was viewed (darker means more views). While the participant at figure 6.6a covered most of the map without repeating too many areas, the participant at 6.6b focused on a small portion of the map and viewed it repeatably. It is no surprise that the former found 15 victims, while the latter found only 4. On a larger scale, we used two measures to check whether one interface allows users to better cover the map.

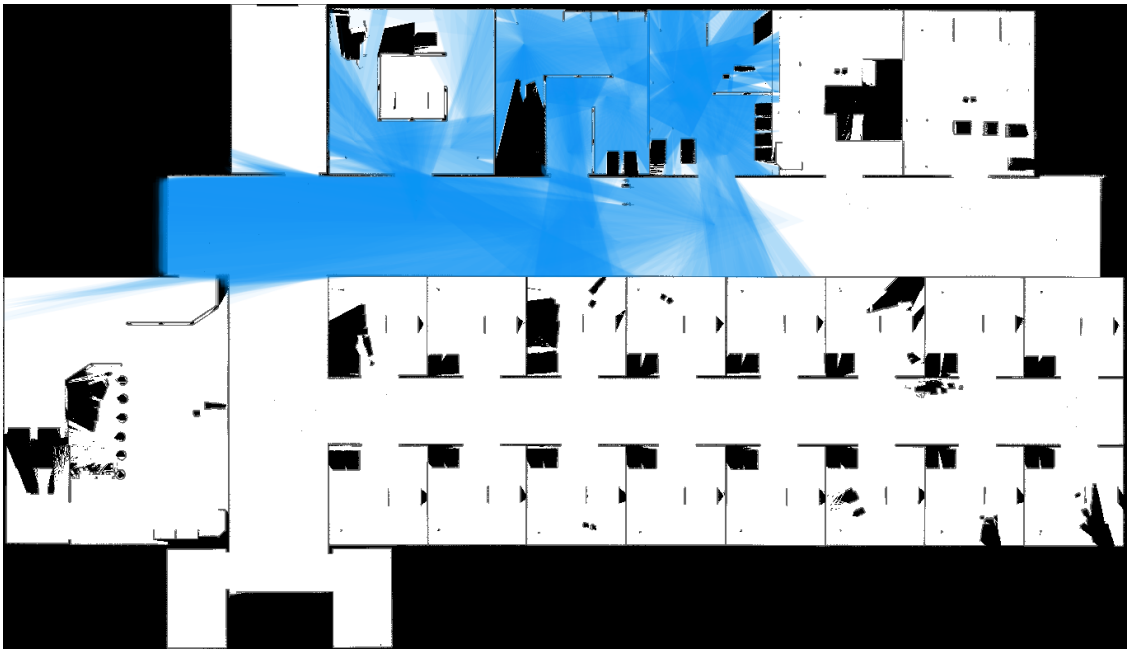
1. *coverage* - the percentage of visible map area that was actually seen.

2. *entropy* - a measure of how balanced the coverage is. A map is covered in a completely balanced manner, if each map point was viewed the same number of times.

For each map and participant, we computed a list of counters $c_0, c_1, \ldots c_n$, where c_i is the number of map points that were viewed exactly i times. $c_1, \ldots c_n$ were computed by using the set of images that was viewed by the participant, and c_0 by subtracting $\sum_{i=1}^{n} c_i$ from the number of visible map points (based on the union of all image polygons from that map). From here, *coverage* was computed by simply dividing the counters by the total number of visible map points, and the *entropy* was computed using Shannon's entropy H (normalized).



(a) High quality map coverage. Found victims: 15.



(b) Low quality map coverage. Found victims: 4.

Figure 6.6 Map coverage analysis of two participants.

Figure 6.7 summarizes the results over both interfaces and environments. In the first environment, *POI* participants covered 73% of the area on average, while *Best-First* participants covered 61%. We found a significant difference in this case $(t(30) = -2.19, p < 0.036)$, and no significant differences in any of the other cases ($p > 0.28$).

Figures 6.8 and 6.9 provide more insights on the results, by presenting the *coverage* and *entropy* measures over session time. Again, each colored line represents a different participant, while the black line is a second order polynomial regression line. Figure 6.8 illustrates how *POI* participants begin with a low map coverage, but ends up with higher coverage than *Best-First* participants. *coverage* in the second environment shows similar behavior, although not to the same extent as the first.

coverage results for the first environment are quite surprising, considering the nature of the *Best-First* implementation (show the image with the largest area that wasn't already seen). One explanation for this finding is, that *Best-First* participants spent more time on each image, thus making a slower progress.

6.4 Interface usage

Figure 6.10 shows what UI components participants chose in both of the environments, in order to navigate between images. As a reminder *POI* participants navigated between images using the map, and could choose to refine the selection using the map or "Relevant Images" (thumbnails) display. *Best-First* participants used the "Next/Prev" buttons and the thumbnails display. Note that thumbnails display is used differently across the interfaces, and therefore they interfaces cannot be compared in this case.

Participants in the *POI* condition relied mostly on the map, using it for selecting images more than 80 percent of the time on average. This might suggest that in most cases the automated ranking

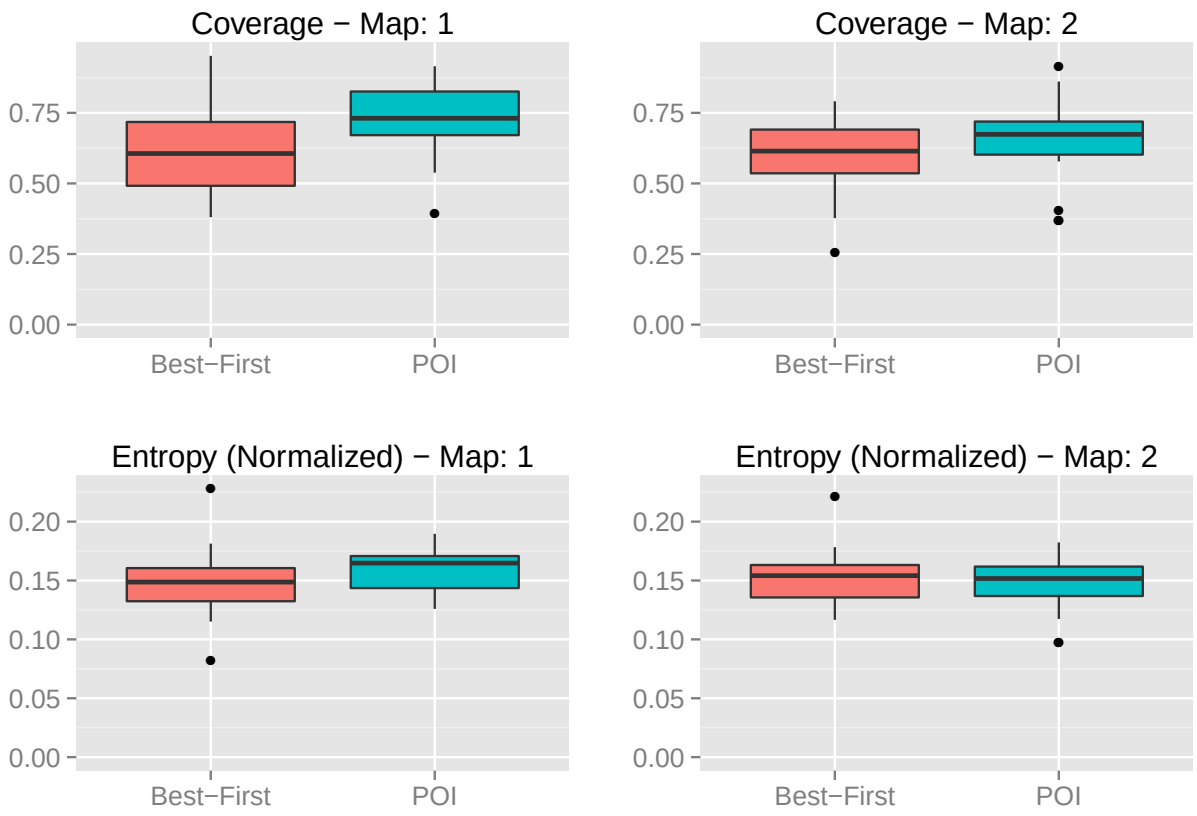


Figure 6.7 Summary of coverage quality measures.

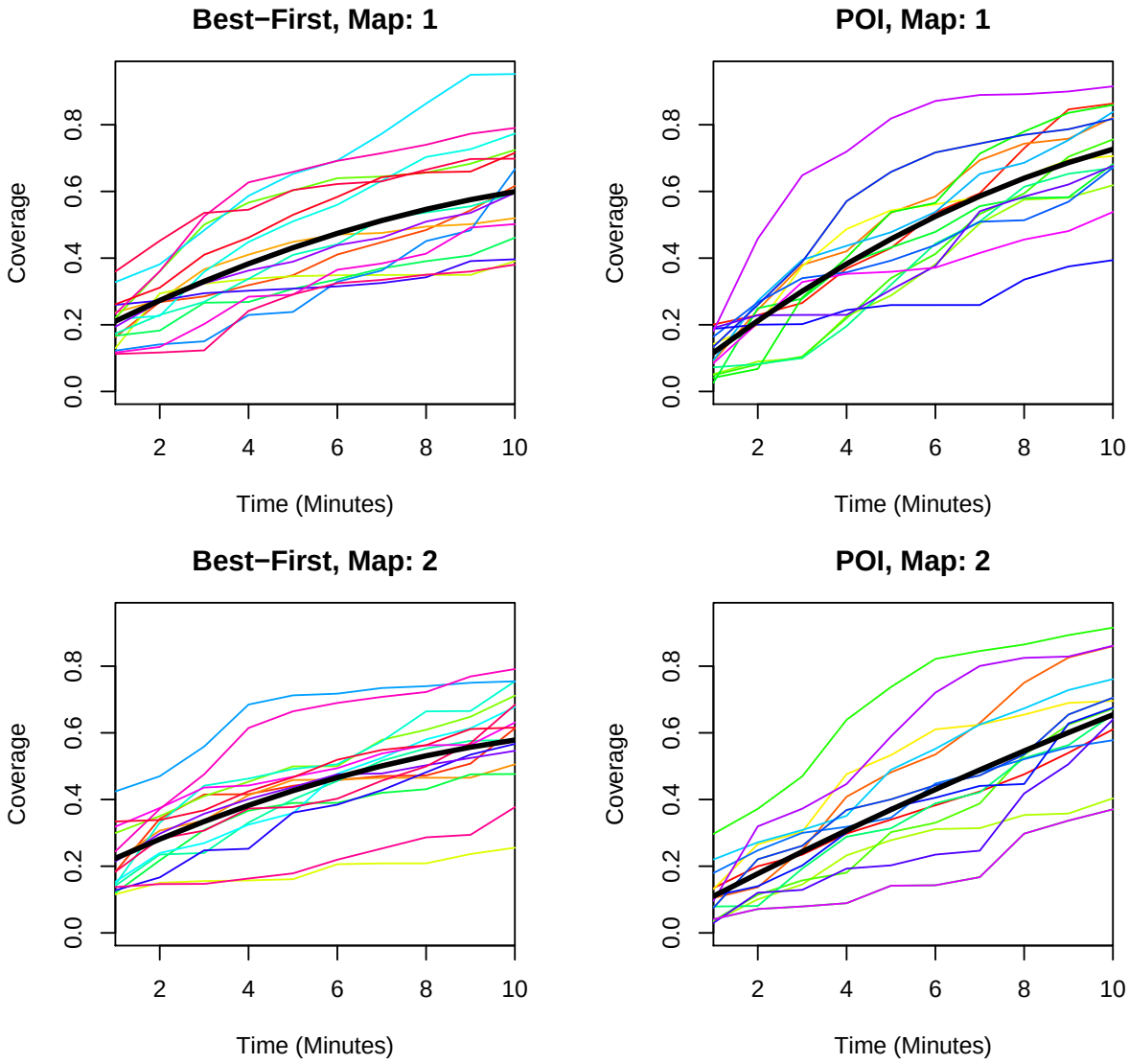


Figure 6.8 Coverage over session time.

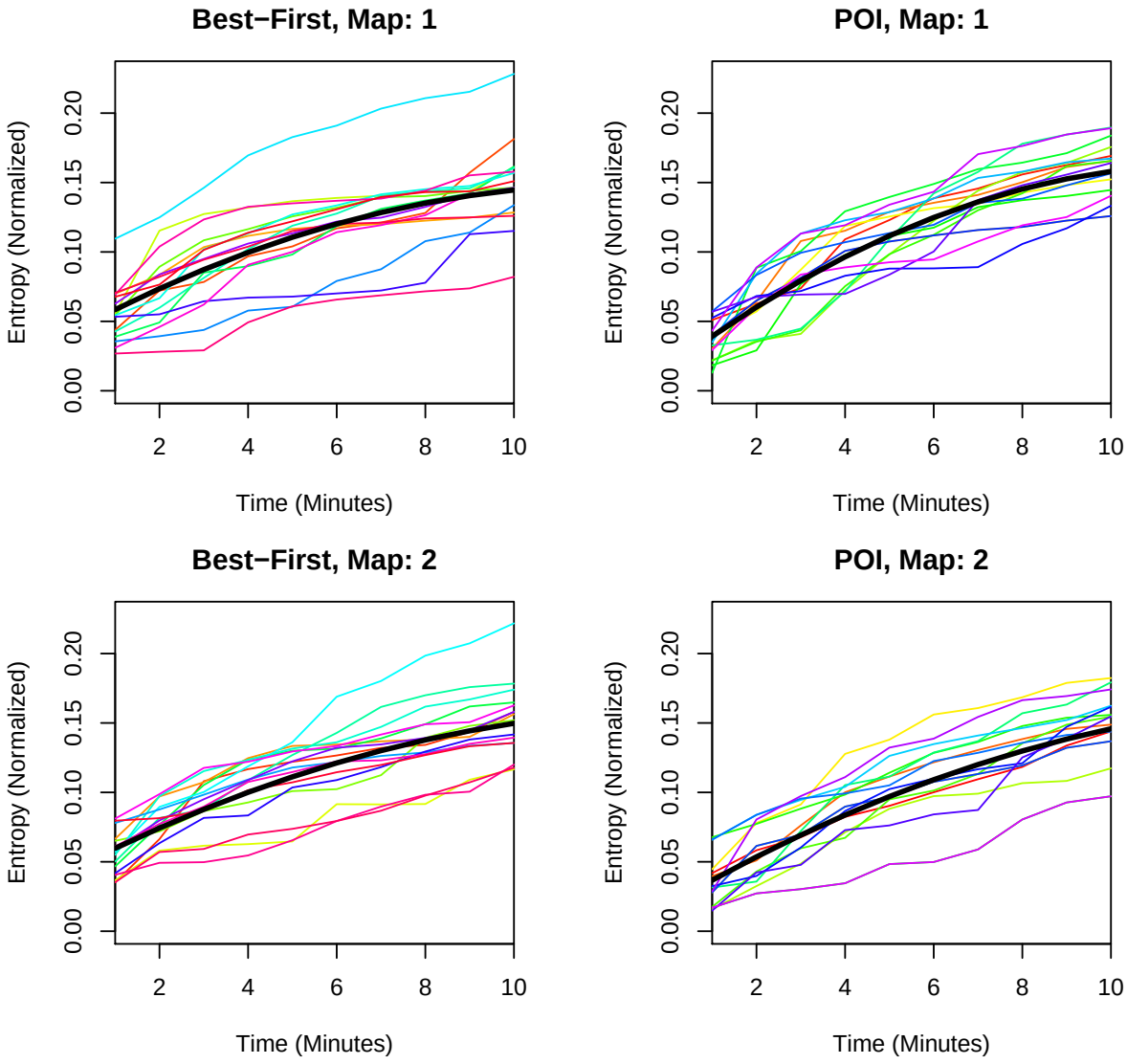
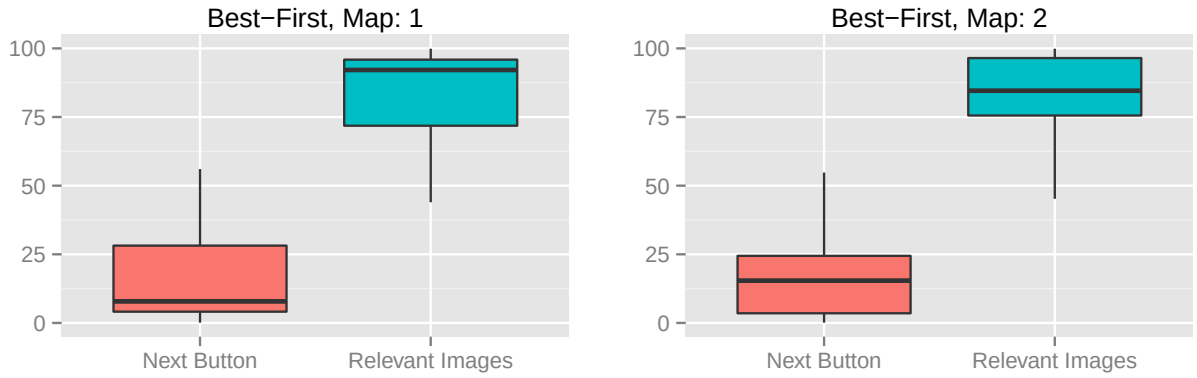
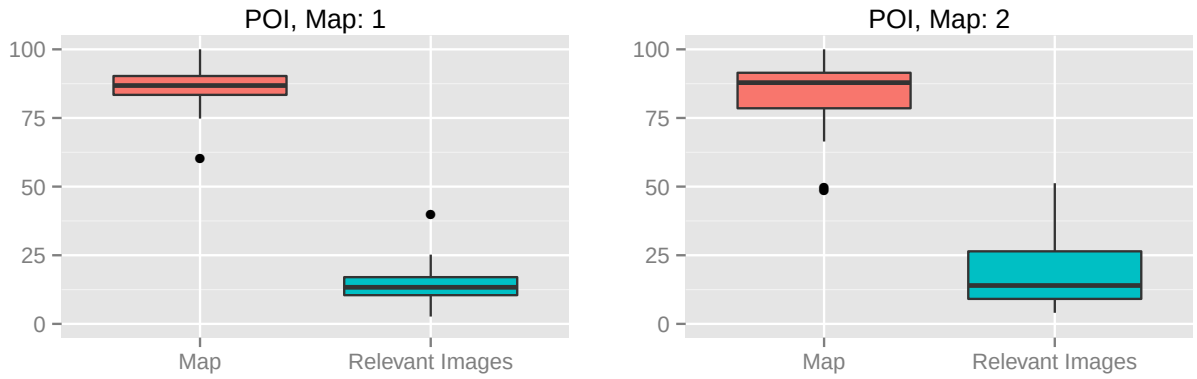


Figure 6.9 Entropy over session time.



(a) *Best-First* participants interface usage for image selection.



(b) *POI* participants interface usage for image selection.

Figure 6.10 Interface usage for image selection.

process produced adequate results (and thus there was no need to refine the selection), or that users preferred to refine the selection by choosing a new POI.

Participants in the *Best-First* condition relied mostly on the "Relevant Images" (thumbnails) display, using it for more than 75 percent of the time on average. Closer inspection of several of the *Best-First* participants showed they used the "Next" button to navigate between large unseen areas, and the thumbnails display to search for victims locally in those areas. This might explain the interface usage results in this case.

6.5 Subjective Assessment

6.5.1 Workload

As mentioned in the previous chapter, we used the NASA-TLX [20] test in order to rate perceived workload. The test consists of six measures ("subscales"), rated on a scale of 0-100 with 5-point steps. Participants were asked to rate each task based on the six measures. Figure 6.12 shows the workload subscales in their raw form, before further computation. For all subscales lower values are better. For example, participants who reported lower *Effort* values, found the mission to require less effort.

To get an estimate of the overall workload, we used an an average of the six subscales. This is a common method of analysing the NASA-TLX test results [19]. Computed workload values from both of the environments, considering the interface in use, are presented in Figure 6.11. Again, lower values are better and represent less workload.

We found no significant difference between the interfaces in the computed workload, in any of the environments ($p > 0.76$). We did find a significant difference in the reported *Mental Demand* for the second environment, with an average of 83.1 for *Best-First* participants and 74.3 for *POI* participants $(t(30) = 2.59, p < 0.014)$. *Mental Demand* measures the amount of mental and perceptual activity that was required for performing the task. We found no significant difference for any of the other measures ($p > 0.25$).

6.5.2 Final questionnaire

After completing all sessions, participants were given a final subjective questionnaire to conclude their experience. Three questions were asked:

1. Describe how you performed the mission.

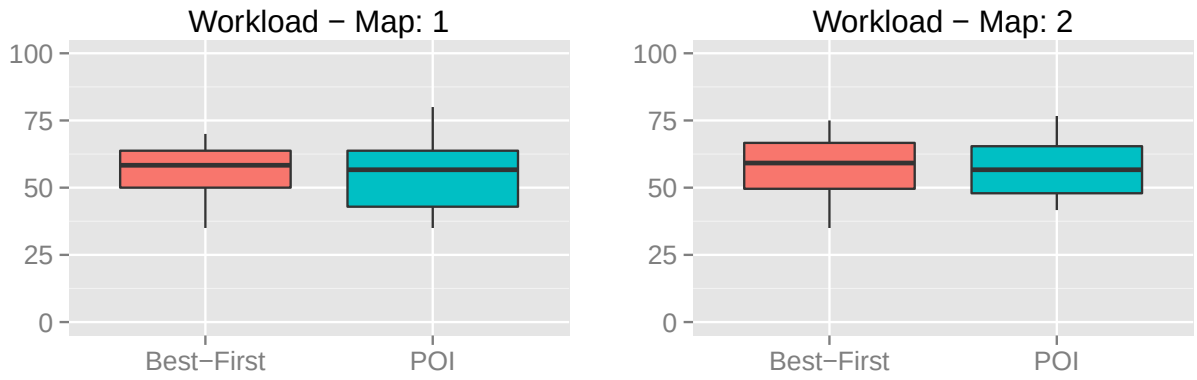


Figure 6.11 Computed workload for both of the environments.

2. Did you have any difficulties while performing the mission?

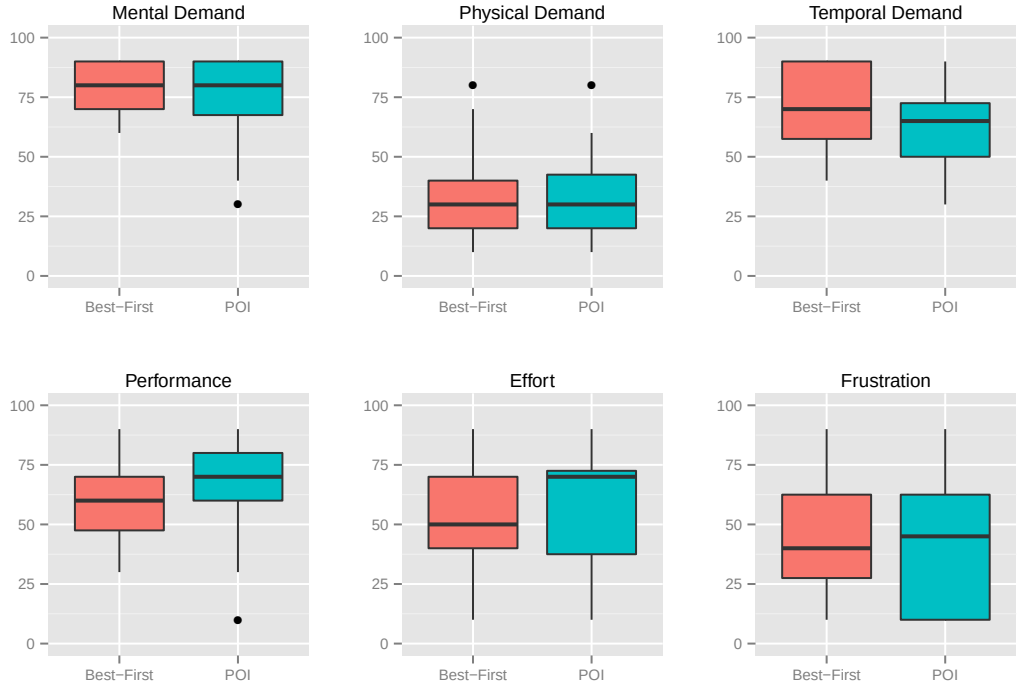
3. What was missing in the interface? how can it be improved?

Due to the open-ended nature of the questions, we received a wide range of answers. We present the common answers for both of the interfaces below.

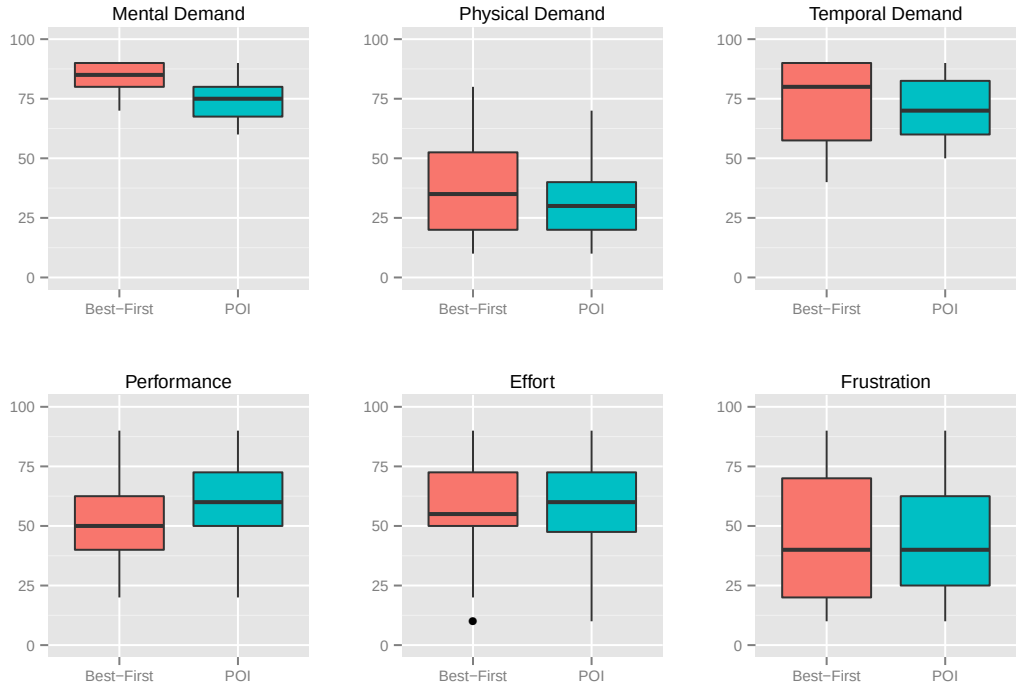
How did you perform the mission?

Most *POI* participants reported their map coverage strategy. They covered the map by either going room-by-room or by simply clicking on unvisited areas. Some participants tried to scan large areas first. Many participants mentioned observing locations from multiple point of views, in order to determine whether a victim is present.

Best-First participants reported using the "Next" button to switch between rooms or other unseen areas, and the thumbnails to search for victims in those areas. Some participants verified their marks by looking for additional images of the same location, while others focused and performing the mission quickly. Most participants mentioned that they had to perform the mission in a hurry.



(a) Environment 1



(b) Environment 2

Figure 6.12 Reported workload subscales for both of the environments.

Did you have any difficulties?

Several difficulties were mentioned for both of the interfaces. Some participants had troubles understanding which map area correspond to a certain part of an image. Some found it difficult to stay focused and concentrated. In addition, participants mentioned being in a hurry and how that might have affected their performance.

Some *POI* participants reported it took them some time to understand how to refine the automated selection using the map. Many *Best-First* participants reported having troubles to determine a victim position accurately.

What was missing and what could be improved?

Several participants who used one of the interfaces, suggested allowing users to zoom into images (and not just the map) and provide better images by placing the camera higher.

Several common suggestions by *Best-First* participants, were to make the map easier to understand, and to suggest which areas might contain a victim. They also suggested to allow users to mark visited areas, and to allow rotating the view in 360 degrees. Some *POI* participants suggested to allow navigating between images continuously (rather than one-by-one by clicking), and others suggested to mark victims on the image display (rather than on the map).

6.6 Discussion

At first, we were perplexed by the results presented in Section 6.2. Although the *POI* interface proved superior to the *Best-First* interface, we initially hypothesized that both interfaces will perform better in the second (larger) environment than in the first, since victims were more visible (recall Table 5.1). In contrast to our expectations, we found the *POI* interface to have a very similar performance in both of the environments, and the *Best-First* interface to perform worse in the

second environment.

We suspected that victims visibility in the environments had no effect on mission performance. Rather, it was the size of the environment that was affecting *Best-First*, and not-affecting *POI*. To support this claim, a simple experiment was conducted using the same data set. The experiment is described in the next chapter.

Chapter 7

Complementary Experimental Results

In this chapter, we present additional experimental results that fall outside the scope of the initial interface evaluation experiment, reported in Chapter 6. In Section 7.1, we describe an experiment that was conducted to test whether an image victim visibility, has an effect on the ability to quickly determine whether a victim exists in that image. Section 7.2 describes another experiment, that compared the performance of three methods for finding all polygons that contain a certain point.

7.1 Image visibility experiment

In the previous chapter we presented unexpected results of the *Best-First* interface, and suggested that they might be related to the map size, rather than victim visibility.

Our goal is to demonstrate that on our data set, visibility is not a factor, i.e., that the human operators are able to quickly reach a correct decision on the visibility of a victim in the image, essentially regardless of the size of the victim's body part within the image. For this purpose, a simple experiment was conducted using the same data set as in the interface evaluation experiment.

This experiment used images from both of the environments. We selected images from four visibility categories: *High* (a victim is clearly visible in image), *Medium*, *Low* (a fraction of body

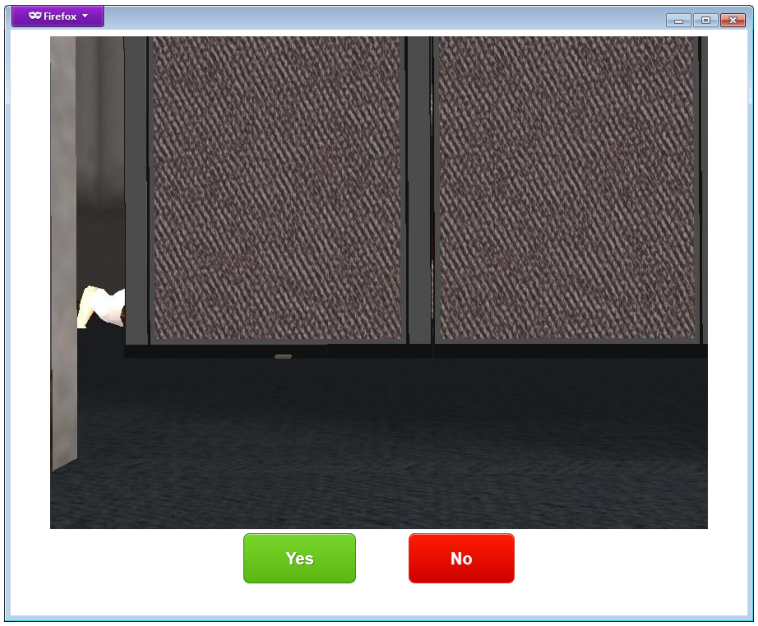


Figure 7.1 Interface for the visibility experiment. Participants were instructed to press "Yes" if there was a visible victim in the image, or "No" otherwise. The presented image is considered to have *Low* visibility.

part in image) and *Empty* (no victim in image). 16 participants were recruited to perform the experiment online, using a web browser. Participants were presented with a series of images, and had to decide for each image whether it contains a victim or not. Figure 7.1 shows the interface that was used in the experiment. In total, 70 images were presented, with the first 20 used for training purposes.

Table 7.1 shows the mean number of errors and mean time spent on images, for each of the categories. In general, the mean error rate was low for all categories. Images from the *High* category required less time to handle, and the mean time difference from *Empty* images was 240 milliseconds.

Results suggest that visibility indeed had no effect on mission performance, and that we can attribute performance difference of *Best-First* between the first and second environment, to the different map sizes.

	Empty	Low	Medium	High
Mean percentage of errors	1%	2%	1%	0%
Mean time (seconds)	1.31	1.26	1.20	1.07

Table 7.1 Visibility - errors and time per image

7.2 Point in polygon performance

In Section 4.2.1 we described three methods for implementing the "point in polygon" query. These methods find all polygons that contain a certain point, the user-selected POI. We have conducted an experiment to evaluate the performance of these methods on our data set. The evaluated methods:

1. Naïve - run CN (Crossing-Number algorithm) on all polygons.
2. Bounding Rectangle - for all polygons, test if the POI is contained in the bounding rectangle, and run CN if it is.
3. R-Tree - search the tree for all polygons where the POI is contained in the bounding rectangle, then run CN.

The test data consisted of polygons from both of the environments, and all POIs that were selected by participants during the interface evaluation experiment. Obviously, the POIs originated only from participants who used the *POI* interface. Table 7.2 contains a summary of the test data. Each POI was tested for containment in all of the polygons. The test hardware consisted of a desktop computer containing AMD Athlon II X4 630 CPU and Random Access Memory (RAM) in size of 8 GB. Although this CPU model has four cores, only one of them was used.

Methods 1 and 2 were implemented in the same manner as in the *POI* interface. The R-Tree method (3) was implemented using the NetTopologySuite software library, a port of the JTS (JTS Topology Suite) library to the C# language. JTS is a popular Java software library that provides

	Polygons	Points of interest
Environment 1	3205	3401
Environment 2	4579	2897

Table 7.2 Data for the point in polygon experiment

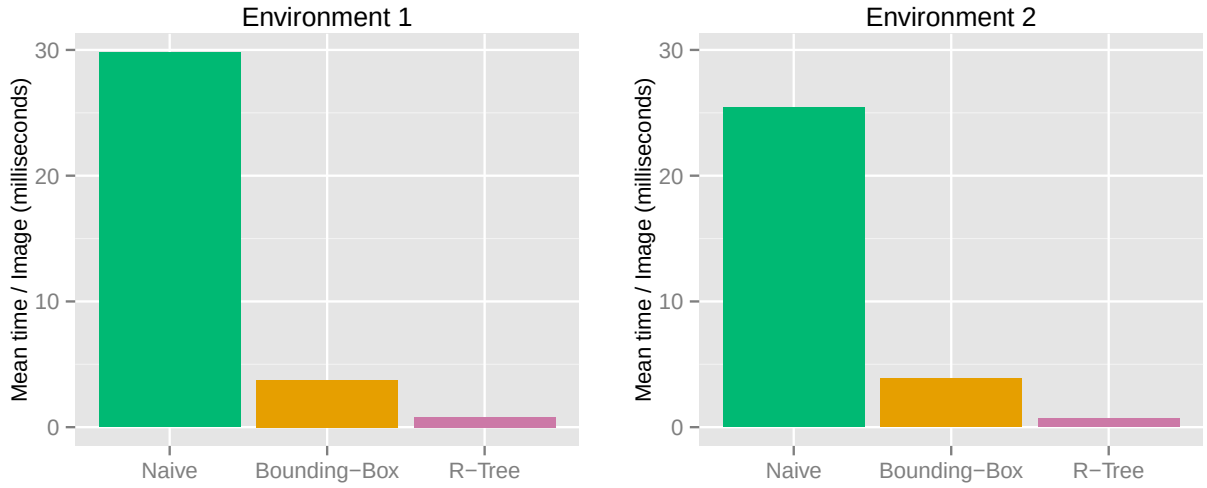


Figure 7.2 Point in polygon performance in each of the environments.

implementations of 2D spatial algorithms. In particular, JTS provides an R-Tree implementation which we found adequate for this experiment.

Figure 7.2 shows the mean execution time of the three methods on the test data, averaged over ten runs. Results show that the Bounding Rectangle method runs faster than the Naïve approach in almost an order of magnitude. The R-Tree method performed better than the Bounding Rectangle method, although not to the same extent.

With a mean execution time of less than 4 milliseconds, the bounding rectangle method can be considered to perform fast enough on our data set. In other words, there was no need to implement the R-Tree method, as the gain in execution time would have a negligible effect on the performance of *POI* users.

Note that it is unnecessary to consider the variance in execution time of the bounding rectangle method, as it always runs on all polygons and performs the same tests. In other words, the bounding rectangle method performs similarly regardless of the POI.

Chapter 8

Conclusions and Future Work

This chapter contains a summary of the presented work (Section 8.1), presents several possible design extensions (Section 8.2) and discusses suggestions for future work (Section 8.3).

8.1 Conclusions

We have presented a user interface for multi-robot search missions. The interface applies a novel operation scheme, in which the user determines how to view recorded images by selecting points of interest on a map. Images are grouped into sectors, and then ranked by utility value. The interface was compared to the current state of the art, and was found to perform better under certain conditions, in terms of higher number of found victims.

We find several advantages to the *POI* method. It allows for more freedom in image navigation and selection, providing users with the ability to control mission progress. It is also less sensitive to changes in map size, achieving consistent results regardless of map size.

However, there are also a few disadvantages. Participants results were highly variable over time, making it harder to predict mission progress. Higher variance in this context, can suggest that some users were able to leverage the freedom in image selection that is provided by the *POI*

interface, while others found it to be confusing or distracting.

The interface presented in this work is not limited to robotic systems in USAR missions, but can rather be used in a wide range of applications where covering recorded imagery in space is beneficial. The robot may be replaced by another platform, for example a handheld camera. Examples of such applications may include -

- Indoor settings - inspection of a building, for collecting information about its general state, finding signs of hazardous materials or building deterioration, and more. A robot may also be used to inspect for security clearing or for finding suspicious objects.
- Outdoor settings - finding images of an arbitrary point in the world, given a database of images taken in the area (e.g., tourist images).
- Medical - find images for a target point in the body, given an x-ray or another image source.
- Other - the interface may also be used to inspect ships, gasoline tanks and other containers for external and internal damage.

In addition, we have presented methods to store and retrieve recorded images, so that the user interface may be implemented efficiently. This included methods to store recorded images in a database, finding images that cover the point of interest, and ranking multiple images that cover the same location. We found these methods to perform adequately, in terms of quality of the responses to user queries, and execution time.

8.2 Possible design extensions

This section discusses several possible design extensions to the methods presented in Chapter 4.

8.2.1 Combining sensory data

In Section 4.1 we presented a simple approach for combining image and position data, produced separately. Another method, based on the simple approach, is to add the unmatched items as additional entries to the image database. When the camera is the "fast" sensor, there will be several images for each robot location. In this case, the interface should display the additional images in some way. When the camera is the "slow" sensor", there will be several locations for the same image, and the *Image Retrieval* component should consider this and return only one image.

Moreover, when the camera is the "fast" sensor and the storage device is limited in size or performance, the number of stored images needs to be limited. The decision on how many images should be stored for each polygon can be based on several factors:

- The difference between the first and last image that are associated with the polygon, as measured by some user-specified way. Such measurements may include the amount of overlap in local features (such as those computed by standard feature detection methods), or the difference between the color histograms. The greater the difference, the more images should be stored. The images stored should be equally divided in terms of the time they were taken along the time interval in which the polygon was stored.

- Speed of movement/rotation, values that can be supplied by the robot. The greater the speed of movement, the more images should be stored for each polygon.

- Blocked areas along the diagonal - by calculating the "seen space" along the diagonal of an image (simple image processing, requires measuring the vertical distance of non-blocked part of the image along the diagonal). The greater the level of blocking, the more images should be stored for each polygon.

8.2.2 Dealing with unknown areas

Another approach to deal with unknown areas would be to find the boundary of the area, by either using the map if represented as an occupancy grid, or performing edge detection on the map image itself. Then, we have two options: we can either display all images that cover the boundary (from all sides), or display only those who cover the nearest boundary to the POI.

8.2.3 Grouping process

The grouping process can be extended. Currently the division to sectors around the POI is done in a fixed manner, regardless of the POI location and the number of images that cover it. A dynamic process could consider these attributes in order to provide better view points for different points of interest. We can consider the following attributes in order to provide better view points for the selected points of interest.

- Area - what size of area does the image covers?
- Distance - what is the image distance from the POI?
- Is the image “centered” (directed towards the POI)?
- Available data - how many images cover the POI? are there any areas with more images than others?
- Surroundings - is the POI located in a small room or a large open space?
- Visibility - are there any areas from which there is clear sight of the POI?

Each of these attributes can be used to dynamically group the images, either separately or in combination.

8.2.4 Image ranking

We suggest to improve the ranking process of our interface. In the reported experiment, a fixed set of weights were used throughout the sessions. The weights were determined in a pilot session. An alternative approach would be to adjust the weights dynamically, so that the highest-ranked images are better suited to the environment and user preferences.

We can consider two types of input sources:

1. Content-based - by considering the attributes of selected POIs and images: position, area, heading angle, and other attributes as described in Section 4.3. This way, we can understand which attributes are more important than others, and build a model of user preferences.

2. Collaborative - by allowing users to rate the images selected by the system, we can improve the above model. The ratings would apply to the same set of attributes mentioned above, thus providing another input source.

8.2.5 3D exploration

The user interface, image database and retrieval methods that were described so far, are all designed for 2D model of the explored area. As 3D exploration methods gain more and more popularity, we describe how to adjust our interface to use a 3D map model.

User interface The user interface should be able to display 2D images that match a POI on a 3D map (model). We suggest the following view layout, when changing the point of view in one view will affects all other views. This view layout is very popular in 3D applications, in particular CAD applications.

- Front, Side, Top - display the mapped environment using an orthographic projection. The user is able to perform the same operations as in the 2D map (drag, zoom, select).

- Freeform - display the mapped environment using a perspective (realistic) projection. In addition to the operations of the other views, the user can rotate the point of view.

Image database The image database will now store the robot location and 3D orientation (heading, pitch, roll), rather than just the heading. In addition, FOV polyhedrons should be stored instead of FOV polygon.

Point in polyhedron The point in polygon methods should be adjusted to search in polyhedrons, instead polygons. The Bounding Rectangle method can simply use a bounding *box* instead of a rectangle. Spatial access methods can also be used, either with R-Tree (which supports higher dimensions than 2D), or other data structures such as Octree and binary space partitioning (BSP).

8.3 Future Work

In the future, we would like to evaluate our system in settings where images are added at real-time. We suggest to evaluate the *POI* and *Best-First* interfaces when combined together, to see if they complement each other and if results are improved. We would also like to improve to storage and retrieval methods, to support massive amount of images. The R-Tree method is a first step in this direction, but other methods may be better for our use case. We suggest to improve the ranking and grouping process, which currently does not consider many environmental factors that can be used to improved the results.

Another aspect to investigate is training, especially with users that do not perform well with the *POI* interface. Can we improve the training method, for example by providing users with a systematic method for covering the map, and improve results by doing that?

Bibliography

[1] EyeDrive. http://www.odfopt.com/eyedrive/Eye_Drive_home.htm, 2013.

[2] Karto SDK. http://www.kartorobotics.com, 2013.

[3] D. Apostolopoulos, L. Pedersen, B. Shamah, K. Shillcutt, M. Wagner, and W. Whittaker. Robotic antarctic meteorite search: Outcomes. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 4174–4179, 2001.

[4] N. Brooks, P. Scerri, K. Sycara, H. Wang, S.-Y. Chien, and M. Lewis. Asynchronous control with ATR for large robot teams. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 55(1):444 –448, 2011.

[5] D. J. Bruemmer, D. A. Few, M. C. Walton, R. L. Boring, J. L. Marble, C. W. Nielsen, and J. Garner. "Turn off the television!": Real-world robotic exploration experiments with a virtual 3-d display. In *Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences - Volume 09*, page 296.1. IEEE Computer Society, 2005.

[6] E. A. Bustamante and R. D. Spain. Measurement invariance of the nasa TLX. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 52(19):1522–1526, Sept. 2008.

[7] S. Carpin, T. Stoyanov, and Y. Nevatia. Quantitative assessments of USARSim accuracy. In *Proceedings of Performance Metrics for Intelligent Systems Workshop (PerMIS '06)*, 2006.

[8] S. Carpin, J. Wang, M. Lewis, A. Birk, and A. Jacoff. High fidelity tools for rescue robotics: Results and perspectives. In A. Bredenfeld, A. Jacoff, I. Noda, and Y. Takahashi, editors, *RoboCup 2005: Robot Soccer World Cup IX*, number 4020 in Lecture Notes in Computer Science, pages 301–311. Springer Berlin Heidelberg, Jan. 2006.

[9] J. Casper and R. Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 33(3):367–385, 2003.

[10] P. C. Cozby. *Methods in Behavioral Research*. McGraw-Hill, Nov. 2008.

[11] J. Crandall, M. Goodrich, D. Olsen, and C. Nielsen. Validating human-robot interaction schemes in multitasking environments. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 35(4):438–449, 2005.

[12] R. P. Darken, K. Kempster, M. K. Kempster, and B. Peterson. Effects of streaming video quality of service on spatial comprehension in a reconnaisance task. In *Proceedings of the meeting of I/ITSEC*, Orlando, FL, 2001.

[13] F. Driewer, M. Sauer, and K. Schilling. Design and evaluation of an user interface for the coordination of a group of mobile robots. In *Proceedings of the 17th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 237–242, 2008.

[14] J. L. Drury, B. Keyes, and H. A. Yanco. LASSOing HRI: analyzing situation awareness in map-centric and video-centric interfaces. In *Proceeding of the ACM/IEEE international conference on Human-Robot Interaction - HRI '07*, page 279, Arlington, Virginia, USA, 2007.

[15] H. Durrant-Whyte and T. Bailey. Simultaneous localisation and mapping (SLAM): part i the essential algorithms. *IEEE Robotics And Automation Magazine*, 2:2006, 2006.

[16] M. A. Goodrich and A. C. Schultz. Human-robot interaction: A survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275, 2007.

[17] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *SIGMOD '84: Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57, Boston, Massachusetts, 1984.

[18] E. Haines. Point in polygon strategies. In P. S. Heckbert, editor, *Graphics gems IV*, pages 24–46. Academic Press Professional, Inc., San Diego, CA, USA, 1994.

[19] S. G. Hart. Nasa-task load index (NASA-TLX); 20 years later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 50(9):904–908, Oct. 2006.

[20] S. G. Hart and L. E. Staveland. Development of NASA-TLX (task load index): Results of empirical and theoretical research. In Peter A. Hancock and Najmedin Meshkati, editor, *Advances in Psychology*, volume 52, pages 139–183. North-Holland, 1988.

[21] D. F. Hougen, S. Benjaafar, J. Bonney, J. Budenske, M. Dvorak, M. L. Gini, H. French, D. G. Krantz, P. Y. Li, F. Malver, B. J. Nelson, N. Papanikolopoulos, P. E. Rybski, S. Stoeter, R. M. Voyles, and K. B. Yesin. A miniature robotic system for reconnaissance and surveillance. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 501–507, 2000.

[22] S. Hughes and M. Lewis. Task-driven camera operations for robotic exploration. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 35(4):513–522, 2005.

[23] C. M. Humphrey, C. Henk, G. Sewell, B. W. Williams, and J. A. Adams. Assessing the scalability of a multiple robot interface. In *Proceeding of the ACM/IEEE international conference on Human-Robot Interaction - HRI '07*, page 239, Arlington, Virginia, USA, 2007.

[24] S. Kosti, D. Sarne, and G. A. Kaminka. Intelligent user interface for multi-robot search [extended abstract]. In *HRI 2012 Workshop on Human-Agent-Robot-Teamwork (HART 2012)*, 2012.

[25] M. Lewis, H. Wang, S. Y. Chien, P. Velagapudi, P. Scerri, and K. Sycara. Choosing autonomy modes for multirobot search. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 52(2):225–233, May 2010.

[26] M. Montemerlo, N. Roy, and S. Thrun. Perspectives on standardization in mobile robot programming: the carnegie mellon navigation (CARMEN) toolkit. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-03)*, volume 3, pages 2436–2441 vol.3, 2003.

[27] C. W. Nielsen and M. A. Goodrich. Comparing the usefulness of video and map information in navigation tasks. In *Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-Robot Interaction - HRI '06*, page 95, Salt Lake City, Utah, USA, 2006.

[28] I. Nourbakhsh, K. Sycara, M. Koes, M. Yong, M. Lewis, and S. Burion. Human-robot teaming for search and rescue. *Pervasive Computing, IEEE*, 4(1):72–79, 2005.

[29] J. O'Rourke. *Computational Geometry in C (2nd Edition)*. Cambridge University Press, New York, NY, USA, 1998.

[30] S. D. Roth. Ray casting for modeling solids. *Computer Graphics and Image Processing*, 18(2):109–144, Feb. 1982.

[31] P. Scerri, P. Velagapudi, K. Sycara, H. Wang, S.-Y. Chien, and M. Lewis. Towards an understanding of the impact of autonomous path planning on victim search in USAR. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10)*, pages 383–388, 2010.

[32] M. Shimrat. Algorithm 112: Position of point relative to polygon. *Communications of the ACM*, 5(8), Aug. 1962.

[33] A. Valero. Evaluating a human-robot interface for exploration missions. In *Intelligent Autonomous Vehicles*, volume 7, pages 270–275, University of Salento, Lecce, Italy, Sept. 2010.

[34] P. Velagapudi, P. Scerri, K. Sycara, H. Wang, and M. Lewis. Potential scaling effects for asynchronous video in multirobot search. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, pages 35–42, Gaithersburg, Maryland, 2008. ACM.

[35] P. Velagapudi, H. Wang, P. Scerri, M. Lewis, and K. Sycara. Scaling effects for streaming video vs. static panorama in multirobot search. In *Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, pages 5874–5879, St. Louis, MO, USA, 2009. IEEE Press.

[36] P. Velagapudi, J. Wang, H. Wang, P. Scerri, M. Lewis, and K. Sycara. Synchronous vs. asynchronous video in multi-robot search. In *Proceedings of the First International Conference on Advances in Computer-Human Interaction*, pages 224–229. IEEE Computer Society, 2008.

[37] H. Wang, S. Y. Chien, M. Lewis, P. Velagapudi, P. Scerri, and K. Sycara. Human teams for large scale multirobot control. In *Proceedings of the 2009 IEEE international conference on Systems, Man and Cybernetics*, pages 1269–1274, San Antonio, TX, USA, 2009. IEEE Press.

[38] H. Wang, A. Kolling, N. Brooks, M. Lewis, and K. Sycara. Synchronous vs. asynchronous control for large robot teams. In *Proceedings of the 2011 international conference on Virtual and mixed reality: systems and applications - Volume Part II*, pages 415–424, Berlin, Heidelberg, 2011. Springer-Verlag.

[39] H. Wang, A. Kolling, N. Brooks, S. Owens, S. Abedin, P. Scerri, P.-j. Lee, S.-Y. Chien, M. Lewis, and K. Sycara. Scalable target detection for large robot teams. In *Proceedings of the 6th international conference on Human-robot interaction*, HRI '11, pages 363–370, New York, NY, USA, 2011. ACM. ACM ID: 1957792.

[40] H. Wang, M. Lewis, P. Velagapudi, P. Scerri, and K. Sycara. How search and its subtasks scale in n robots. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction - HRI '09*, page 141, La Jolla, California, USA, 2009.

[41] H. Wang, P. Velagapudi, P. Scerri, K. Sycara, and M. Lewis. Using humans as sensors in robotic search. *12th International Conference on Information Fusion FUSION 09*, pages 1249 – 1256, 2009.

[42] J. Wang and M. Lewis. Human control for cooperating robot teams. In *Proceeding of the ACM/IEEE international conference on Human-Robot Interaction - HRI '07*, page 9, Arlington, Virginia, USA, 2007.

[43] H. A. Yanco. "Where am i?" acquiring situation awareness using a remote robot platform. *In IEEE Conference On Systems, Man And Cybernetics*, pages 2835—2840, 2004.

[44] H. A. Yanco and J. L. Drury. Rescuing interfaces: A multi-year study of human-robot interaction at the AAAI robot rescue competition. *Autonomous Robots*, 22(4):333–352, Dec. 2006.

[45] H. A. Yanco, B. Keyes, J. L. Drury, C. W. Nielsen, D. A. Few, and D. J. Bruemmer. Evolving interface design for robot search tasks. *Journal of Field Robotics*, 24(8-9):779–799, Aug. 2007.

תקציר

אינטראקציית אדם-רובוט (HRI) היא חלק חשוב ממערכות רובוטיות רבות, ובפרט מערכות המיועדות למשימות חיפוש או חקירת סביבה. שיטות HRI המסורתיות לשליטה ברובוט סובלות מחסרונות, למשל הצורך לצפות באופן מתמשך בתצוגת וידאו חיה (סינכרונית). כאשר מספר הרובוטים במערכת גדל, קשה עוד יותר להשתמש בשיטות המסורתיות, במיוחד כאשר מנסים להציג תמונת וידאו ממספר רב של רובוטים.

בעבודה זו, אנו מציגים ממשק אסינכרוני ומונחה-משתמש למפעילים של מערכת רובוטית, התומך בסריקה (חיפוש) של איזור לא-מוכר. תוך התמקדות בכלי רכב קרקעיים בלתי-מאוישים ומשימות חיפוש והצלה, אנו מתארים ממשק אסינכרוני ותפיסת הפעלה חדשניים. באמצעות הממשק שלנו, המפעיל יכול לצפות בתמונות שהוקלטו, ע״י בחירת נקודות עניין על מפה וצפייה בתמונות רלוונטיות המכסות אותן. ביצענו הערכה של הממשק ע״י השוואתו לממשק אסינכרוני קיים (state-of-the-art), ומצאנו ביצועים טובים יותר בהינתן תנאים מסוימים, במושגים של מספר המטרות שנמצאו.

עבודה זו נעשתה בהדרכתם של פרופ׳ גל קמינקא וד״ר דוד סרנה מן המחלקה למדעי המחשב של אוניברסיטת בר-אילן.

Bar-Ilan University
אוניברסיטת בר-אילן

ממשק משתמש חכם עבור חיפוש מרובה רובוטים

שחר קוסטי

עבודה זו מוגשת כחלק מהדרישות לשם קבלת תואר

מוסמך במחלקה למדעי המחשב של אוניברסיטת בר-אילן

רמת-גן תשע״ג