REPORT ON DEVELOPMENT AND IMPLEMENTATION OF THE SOFTWARE FOR INTEGRATED TOPOLOGICAL MAP BUILDING WITH SIMULATION DATA AND ITS VALIDATION

COMBINED DELIVERABLE 6.2.2-6.2.3

VERSION FINAL

Editor: Lyuba Alboul
Contributors: Hussein Abdul-Rahman, Paul Haynes
Sheffield Hallam University
Julien Tharin (K-TEAM)

Summary

The GUARDIANS robot swarm is self-organising and can be seen as a hybrid of a (heterogeneous) swarm, a mobile ad-hoc network and an (evolving) topological map of the environment. The Map Building process is not a separate activity, but an inherent by-product of the GUARDIANS self-organising system. The robots are equipped with sensors and enhanced with a wireless communication network. The GUARDIANS approach to map building is a novel approach of distributing robots on site that takes advantage of a cooperating robot team (with respect to a single robot) allowing us to accurately determine robot positioning and guide deployment in a pre-determined manner. This in turn leads not only to a topological representation of the environment in the form of topological graph but also to its initial metric representation as the edges of the topological graph are assigned with lengths. This gives us a pretty good sketch of the environment which can be further developed to a full metric map and used as the basis of building ad-hoc mobile wireless communication and sensor networks. The presented algorithms take into consideration also sensor limitation.

The algorithms are tested on a group of Khepera III robots, specially upgraded to fulfill the needs of our approach. This document is an extended and updated version of the RISE 2010 paper [1].

Multi-robot team : *Self-organising system, topological representation, relative positioning, cooperative map building, ad-hoc communication network*

1 Introduction

In the GUARDIANS project[1] scenario the robots are autonomous and assist fire-fighters in search and rescue operations in an industrial warehouse in the event or danger of fire [14]. The tasks of the robots can be roughly split into two partially overlapping categories. In the first category the robots directly assist the fire-fighters, namely by guiding or accompanying a fire fighter and indicating possible obstacles and locations of danger. The second category comprises the tasks for a robot team acting without a human squad-leader, such as on site deployment, positioning as beacons and maintaining communication. The robots also need to be able to gather relevant environmental information. In this paper we focus on the tasks for GUARDIANS robots in the second category, and propose a new approach for on site deployment of robots for setting up a communication network and building a map of the environment.

Maintaining communication faces two major problems. The first is that the metal cages present in a warehouse render radio reception problematic. The second problem is that of position detection or localisation. For indoor environments GPS is not generally available and simultaneous localisation and mapping (SLAM) has to be based on other sensors. However, because of smoke conventional light based sensors may not produce useful data, in particular vision sensors such as cameras. The radio signal for the wireless

[1] GUARDIANS, Group of Unmanned Assistant Robots Deployed in Aggregative Navigation supported by Scent Detection, EU FP6 ICT 045269

communication will not be disturbed by smoke and serves as a coarse fall back. For communication various wireless technologies are available including Wireless LAN, Bluetooth and ZigBee.

In our approach we assume that the use of the LRF is still possible. Such assumption has sense even in the presence of smoke as smoke starts developing close to the ceiling and the robots involved such as Erratics or Khepera III are low or very low in height. The main use of the LRF is two-fold: for robot detection and for distance estimation, which eventually yields accurate localisation and positioning of robots; detection is backed up by wireless communication. However, the idea behind the approach is generic, and the LRF can be substituted by other sensors or sensor system, such as the system based on infrared developed in [15]. Besides the LRG and WiFi, odometry is also taken into consideration. The supplementary, more conventional use of the LRF is for obstacle avoidance and building metric maps. However, the main result of our approach is a topological representation of the environment; the robots play active roles in constructing such a representation, being 'dynamic nodes' in the representation as well as acting as landmarks. One novel feature is that the topological map is the first to be built, that can later be upgraded to a more detailed metric representation. In robotic literature topological approaches are, in general, not viewed as independent, as topological maps are built on the top of grid-based map or feature-based maps by partitioning them into coherent regions [10].

Our algorithms have been implemented in C++. We use Player/Stage middleware [18] for validation and testing in simulation environments. Player, which is a distributed device repository server for robots, sensors and actuators, can control either a real or 'simulated' robot thus allowing direct application of developed algorithms to real-life scenarios. The robots used for experimentation are Khepera III, produced by the partner K-Team. The robot has been specially upgraded for installing the LRF, Hokuyo; both models URG-04LX and URG-04LX-UG0 can be used [6]. The set-up for a real-life implementation is in the final stage of developing.

We also developed a TCP/IP suite for wireless communication, which is independent of Player.

2 Related work

The problem of global self-localisation when no *a priori* information about the environment is known, is considered as one of the most difficult in robotics. This problem is related to the famous SLAM problem of a robot simultaneously localising and building a map of the environment. Most existing work on the subject focuses on a single robot only, and the approaches are mainly probabilistic in their nature due to uncertainty.

In the last decade, several works appeared that tackle the problem of cooperative multi-robot localisation [5, 16, 8].

These works fall mostly into the three categories. The first category is related to the multi-SLAM problem when robots build the map separately and then the obtained maps are fused in a global one. The second category is related to on site robot distribution. The third category that partially overlaps with the

first two is when the advantage of the use of several robots is applied in full strength: the robots are used as temporary landmarks for mutual recognition and hence, for more accurate representation of the environment.

Our approach belongs to the third category. Our robots represent a self-organising system, and can act as permanent or temporary beacons, whilst other robots are mobile. We also achieve on site distribution consisting of nodes connected by virtual edges, thus describing the environment by means of a graph. The nodes in these graphs are either virtual or real (robots) and their positions are calculated by applying our well-defined schema of robot movement and detection. Depending on the number of robots some robots can become static beacons that can be used for building a robust ad-hoc communication network. Our approach is 'in tune' with the methods proposed in [13] and [16].

In our approach four or five robots are sufficient for accurate topological representation of the environment. Three robots are actually required in order for reliable detection of each robot position, but four robots provide a more robust system, as one robot becomes a stationary beacon and may be used as a point of reference.

We also propose a robot detection approach, based on the robot shape and size, as well as taking into consideration that robots are 'dynamic' objects. This detection system can be further improved by labelling each robot with a special retro-reflective tag that can be detected by a laser range finder by analysing the intensity of the reflected laser beam. A similar approach was used in the works of Howard et al. on multirobot simultaneous localisation and mapping (see, for example, [8]).

3 Schema of topological map building in the GUARDIANS project

3.1 A brief historic perspective

The theoretical consideration behind our approach to topological map building is described in detail in [2]. In this schema there are ten proposed layers:-

1. Initial topological layer: This layer is determined by nodes and their connections in the ad-hoc wireless communication network formed by the swarm. It represents a topological graph, edges of which reflect wireless links between robots.

2. Initial (global) metric layer: In addition to the communication capability, the ad-hoc network of the GUARDIANS can provide position data to support localization of the mobile robots and humans (relation with WP3). This layer is determined by localizing the positions of robots within the dynamic triangular network built by the swarm and adding the distance measurements to the edges. The topological graph of the previous layer will become a geometric graph, possibly with some uncertainty in some of its regions.

3. Local (metric) maps layer: Local 2D metric maps (occupancy grids, possibly irregular), obtained on

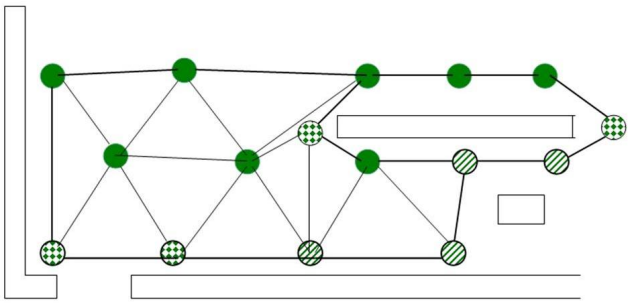


Figure 1: A full triangulation with obstacles. The dotted robots are beacons, striped robots represent possible beacons. The thicker solid lines indicate the boundary of the environment, covered by the built network.

the base of the sensorial information gathered by its node in the sensor networks (individually by each robot).

4. Skeleton layer. Here possibilities to obtain skeletons directly from the network built by robots, or to use local (global) metric maps, were planned to be explored. The two initial layers provide a partition of the environment, and this partition can be used for skeletonisation to indicate possible routes for navigation, which can significantly reduce costs and uncertainty in skeletonisation algorithms.

5. (Global) Topological map: is built by integrating structures, obtained in the previous layers, namely layers 1 and 4.

6. Global 2D metric map: is obtained by the fusion of local maps.

7. 3D local maps of the environment (combination of sensorial maps and geometric maps).

8. Global sketch of the environment : This layer is a fusion of the global topological map and local metric (mostly 2D) maps at the points of interest (such as obstacles, opening etc).

9. 3D global representation of the environment: Fusion of 3D local maps

10. Semantic layer: this layer is to enhance the maps of the environment with semantic information

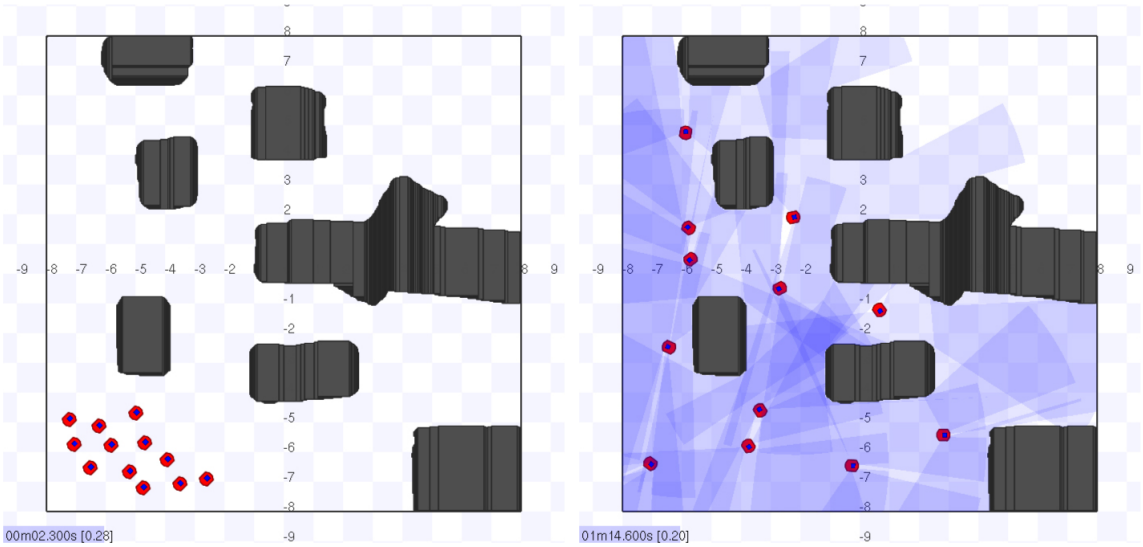
The main layers in the schema are the first two, as their construction provides enough information about the environment.

After the distribution of the robots in the environment the network layout can indicate the boundaries of the environment as well as obstacles present, as schematically represented in Fig. 1.

In this document we describe in detail how these layers can be constructed. We also give indications of how (some) higher layers can be built on the top of the initial two layers.

Our approach is a combination of guided positioning and distribution strategies. The main sensors used are the Laser Range Finder (model Hokuyo) and WiFi. Each robot, therefore, has two fields of view, one related to the LRF and the second - to WiFi. The sensing ranges of the aforementioned sensors differ considerably. The LRF Hokuyo sensing range is within 4 or 5.6 meters depending on the model, whereas the transmission range of a reliable radio signal (IEEE 802.11) is 30m or more.

Initially we thought of using only the strength of the radio signal, to distribute the robots on the site.



(a) (b)

Figure 2: Robot deployment on a site: (a) Initial configuration, (b) Final distribution

Several algorithms have been implemented with Player/Stage. The algorithms are concerned with deployment of robots/nodes as fast as possible while achieving maximised coverage and keeping the network/graph formed by robots connected. An algorithm based on the Clique intensity method [4] has been implemented and compared with the Trivial algorithm [3]. In the first algorithm wireless signal intensity is used as a rough approximation of distance to assist a large number of small robots to disperse (without knowing the relative location of neighbouring robots). This method is applicable in the 'worst case scenario', when most sensors fail, and only radio signals and tactile sensors are functional. The performance of the algorithm regarding time and area coverage was similar to the trivial algorithm for the same number of robots. The Trivial algorithm is used for area coverage benchmark but does not guarantee connection between robots.

In Figure 2 a result of the algorithm based on the clique intensity method, applied to 12 robots, is presented.

Nevertheless, despite encouraging simulation results, the distribution based solely on radio signal intensity has several flaws. The first one is that the strength of a radio signal is not a precise measure, and therefore the estimated positions might be not reliable. Another flaw is that the graph/network formed by robots is non-uniform. One reason for this is that for collision avoidance the potential artificial field method was used.

Positioning based solely on radio signal is therefore employed only as a fall back. The LRF is used as the main sensor 'responsible' for position detection now, and all our robots are equipped with the Hokyuo model of LRF.

Also as we have already mentioned, in the aforementioned algorithm we apply artificial potential forces to guide robot distribution. This method is widely used, but has the drawback of local minima which requires complex analysis. Robot distribution as a result of this method, although not strictly speaking, may appear random and unpredictable. We applied successfully an artificial potential field method for robot formation and maintenance which is related to the first category of tasks in the GUARDIANS project, and provided geometric analysis of the method. However, for our requirements one group of robots should stay together, whereas in the second category of tasks the robots should disperse in an unknown environment. For these reasons a discrete graph-theoretic approach was taken, which is completely 'in tune' with the GUARDIANS schema of topological map building. Coordination of robots whilst correcting for odometry errors then becomes more manageable and cooperative exploration algorithms have been developed.

3.2 Main concepts behind the schema

The idea to use the robots as the nodes of the graph has lead to the following strategy. The (unknown) site is initially covered by a virtual triangular grid (triangular tiling), depicted in Fig. 3

The grid can be seen as infinitely spanning in all directions, and the GUARDIANS robot, while exploring the site, will make the local part of the grid 'real'.

Each robot forms an attainable visibility domain (*AVD*), determined *a priori*. As it will become clear later, the radius of this domain is equal to at most half (or smaller, such as one third) of the sensing range of the measurement sensor to be used. In our case this sensor is the LRF, but it can be substituted by another, suitable sensor. This domain is depicted in Fig. 4.

As one can see, the domain is in the form of a hexagon, where the black node indicates the robot, and white dots are nodes in the grid that the robot can 'sense'. The 'sensed' nodes are also the grid nodes to which the robot can also move. It is not necessary that the robot visits all 'sensed' nodes, but the 'sensed' nodes mean that this part of the environment is explored. Initially, the AVD is virtual, but after robot's deployment on the site it becomes 'real' (see Fig. 5).

The actual range of the Hokuyo LRF is 240 degrees, but as the robot can turn, we consider the AVD as

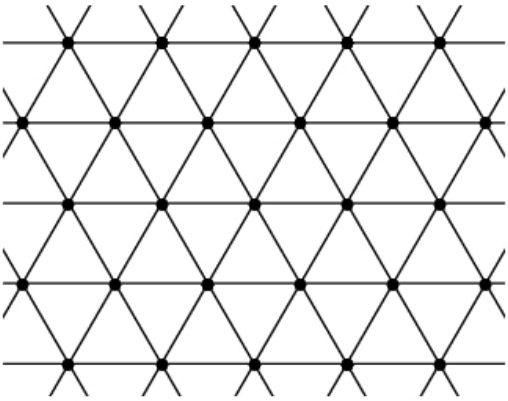


Figure 3: Virtual triangular grid.

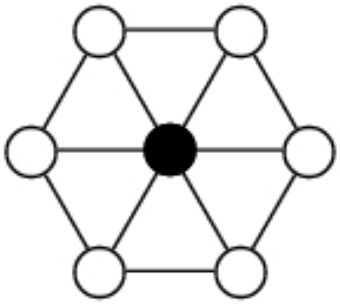


Figure 4: Visibility domain of a robot

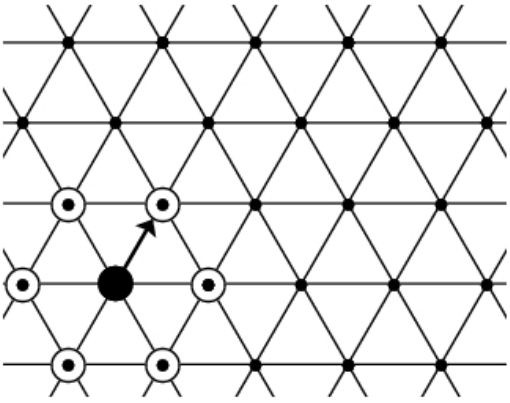


Figure 5: Domain of a robot on the site.

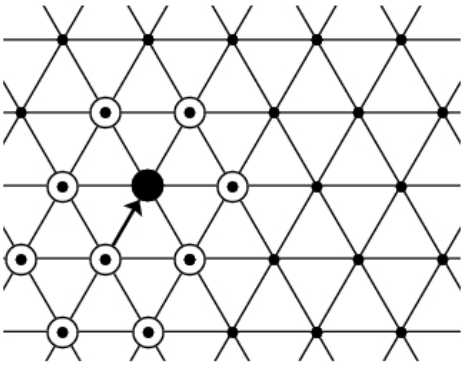


Figure 6: Explored part of the site.

360 degrees. Moreover the node behind the ‘rear’ of the robot is often the node already visited. The robot moves along the edges of the grid to available nodes, and while it moves the part of the environment visited transforms into a (topological) graph referred to as the *initial topological sub-map of the environment* or ITSM (see Fig. 6).

The robot is indicated by the black disk, and the explored (either ‘sensed’ or visited) nodes are indicated as white circles with a black dot inside.

The ITSM is being expanded while the robot moves.

If the robot encounters an obstacle, then all the incident edges of the node whose location ‘falls’ on the obstacle, are removed. However the node itself is kept as it may happen that its position will be ‘behind’ the obstacle and therefore can be explored later on. However, if the node as it will turn out later during the process of exploration, is really ‘within’ the obstacle it will be also removed from the AVD and from the ITSM (Fig. 7).

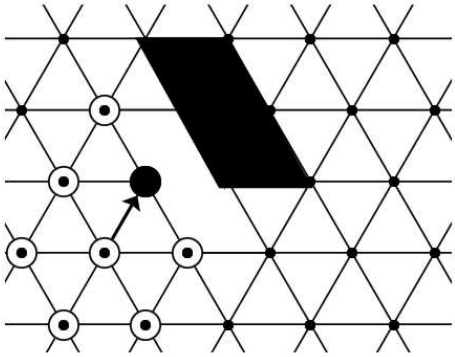


Figure 7: Explored part of the site with an obstacle present.

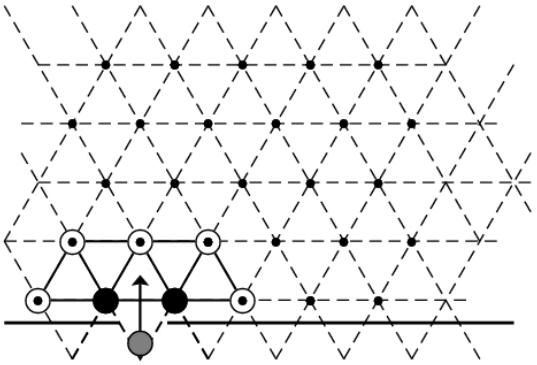


Figure 8: Robots guided distribution. Initial step.

Only those edges and vertices of the point lattice considered explored (either by occupation or laser sensing) become real (i.e physically instantiated within memory). As robots within the infinite point lattice move around they extend the real graph by a field of given radius. Importantly, the robots also use laser to detect obstructions and remove edges between vertices to indicate that robots cannot move to adjoining vertices. This is the main structure and principle governing the manoeuvrability of the collective.

3.3 Guided distribution and positioning

In the previous subsection we described the movement strategy for robots. In this subsection we upgrade this strategy to a group of robots that additionally allows for accurate robot positioning and localisation. The main step is depicted in Fig. 8.

The first two robots (black discs), take the initial positions by the entrance to the site. The distance between robots is predefined (the radius of the AVD). The third (anonymous) robot (depicted as the grey disc) enters the site and moves to the apex of the equilateral triangle, the base of which is formed by the first two robots. The first two robots are stationary and referred in what follows as *sentinels*.

Recognition of the anonymous robot is carried out by the two sentinels, following which the new robot is informed of its position (calculated relative to the sentinels), and the new robot is instructed to manoeuvre to its new position, periodically requesting laser data from the sentinels as required. The whole process operates over wireless TCP/IP communication.

The robots finally form a triangular cell, and their AVD are fused (Fig. 9).

The next step is similar in that one of the base robots will move, initially on command of the new sentinels, and then autonomously on its own (requesting sentinel laser data when necessary as before). Decision over which sentinel will move next depends on collision information gathered by the sentinels and relies on the general underlying algorithm (see section 4.4) Suppose it is the left sentinel. Now the previous

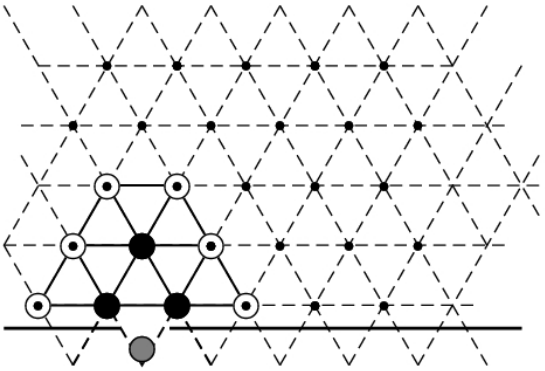


Figure 9: Robots guided distribution. Forming a triangular cell.

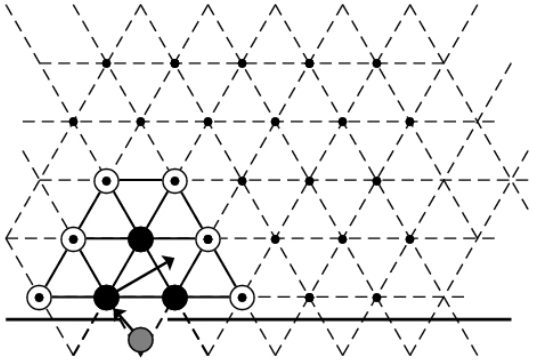


Figure 10: Robots guided distribution. Next step.

right sentinel and the apex robot become the new sentinels and the procedure regarding the movement of the (previous) sentinel is identical to the initial step. A new anonymous (fourth) robot assumes position of the left sentinel. This is depicted in Fig. 10

The procedure is now as follows. The robots will explore the environment by propagating the triangular cell formed by three robots either to the right, or up, depending on the structure of the environment. When the possibility to move to the right is exhausted, the robots first move up and then again, depending on the 'sensed' environment, will move again right, or go to the left. The fourth robot stays as the beacon for maintaining the wireless communication. The fifth robot can enter the site and take the initial position of the second robot. As we see the three robots can move in the described manner and 'swipe' the site while our set up ensures accuracy in robot localisation (see Fig. 11).

The map of the environment is formed by the boundary nodes of the fused AVD of the robots. If there

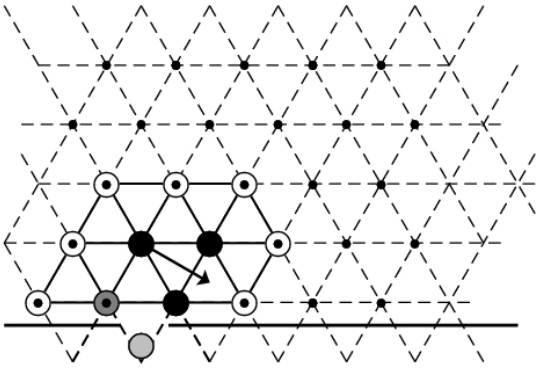


Figure 11: Robots guided distribution. Moving around.

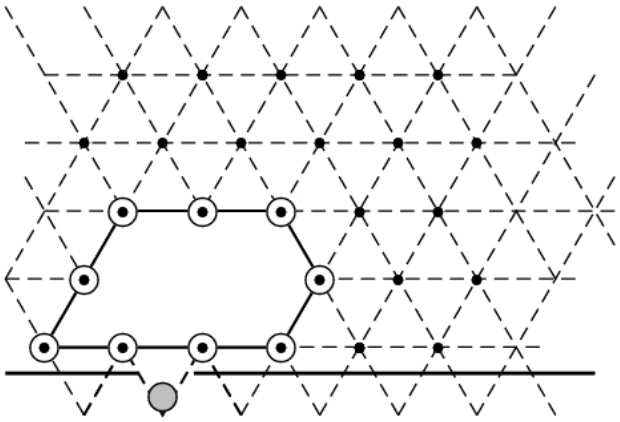


Figure 12: An example of the topological map, built by the robots.

is no obstacle, the map will represent a simple polygon. An example is given in Fig. 12.

3.4 Triangular system

The proposed triangular grid provides us also with a nice coordinate system. Each node can be addressed by two numbers: (i) the number of the horizontal line where the node is situated, and (ii) the number of the position of the node in the line (i.e. row, column coordinates). Both numbers can be negative as well. An example of labelling is given in Fig. 13.

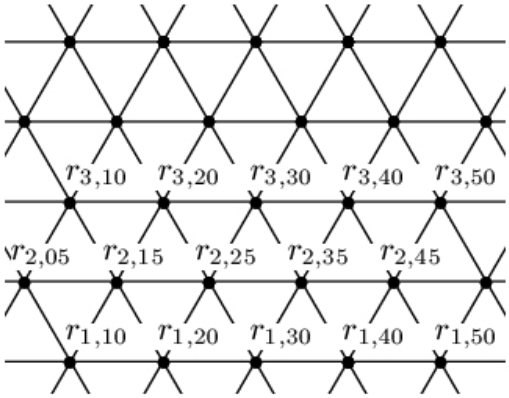


Figure 13: Grid coordinate system.

3.5 Corners and obstacles

Fig. 14 depicts the strategy when robots encounter a corner, for example. An obstruction arises when a robots laser range identifies a collision in a direction (and distance) known not to correspond to a robot within the graph. In such circumstances, the edge connecting the robots real node with the virtual node is removed. Thus, that area of the graph is no longer navigable from the robots current position (by any other robot finding itself in that position). On encountering a corner, the robot collective co-operate to propagate upwards, until they are able to move left again. This results in a snaking movement of the robots which is dependent upon the obstacles within the environment.

The general strategy for graph construction is to remove edges from the graph where collisions occur in directions known not to contain robots. Robots examine a pre-defined small range of laser data in the directions of the graph edges (where the laser range allows) in order to do this. This approach allows new robots to construct the correct graph which later robots can use to successfully navigate the environment.

4 Algorithms and its testing on simulation data

For development purposes and testing algorithms on simulation data, the popular Player/Stage open source robotics environment was employed, although later development of in-house classes may deprecate this necessity to address efficiency and speed. The purported benefits of Player/Stage are the seamless transition from simulation to practice, although preliminary investigations suggest this may not be the case. The whole system was written in C++ on Debian Linux using the Eclipse Galileo IDE, with some later usage of Windows 7 and Microsoft Visual Studio 2010 IDE (Express Edition).

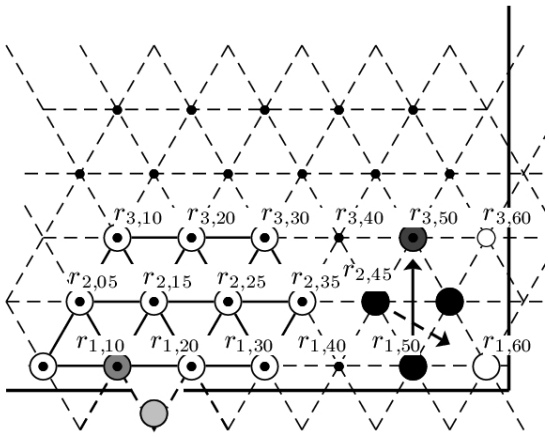


Figure 14: Corner strategy.

4.1 TCP/IP Communications suite

The backbone of any cooperative system is some means of communication, without which information could not be exchanged. Communications are achieved via a TCP/IP class written specifically for the system and based on Beej's Guide to Network Programming [20]. This is a standard TCP/IP server/client consisting of a main listener thread function. This thread listens out for and establishes new client connections and also listens for incoming messages, which it either (i) pushes to the general message queue, or (ii) acts upon directly before autonomously transmitting requested information back to the sender. The latter function of the listener thread is concerned with non-intensive trivial data processing such as position and laser range requests. This mechanism enables simple requests such as these to be processed behind the scenes leaving the main program to act upon more processor intensive data processing instigated by messages arriving at the general message queue. A non-trivial message might, for example, enter the main function into a "move robot to new position using sentinels" sub-function which would give the non-trivial function full control. This would put further non-trivial processing requests on hold (via the queue), but still allow trivial requests to be processed via the listener thread which, of course, runs all the time. It should be noted here that the Boost threading library [21] was used because, unlike the standard C "pthread" library, it is claimed to be thread-safe; the C pthread functions caused problems.

4.2 Inauguration of Robots into the Team/Collective

The presented system deals mainly with three robots working in cooperation. However, further developments will see the cooperation of many robots. To achieve this a mechanism for recognising and introducing anonymous robots into the collective is required. Consider a number of robots each having their own unique ID, and two sentinel robots whose positions and orientations are well known. When the current collective

requires addition of another robot, it must recognise the presence of the robot, mutually agree upon a position of the anonymous robot (since the new robot will be unaware of its position), and iteratively guide itself into position. This function requires communication between all three robots and can be implemented in a number of ways depending on which robot(s) are in control. The approach adopted here is for the sentinels to indicate the necessity of another robot, mutually compute its position, and then pass control over to the new robot to manoeuvre itself into position whilst periodically requesting laser range data from the sentinels to correct for errors. The pseudo code for recognizing the identity of an anonymous robot is given below:-

- From the higher-level algorithm (presented in 4.4), the collective recognises it must inaugurate a novel anonymous robot into the collective.
- The sentinels rotate such that their laser ranges begin at a 90 degree angle to the chord joining the two sentinels. The sentinels then expect any collision within a, say, 70 degree range of the laser starting indexes to be a robot. This may be thought of a detection region in which anything is classed as a robot.
- Given the known positions, orientations, and laser data of the sentinels it is possible to compute the position of the object within the detection region.
- The distances to the detected object and the reverse angle of incidence are broadcast to all anonymous robots not in the collective.
- The anonymous robot matching these parameters is that within the detection region and the anonymous robot is identified.
- The sentinels then transmit a message to the new robot telling it to move into its new position (the new position will be a node within the graph) whilst using laser to correct itself.

The aforementioned steps are illustrated by two snapshots of its implementation in PLayer/Stage, presented in Fig. 15 and Fig. 16.

4.3 Robot Manoeuvre with Error Correction

Robot manoeuvre with error correction follows robot inauguration, but is actually independent of robot inauguration. This is because manoeuvrability is also required of robots within the graph as the collective explores the graph further, hence the separate treatment in this subsection. However, the pseudo code which follows may be read as an extension to that in the previous subsection.

- On receiving the ‘manoeuvre into position’ command a robot orientates itself to the destination and travels to the new position. It should be noted that speed accelerates and decelerates according to the gradient of a parabola, thus minimizing sharp changes in speed.

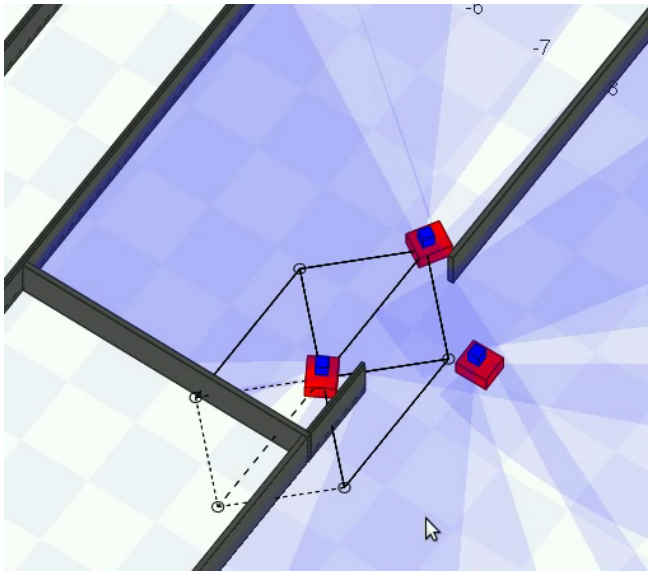


Figure 15: Robots guided distribution. Initial step. STAGE snapshot.

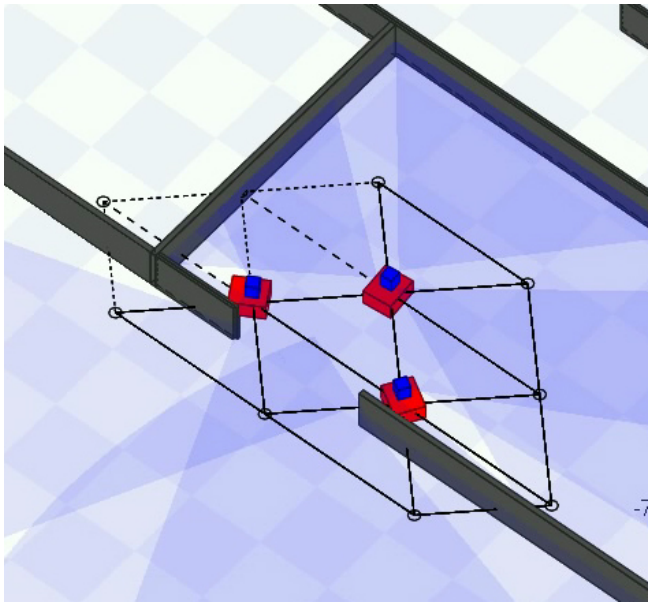


Figure 16: Robots guided distribution. Inauguration of a new robot. Final step. STAGE snapshot

- Every half distance travelled, the new robot stops and requests laser data from the sentinels between which it travels so that it can compute its inherent and expected position. If the two positions do not match, then the new robot compensates by reorientating before moving off. This step is repeated until...
- The robot reaches its destination within a pre-defined tolerance. It then transmits an arrival message back to the sentinel(s).

4.4 A General Algorithm

The aforementioned processes are mainly concerned with implementation of real robots (simulation and practice), in particular to compensate for localisation errors due to odometry. However, an overall algorithm has not yet been described with which a team of three robots can successfully explore an unknown environment. The algorithm to do this can be written as a recursive or non-recursive function. The recursive function is a little simpler, but the non-recursive version is presented here because it is the most scalable (due to storage of variables and function arguments when recurring) Three robots within the infinite triangular point lattice are allowed to move in one of six directions (depending on the presence of the corresponding edges within the graph). These directions are referred to as NORTH-EAST (NE), NORTH (N), NORTH-WEST (NW), SOUTH-WEST (SW), SOUTH (S), and SOUTH-EAST (SE). A well-founded ordering is imposed on the directions to allow precedence over others, e.g.

$$S > SE > NE > SW > NW > N.$$

Using these movements a team of three robots can fully explore any triangular point grid we are likely to encounter. The algorithm pseudo code is presented after which a short descriptive step through is given. All variables are underlined in Courier New font:

- [Start] Clear `clique_list`
- Construct list of local cliques of size 2, `clique_list` (local to the `current_node`)
- Clear `expansion_directions_list`
- Consider expansion rules for each clique in `clique_list` (with respect to the graph) and store in `expansion_directions_list`. Each element of `expansion_directions_list` contains a direction, the 'from' node, and the 'to' node $x = (dir, from, to)$
- for each `x`$\in$`expansion_direction_list`
  - if `x` compromises next move then

        * mark `to` as `visited`
        * remove `to` from `junction_stack`
        * remove `x` from `expansion_direction_list`
    - endif
- next
- if `expansion_direction_list` is *not* empty,
    - sort `expansion_direction_list` according to well founded ordering
    - actualize the expansion direction (`x`)
    - `current_node` = `to`
    - remove `to` from `junction_stack`
- else
    - at this point a dead-end has been reached. Manoeuvre to the location at the top of the `junction_stack` (removing nodes from `junction_stack` as we go)
    - `current_node` = juncture node (from top of `junction_stack`)
- endif
- goto [Start]

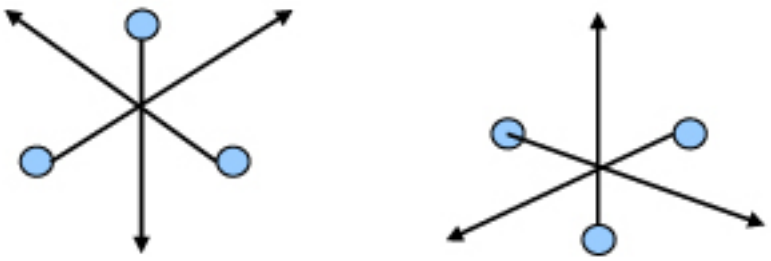


Figure 17: Possible expansion directions

The algorithm works by taking the `current_node`, and producing a list of cliques of size 2 surrounding it. A clique of size 2 consists of two nodes if those nodes contain a robot, so for a team of three robots there will be a maximum of 3 cliques of size 2. Using these cliques it is then possible, along with the local

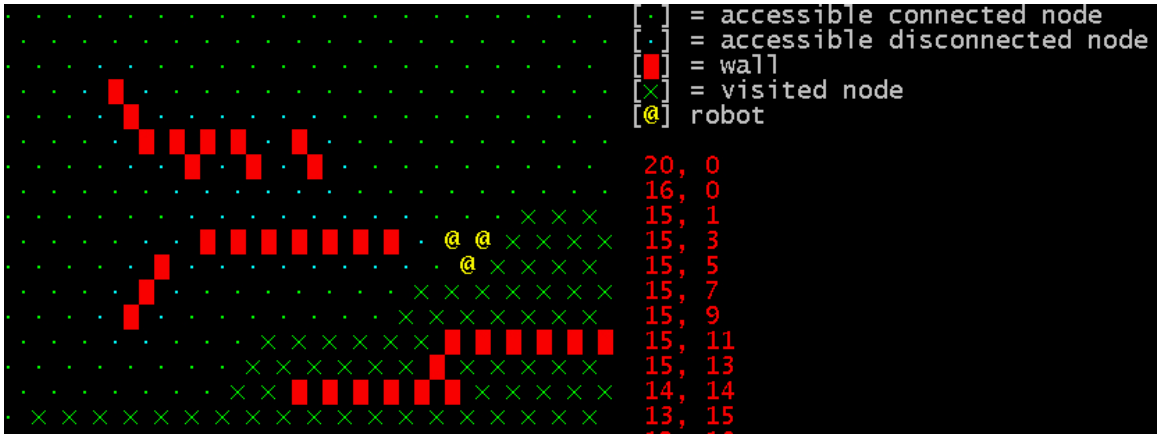


Figure 18: Robots guided distribution. Exploration of the site. Snapshot 1

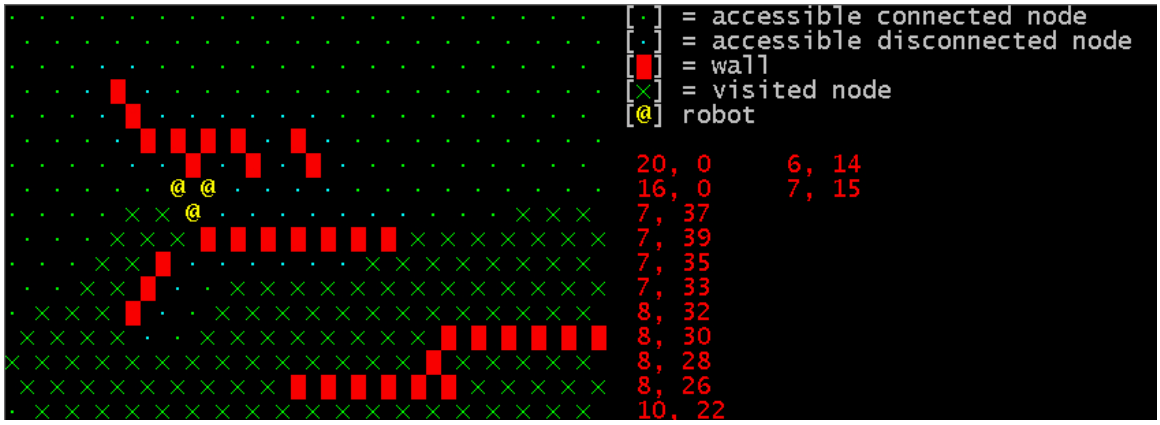


Figure 19: Robots guided distribution. Exploration of the site. Snapshot 2

graph, to determine which expansion directions are permitted for each clique. Example expansion directions are given in diagram 17 where robots are shown as blue circles.

In all there are six possible expansion potentials to choose from corresponding to N, NE, etc. The first task is to deal with compromising expansion directions, i.e. those directions for which were we to move to them would result in zero possible expansion directions. Such directions should be marked as visited and removed from the `expansion_direction_list`. The associated node is also removed from the `junction_stack` to avoid returning to that position in future.

If at this point the `expansion_direction_list` is not empty, then the priorital expansion direction is actualized, otherwise we have reached a dead end and must return to the most recent junction. Several snapshots of the algorithm implementation in simulation environment are given in Fig. 18, 19 and 20.

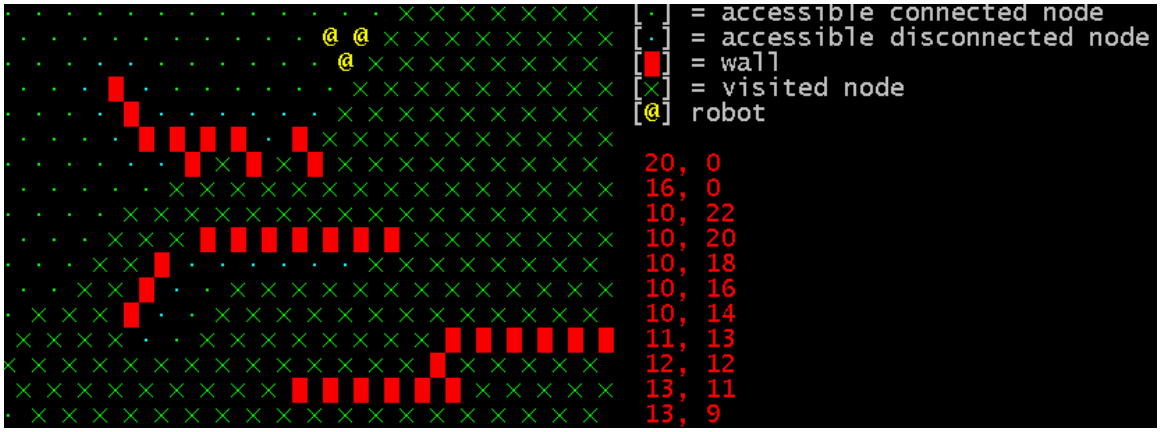


Figure 20: Robots guided distribution. Exploration of the site. Snapshot 3

Future work aims at developing this approach further to include multiple robots for which it will be necessary to *rectify* previous graph topology (according to pre-defined graph constraints) after actualizing expansion directions between cliques of robots of size three.

4.5 Building the map

Exploring the environment the robots simultaneously build not only a topological graph of the environment but also its geometric realisation. The process of building the geometric graph of the environment is depicted in Fig. 15, 16, and 21.

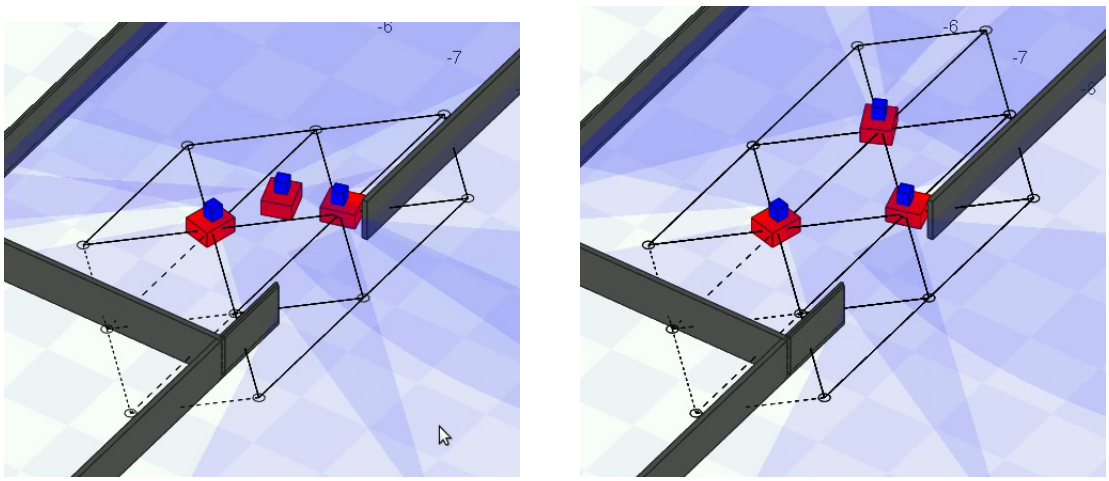


Figure 21: Robots guided distribution. Continuation of exploration and map building. STAGE snapshots.

5 Implementation on real robots

In order to test and validate our approach miniature Khepera III robots are used. The robot size allows us to test the algorithms in environments of varying complexity, but the robots are still compact enough to be used in the laboratory. On the other hand, the robots are sufficiently powerful to apply to them sophisticated algorithms. Detailed descriptions of the robots and the sensors are given below.

5.1 Robots and sensors

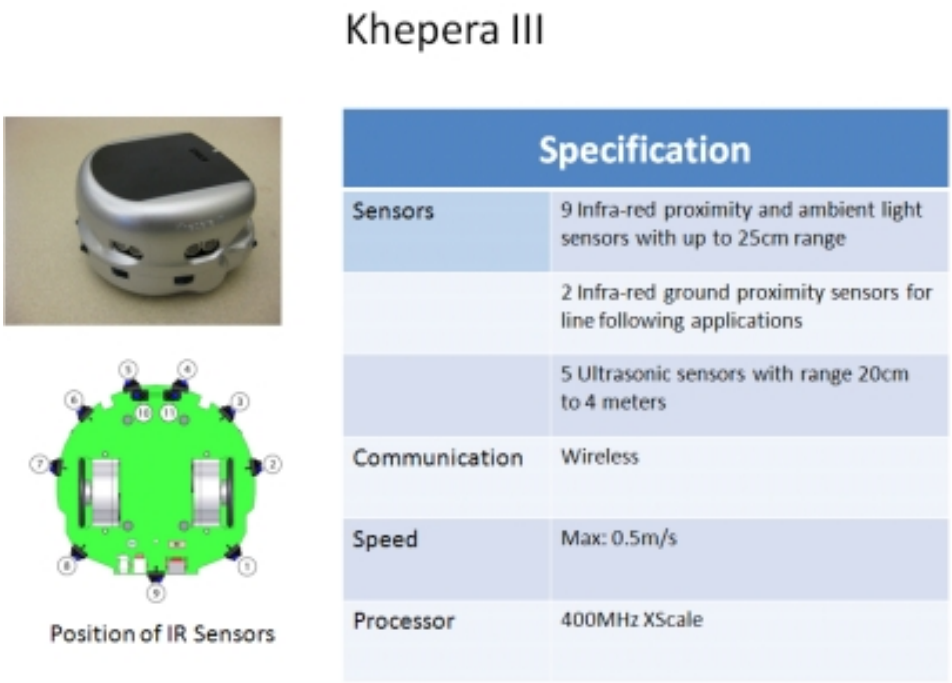


Figure 22: Khepera III specifications

The Khepera III robot [12] is based on 10 years of expertise in miniature robotics. Features available on the platform can match the performances of much bigger robots: velocity max of 5m/s for a robot diameter of 130mm. Khepera III has a weight of 690g with a payload of 2kg. Upgradable embedded computing power using the Korebot II system [11] can be added (800 MIPS 624MHz PXA270 processor, 128MB Ram, 32 MB Flash). The robot base includes an array of 9 Infrared Sensors for obstacle detection (0.5-25 cm) as well as 5 Ultrasonic Sensors for long range object detection (0.2-4.0 m). It also provides an optional front pair of ground Infrared Sensors for line following and table edge detection. The robot motor blocks are Swiss made quality mechanical parts, using very high quality DC motors for efficiency and accuracy.

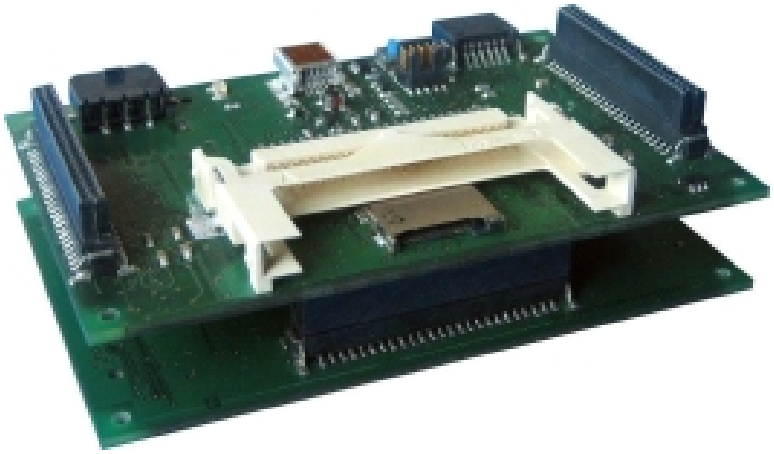


Figure 23: KoreBot II module

The replaceable battery pack system provides a unique solution for almost continuous experiments (1.5h), as an empty battery can be replaced in a couple of seconds. Khepera III robot is depicted in Fig. 22.

Through the KoreBot, the robot is also able to host standard Compact Flash extension cards, supporting WiFi, Bluetooth, extra storage space, and many others. A Player/Stage driver for this robot was developed in a previous work during the Guardians project [17].

Korebot II (see Fig. 23) uses an Arm processor which can be easily interfaced with the robot components to build a complete system and to develop a working scenario. Using KoreBot, which runs an embedded Linux distribution, the robot can perform many tasks including:-

1. Communicate with a base-station or other robot using WIFI, Ethernet or Bluetooth.
2. Operate any USB device such as a webcam, audio device, Laser Range finder, etc.
3. Install and run Linux packages such as Player.

5.2 Installation of Arm-Linux 2.6 cross-compiler and tool-chain

Installing Arm-Linux 2.6 cross-compile and the tool-chain is a key step in developing algorithms for the Khepera III robot. A cross compiler is a compiler that needs to be installed on a normal PC in order to develop and create an executable code for the Khepera III robot. So, all algorithms and programs need to be cross-compiled specifically for the robot before they can be used and run on the robot environment. A *tool chain* is the set of programs and tools that are used to create programs and packages for the robot. For example if we want to cross compile a Player package in order to install it on the robot, we have to use the tool-chain which enables us to generate robot compatible software.

Installing the cross-compiler and tool-chain require downloading and installing many packages and software. For simplicity, we address only the main software required to install the tool-chain and cross-compiler which can be obtained from:

1. *OpenEmbedded*; a software framework to create Linux-distributions aimed on embedded devices [22].
2. *Angstrom distribution*; a Linux cross compiler for embedded devices [23].
3. *Bitbake*; used to build packages, and as the basis for the OpenEmbedded project [24].

Full packages and instructions needed to install cross-compiler and tool-chain are provided by K-team on their ftp website, http://ftp.k-team.com.

5.3 Installation of Player Driver for Khepera III robot

The Player project is open source software enabling easy control of many sensors and robots. It provides a variety of functions, procedures and proxies which can read data from, send data to, and control many robots and sensors. Also, the Player project provides a TCP/IP control between a server, sensor or robot, and a client which is usually a PC or base station. Using Player means using the same set of commands for many different platforms. The only thing needed to be changed is what is called the configuration file.

The Player configuration file acts as an interface between player commands and actual hardware. In other words it interprets the Player commands to be understood for a specific hardware or platform.

Player is originally designed to be installed and run on a standard PC, and not on an embedded environment such as Khepera III robot. Therefore, a cross-compiled version of the Player project is required for the GUARDIANS project.

Using the tool-chain and cross-compiler descried earlier, along with help from K-Team S.A., Player software and Khepera configuration files were successfully installed on the robot. A full-detailed instruction of how to install Player driver is available at http://ftp.k-team.com.

5.4 LRF

Each robot is equipped with Hokuyo LRF to get information about the environment as well as for robot localisation and positioning.

Hokuyo company has laser range finder (LRF) sensors with two small models available for small mobile robotics (URG-04LX /URG-04LX-UG0, depicted in Fig. 24).

For an electrical consumption less than 0.5 Amperes at 5 Volts, a serial/usb connection for data transfer and even power supply by USB with the second model above, it provides a measurement range of 0.6-4.1 [m] on a circular aperture of 240° with a measure point every 0.36° and a 0.1s refresh rate with a range error

Figure 24: Two models of Hokuyo LRF. Left: URG-04LX, Right: URG-04LX-UG0

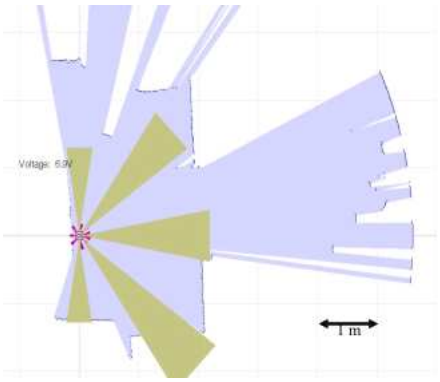


Figure 25: Khepera III and Hokuyo's LRF in Player/Stage.

of 1% resp. 3% (see Fig. 25). Hokuyo provides a software programming guide and a library for this sensor [7]. Player/Stage provides a driver for the LRF sensor named 'urglaser' [19].

An electronic board was developed for integrating LRF sensor on the Khepera III robot. Besides the communication data to the robot, it also contains a battery system for the sensor power supply with an autonomy of about 2.5 hours.

The robot can be controlled remotely by a Player/Stage client and its sensors monitored on it. A Khepera III robot equipped with a Hokuyo LRF (situated on the top) is pictured in Fig. 26. The ultrasonic sensors (range 0.2-4m) can be seen in the second from the bottom 'ring'. The IR sensors, covering the nearest range around the robot are also displayed (next to the bottom).

At the software level, several software modifications were needed to enable the robot to communicate with the new hardware. Firstly, the Player driver was updated to include the new sensor. An updated

Figure 26: Khepera III and Hokuyo's LRF.

configuration file was also needed so that all laser commands could be interpreted and sent to the sensor. These modifications were important to enable full control of the new device using the Player project software only.

For direct control of the laser sensor using the KoreBot module, cross-compiling the Hokuyo libraries was necessary. Cross-compiling the device libraries and API provided by the manufacturer are carried out by using the arm-angstorm cross-compiler and the bitbake software discussed earlier.

5.5 TCP/IP suite for communication. Real robots

In order to achieve efficient on site distribution the robots communicate by an ad-hoc wireless network. This network is built by the robots and represents a part of the self-organising GUARDIANS robotic system. Each robot is equipped with a WiFi flash card (IEEE 802.11b/g, Ambicom WL500G-CF). A purpose built threaded TCP/IP class was developed to handle communication independent of Player/Stage. This class acts as a general message and data transmission server/client, but also serves to relay routine robot queries which are often required by higher level functions. These routine functions include requests and commands for robot positioning and orientating, odometry settings, laser ranges, and so on. Automating such requests 'behind the scenes' results in cleaner and clearer implementations of the more advanced higher-level functions such as sentinel/anonymous robot identification and co-operative sentinel/robot/laser manoeuvring.

A network ID needs to be assigned to all robots and a unique IP address is given to each robot on that network. Full instructions of how to establish an ad-hoc network are available at http://ftp.k-team.com. An ad-hoc network between the robots was successfully established with the help of University of Paderborn/Germany, a partner in the GUARDIANS project.

The TCP/IP communication software has been cross-compiled so the robots can send and receive mes-

sages, share sensor information, and exchange position information

6 Experimental set-up

6.1 Robot detection and recognition

Robot recognition using the laser sensor is one of the major challenges in this project. Each robot has to identify other robots and distinguish them among obstacles that could be found around the robot. Once the robots identify other robots, one can figure out its own position based on the positions of the other robots using the triangulation method described earlier.

As mentioned earlier in this paper, the use of light based sensors, such as cameras, is not reliable due to low visibility. So, in this section, laser based robot detection is developed and tested as will be explained shortly.

Because of the small size of the LRF sensor, it will be difficult to detect such a small shape. So, an extra shape of a larger size is designed and fixed onto the robots as shown in Fig. 27.

(a) (b)

Figure 27: Khepera III robots with extra shape for laser detection (a) Front view. (b) Rear view.

Our robot detection approach is carried out using the so-called ‘two scans’ method. Firstly, the scanning robot scans the surrounding environment using its LRF sensor. Then, this robot broadcasts a message using the TCP/IP socket asking all other robots to move forward by a short distance, 10cm in our example. Afterwards, the scanning robot scans the environments for the second time. Because only the robots have moved between the two scans, the absolute difference between these scans will emphasise the location of these robots leading to detection of the robots and illumination of all obstacles.

The proposed algorithm has been tested to recognise the robots as shown in Fig. 28.

In Fig. 28 the actual experimental setup is presented; the robot near the edge of the area is the scanning

Figure 28: Experimental set up.

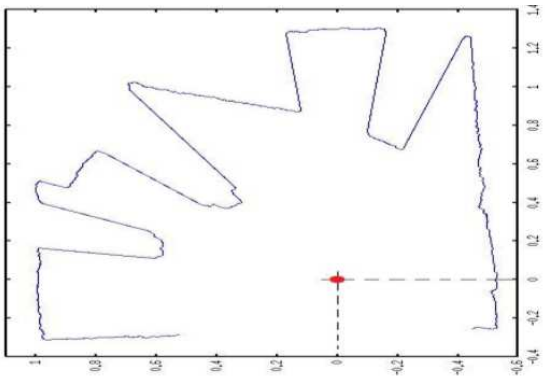


Figure 29: 2D laser scan.

robot which scans the arena to detect the other robot. Figure 29 shows a 2D laser scan of the environment (local map) of the scanning robot.

Figures 30 and 31 show the laser measurements (distances taken) of the first and second scan respectively.

Figure 32 shows the absolute difference between the two profiles shown in previous figures. Clearly, this difference eliminates all of obstacles as they remain static between the two scans. Only the difference in the positions of the robot has non zero values. Finding the robot position is carried out by selecting the central index of the non-zero values. In our particular example, the central index is found at 465 which means that the robot is located 51cm away from the scanning robot and at an angle of 40 degrees which matches the physical angle and distance between robots.

Recall that our strategy for robot movement requires that only one robot moves at a time, which simpli-

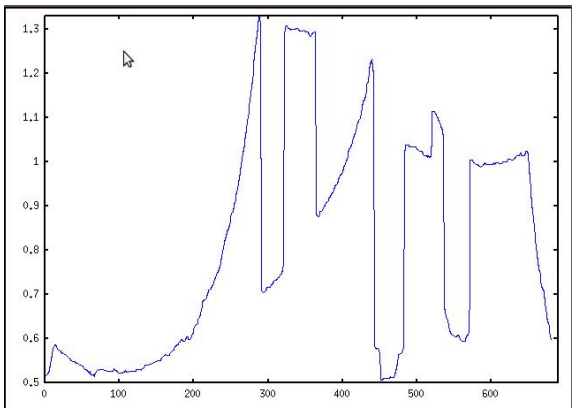


Figure 30: First scan measurements graph.

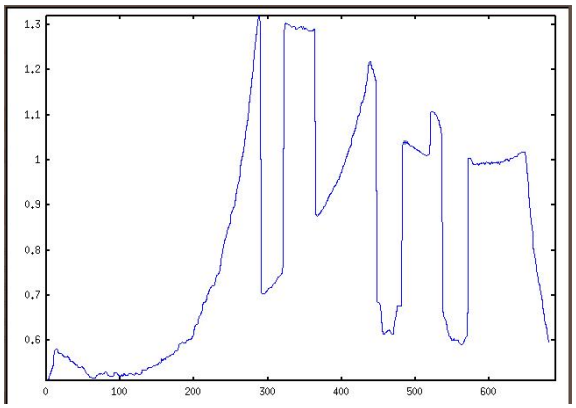


Figure 31: Second scan measurements graph.

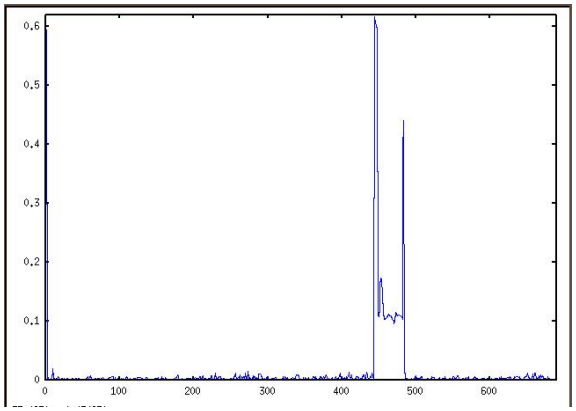


Figure 32: Difference graph of the scan measurements.

fies the recognition process even further. However, our approach is more general, and allows recognition of several robots in the vicinity of the scanning robot. Actually, the method can be used also without communication, but communication provides robustness if there are moving obstacles as well. In the latter case not only one robot scans the environment, but the other robots as well. Based on the obtained differences in the scans and communicated approximate positions each robot can detect other robots in its AVD.

7 Conclusion

We have proposed an approach that consists of cooperative positioning system (CPS) that accurately determines robot positions through cooperative control of individual robots in the group, construction of the topological map of the site and an ad-hoc wireless communication network. The topological map of the environment is obtained together with its *geometric realisation*, yielding in due course the first two layers of the proposed hierarchical schema of map building in GUARDIANS. Further layers, such as local 2D maps, can be sufficiently easily built on the top of the first two, as the positions of grid nodes are established. Skeletons can also be constructed; actually, shortest path algorithms to return to the previously visited positions represent examples of skeletons.

A practical set-up for a real-life scenario is also described and guided distribution has been demonstrated on Khepera III robots.

Current and future work is directed on tackling the following problems:

- Generalisation of guided distribution for a group of three robots to a group of n robots. Here we envisage two sub-directions: (i) one concerns developing guiding strategies of moving the group in formation, (ii) another may focus on splitting the group in several subgroups such that each subgroup would explore only a part of the environment in order to make more efficient exploration.

- At present we consider a triangular grid of a fixed size. In future we are planning to generalise the method to a flexible grid. Depending on the enviromental data obtained from robot sensors some parts of the grid can expand and others contract to accommodate better the geometry of the site.

- Improvement of guided strategy (i.e. dynamic modification of `expansion_directions`) to reduce the number of revisited nodes.

Acknowledgment

The authors would like to thank all the GUARDIANS project partners for their contribution and cooperation especially Veysel Gazi, Ulf Witkowski, Stefan Hebrechtsmeier and Mohammed El-Habbal for their valuable comments.

References

[1] L. Alboul and H. Abdul-Rahman and P. Haynes and J. Penders and J. Tharin, 'GUARDIANS multi-robot team as a self-organising system',(RISE 2010), 2010.

[2] L. Alboul, 'Conceptual Design Document on hierarchical hybrid schema for global map building and localisation', SHU, 2009.

[3] Andrew Howard, Maja J Mataric, and Gaurav S Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *6th International Symposium on Distributed Autonomous Robotics Systems (DARS02)*, Fukuoka, Japan, June 25-27 2002.

[4] Luke Ludwig and Maria Gini. *Robotic Swarm Dispersion Using Wireless Intensity Signals*, chapter Distributed Autonoumous Robotic Systems 7, pages 135–144. Springer Japan, 2007.

[5] Fox, D., Burgard, W., Kruppa, H., and Thrun, S. 'A Probabilistic Approach to Collaborative Multi-Robot Localization', Autonomous Robots, Vol. 8, No. 3, pp. 325-344, (2000).

[6] Hokuyo laser range finder:
http://www.hokuyo-aut.jp/02sensor/07scanner/
urg_04lx_ug01.html

[7] Hokuyo's URG programming guide: http://www.hokuyo-aut.jp/cgi-bin/urg_programs_en/

[8] A. Howard, M. J. Mataric, and G. S. Sukhatme, 'Localization for mobile robot teams: A distributed MLE approach', In *Experimental Robotics VIII, ser. Advanced Robotics Series*, 146–166, (2002).

[9] A. Howard and L. Kitchen. Cooperative localisation and mapping. In International Conference on Field and Service Robotics (FSR99), pp. 92Ű97, 1999.

[10] H. Huang and K. R. Beevers, 'Topological map merging', *The International Journal Robotics Research*, **24**, No.8, pp. 601-613,(2005).

[11] K-Team, Switzerland, Korebot 2 board: http://www.k-team.com/kteam/
index.php?site=1&rub=3&upPage=239&page=197&version=EN

[12] K-Team SA, Switzerland, Khepera III homepage,
http://www.k-team.com/kteam/index.php?site=1&rub=22
&page=197&version=EN

[13] R. Kurazume and S. Hirose, 'An experimental study of a cooperative positioning system', Autonomous Robots, 8(1):43Ű52, 2000.

[14] J. Penders, L. Alboul, C. Roast, and E. Cervera, ‘Robot swarming in the Guardians project’, *In ECCS’07 Proc.*, **6**, (2007).

[15] Pugh, J., Raemy, X., Favre, C., Falconi, R., and Martinoli, Alcherio, ‘A Fast On-Board Relative Positioning Module for Multi-Robot Systems’, IEEE/ASME Transactions on Mechatronics, Focused Section on Mechatronics in Multi Robot Systems, (2009).

[16] Rekleitis, I., Dudek, G. and Milios, E. ‘Multi-robot collaboration for robust exploration’, Annals of Mathematics and Artificial Intelligence, Vol. 31, pp. 7-40, (2001).

[17] Julien Tharin, ‘D4.4 Software package: Khepera III drivers for Player/Stage, User’s/Programmer’s Manual for the Software package’, K-Team SA, Switzerland

[18] Player/Stage website: http://playerstage.cvs.sourceforge.net/viewvc/playerstage/papers/

[19] Player/Stage urglaser driver: http://playerstage.sourceforge.net/doc/Player-cvs/player/group_driver_urglaser.html

[20] http://beej.us/guide/bgnet/

[21] http://www.boost.org/

[22] www.openembedded.org

[23] www.angstrom-distribution.org

[24] http://bitbake.berlios.de/manual/